Oracle® Fusion Middleware

Monitoring Oracle WebLogic Server with SNMP

14c (14.1.2.0.0)

F61310-01

December 2024

Oracle Fusion Middleware Monitoring Oracle WebLogic Server with SNMP, 14c (14.1.2.0.0)

F61310-01

Contents

Preface

1 Understanding the WebLogic Server SNMP Agents and MIB

2 Understanding WebLogic Server Notifications

Preface

This document describes monitoring the Oracle WebLogic Server with Simple Network Management Protocol (SNMP).

Audience

This document is a resource for systems administrators who use SNMP to monitor WebLogic Server. It is assumed that the reader is familiar with SNMP and general network management concepts.

The topics in this document describe the SNMP capabilities of WebLogic Server. The WebLogic Server Administration Console Online Help provides specific, task-related information on configuring SNMP services in a WebLogic Server domain.

Documentation Accessibility

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

Access to Oracle Support

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

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documentation

For information on other technologies for monitoring WebLogic Server, see the following documents:

- *Developing Custom Management Utilities Using JMX for Oracle WebLogic Server*
- *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server*

For background information on SNMP, see `com.protocols.snmp` SNMP FAQ Part 1 and Part 2.

New and Changed WebLogic Server Features

For a comprehensive listing of the new WebLogic Server features introduced in this release, see *What's New in Oracle WebLogic Server*.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1
Understanding the WebLogic Server SNMP Agents and MIB

You can use SNMP to provide monitoring data to enterprise-wide management systems.
To learn about the Simple Network Management Protocol (SNMP) management model and how WebLogic Server implements this model, see the following sections.

Overview of SNMP

With SNMP, a **manager** sends a request for information about managed resources to an **agent**. The agent gathers the requested data and returns a response. You can also configure agents to issue unsolicited reports (notifications) to managers when they detect predefined thresholds or conditions on a managed resource.

To request data about a specific managed resource, a manager must be able to uniquely identify the resource. In SNMP, each type of managed resource is described in a Management Information Base (**MIB**) as a managed object with a unique object identifier (OID). Individual organizations define their specific managed objects in **MIB modules**. Both manager and agent must have access to the same MIB module to communicate about specific managed resources.

Standards Supported by WebLogic Server SNMP

This release of WebLogic Server supports the following SNMP standards:

Table 1-1 Supported SNMP Standards

Features	Supported Standards
SNMP protocol	SNMPv3
Network protocol	UDP, TCP
Authentication protocol for SNMPv3	HMAC-MD5-96, HMAC-SHA-96
Privacy protocol for SNMPv3	• Cipher block chaining or CBC mode of the Data Encryption Standard • The Advanced Encryption Standard (AES) Cipher Algorithm.See RFC 3826 at `https://datatracker.ietf.org/doc/html/rfc3826`

WebLogic Server SNMP Agents

WebLogic Server SNMP agents query the WebLogic Server management system and communicate the results to managers over the SNMP protocol. The WebLogic Server management system exposes management data through a collection of managed beans (MBeans). When a WebLogic Server SNMP agent receives a request from a manager, it determines which MBean corresponds to the OID in the manager's request. Then it retrieves the data and wraps it in an SNMP response.

You can use WebLogic Server SNMP agents to:

- Respond to simple GET requests from an SNMP manager for the current value of WebLogic Server MBean attributes.

> **Note:**
>
> WebLogic Server does not enable SNMP managers to set the values of MBeans or invoke MBean operations. SNMP managers can be used only to monitor WebLogic Server.

- Use JMX monitors to poll WebLogic Server MBeans periodically and send notifications to SNMP managers when the MBean attributes change in a way that you specify.
- Send notifications to SNMP managers when the Administration Server or any Managed Server starts or shuts down.
- Listen for specific log messages and send notifications to SNMP managers when WebLogic Server generates them.
- Act as a proxy agent that passes requests from an SNMP manager to other (non-WebLogic) SNMP agents (such as an Oracle database agent) on the same machine.

- Organizing SNMP Agents in a Domain
- Configuring SNMP Protocols
- Configuring UDP and TCP Ports
- Narrowing the Scope of a Request
- Monitoring SNMP Agents

Organizing SNMP Agents in a Domain

In each WebLogic Server domain, you can create multiple SNMP agents and organize them into a de-centralized or centralized model for SNMP monitoring and communication:

- In a de-centralized model, you create SNMP agents on each Managed Server. SNMP managers communicate with the agents on individual Managed Servers. See Figure 1-1.
- In a centralized model, you create an SNMP agent only on the Administration Server. SNMP managers communicate only with the SNMP agent on the Administration Server and the agent gathers monitoring data from all Managed Servers in the domain. See Figure 1-2.

  This model is convenient and enables a single request to retrieve data for the entire domain, but:

- If the Administration Server is unavailable, you cannot monitor the domain through SNMP.
- If the domain is large, you must filter a large amount of data to find information about a specific resource. Instead of filtering data in the response, you can narrow the scope of the request. See Narrowing the Scope of a Request.
- The model introduces performance overhead. To gather data from all servers in a domain, the agent on the Administration Server queries MBeans in the Domain Runtime MBean server. This MBean server contains MBeans for domain-wide services and acts as a single point of access for MBeans that reside on Managed Servers. Because the Domain Runtime MBean server communicates with all Managed Servers in the domain, it is subject to network latency and increases the amount of memory that the Administration Server uses.

Figure 1-1 illustrates that when you create SNMP agents on individual servers in a domain, the agents query MBeans in the host server's Runtime MBean server. This MBean server contains only the MBeans for the individual host server.

Figure 1-1 De-Centralized Model for SNMP Monitoring and Communication

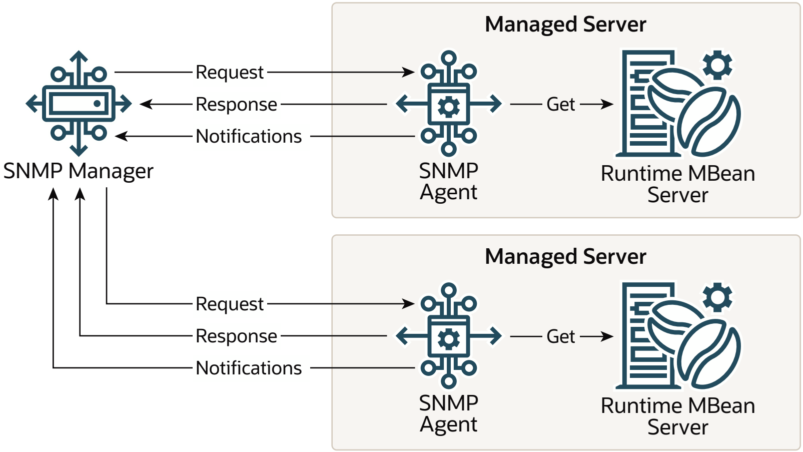
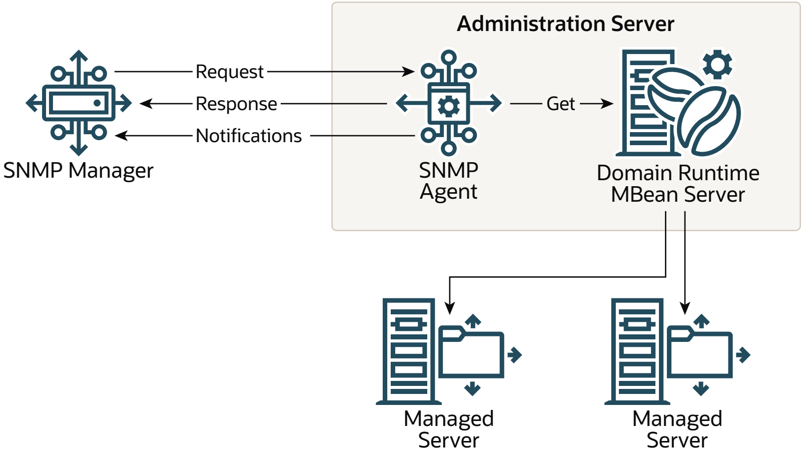


Figure 1-2 illustrates that when you use the SNMP agent on the Administration Server to retrieve data for Managed Servers, the agent queries MBeans in the Domain Runtime MBean server.

Figure 1-2 Centralized Model for SNMP Monitoring and Communication



- Domain-Scoped Agent

Domain-Scoped Agent

To support domains that were created with WebLogic Server release 9.2 and earlier, you can enable and use the domain-scoped SNMP agent instead of configuring SNMP agents on the Administration Server or Managed Servers (server SNMP agents). The domain-scoped agent offers the same features as the server SNMP agent in the centralized model described above. However, its underlying implementation is different and it will eventually be deprecated. The domain-scoped agent is overridden if you target a server SNMP agent to the Administration Server.

Configuring SNMP Protocols

A WebLogic Server SNMP agent can always communicate with managers only using the SNMPv3 protocol. While you cannot prevent an agent from receiving SNMPv3 requests, an agent processes only requests from known users that you configure through the WebLogic Server security realm.

Configuring UDP and TCP Ports

An SNMP agent communicates through a port that accepts UDP traffic and another port that accepts TCP traffic. By default, all TCP traffic uses the host server's listen port. For example, if you target this agent to a server named ManagedServer1 and ManagedServer1 listens for requests on port 7001, then the SNMP agent listens for TCP requests on port 7001. When communicating through a TCP port, WebLogic Server protects SNMP communication from denial of service (DOS) attacks.

If you want to separate SNMP TCP traffic from business traffic, you can create a custom network channel.

Narrowing the Scope of a Request

When an SNMP manager sends a request to an agent on the Administration Server, the agent's response can potentially contain data that describes multiple instances of the object. For example, the object `serverUptime` exists for each WebLogic Server instance in a domain. If a manager sends a request for `serverUptime` to an agent on an Administration Server, the response contains one `serverUptime` instance for each server in the domain.

You can narrow the scope of a request by encoding additional information in the manager's request. In a request that uses the SNMPv3 protocol, encode the name of the Managed Server in the request's context name field.

Monitoring SNMP Agents

For each SNMP agent in a domain, you can access monitoring information, such as how many notifications the agent has sent to managers and how many authentication attempts have failed, using WebLogic Scripting Tool (WLST) or a JMX client to access the new `SNMPAgentRuntimeMBean`. See SNMPAgentRuntimeMBean in *MBean Reference for Oracle WebLogic Server*. For information about using WLST, see *Understanding the WebLogic Scripting Tool*.

Security for SNMP

The security features that are available for SNMP depend on which SNMP protocol an agent uses to communicate with managers.

- Configuring Security for SNMPv3
- Invalidating the SNMPv3 Credential Cache

Configuring Security for SNMPv3

Oracle strongly recommends using the SNMPv3 protocol instead of SNMPv1 and SNMPv2.

In the SNMPv3 protocol, both SNMP agent and manager must encode identical credentials in their PDUs for the communication to succeed. The credentials include several tokens: a user name, an SNMP engine ID, an authorization protocol, and an optional privacy password, all of which are encrypted before being transported over the network.

In WebLogic Server, SNMP agents work with the domain's security realm to secure communication. The SNMP agent decodes SNMP credentials in requests and passes the SNMP user name to the security realm. The security realm maps the SNMP user name to a WebLogic Server user, authenticates the user, and authorizes access to monitoring data in the domain. To map the SNMP credentials to a user in a WebLogic Server security realm, you create a credential map.

Invalidating the SNMPv3 Credential Cache

To optimize performance, an SNMP agent caches the credential maps that correlate WebLogic Server users with SNMP credentials. To make sure that the cache contains the latest set of SNMP credentials, an agent periodically invalidates its cache. After the cache is invalidated, the next time the agent requests credentials, it regenerates its cache.

Note that making a change to the credential map does not automatically update the cache for SNMP agents. Instead, the cache is updated only after it has been invalidated. For example, if you update a privacy password in an existing entry in the SNMP credential map, SNMP agents are not aware of the new password until their caches are invalidated and regenerated. An SNMP user with the old security password can still use the agents to access WebLogic Server data until the cache is invalidated.

After you modify a credential map, you can either wait for each SNMP agent to invalidate its cache, or you can invalidate it immediately.

MIB Module for WebLogic Server

The MIB module for WebLogic Server uses Abstract Syntax Notation.1 (ASN.1) to describe the resource types that can be monitored through SNMP and the notification types that WebLogic Server SNMP agents can send to SNMP managers.

The WebLogic Server installer creates a copy of the MIB module in the following location, where *WL_HOME* represents the root installation directory WebLogic Server:

```
WL_HOME/server/lib/BEA-WEBLOGIC-MIB.asn1
```

With each new release, WebLogic Server appends any new managed objects to the module. The object identifiers for existing managed objects do not change from one release to the next.

The following sections describe the WebLogic Server MIB module:

- Hierarchical Data Model
- Object Identifiers
- Browsing the MIB

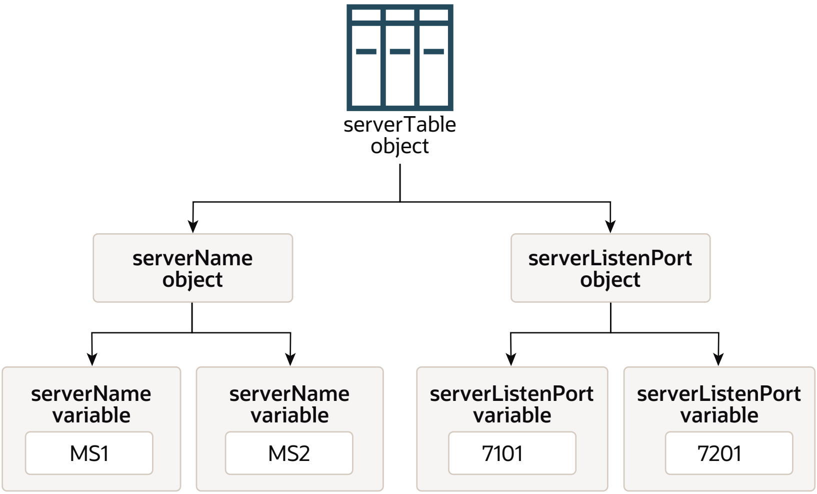
Hierarchical Data Model

WebLogic Server exposes a large number of data points in its management system. To organize this data, it provides a hierarchical data model that reflects the collection of services and resources that are available in a domain. For example, a WebLogic Server domain can contain multiple servers. Each server contains (or hosts) applications, and each application contains Web applications, EJBs, and other Jakarta EE modules.

The WebLogic Server MIB module reflects this hierarchy. For example, a WebLogic Server domain describes its overall configuration in a tabular managed object called `domainTable`. This tabular object refers to (contains) a collection of scalar objects, each of which describes some attribute of the domain. For example, `domainTable` contains a `domainServers` scalar object that names all servers in the domain. The `serverTable` object contains a `serverDeployments` scalar object, which describes all applications currently deployed on a server.

Tabular objects never directly contain object instances (MIB variables). Instead, tabular objects contain scalar objects, and the scalar objects contain variables. For example, if you created two Managed Servers in a domain named `MS1` and `MS2`, the MIB contains one `serverTable` object, which in turn contains a `serverName` object. The `serverName` object contains two variables that contain the value `MS1` and `MS2`. See Figure 1-3.

Figure 1-3 Hierarchy of Objects and Object Instances



- Configuration and Runtime Hierarchies
- Relationship of the MIB Module to the WebLogic Server MBean Data Model

Configuration and Runtime Hierarchies

Instead of one large hierarchy for all of its management data, the WebLogic Server management data model consists of two hierarchies: one for its configuration data and another for the performance and monitoring data that are available only at runtime. All managed objects that describe runtime data contain the word "runtime" in their name; configuration managed objects do not. For example, the MIB contains a `domainTable` that describes a domain's configuration and a `domainRuntimeTable` that describes runtime data.

Relationship of the MIB Module to the WebLogic Server MBean Data Model

WebLogic Server provides managed beans (MBeans) as part of its implementation of Jakarta Management Extensions (JMX). JMX is a Jakarta EE specification for programmatic access to a Web application server's management data, and an MBean is the representation of the management data and operations. JMX's purpose is the same as SNMP: provide standard communication of management information between agents and managers.

At the implementation level, the WebLogic Server SNMP agent and MIB form a protocol-specific layer on top of the WebLogic Server JMX implementation. If you are already familiar with the WebLogic Server JMX implementation, you will notice similarities in the data model for WebLogic Server MBeans and the organization of managed objects in the WebLogic Server MIB. However, there are some important differences:

- WebLogic Server enables JMX clients (similar to SNMP managers) to monitor a domain *and* to modify a domain configuration. WebLogic Server gives SNMP managers only read access to its management system.
- The data model for MBeans is a deep hierarchy, while the data model implied by the MIB is shallow. For example, a JMX client can navigate from a `DomainMBean` to its child `ServerMBeans`, and then to the children of each `ServerMBean`, and so on. The MIB, on the other hand, represents objects using unique identifiers. See Object Identifiers.
- Entries for MBean attributes that are either complex types or arrays of complex types are not included in the SNMP MIB. Only MBean attributes that are either `String` or simple types, or arrays of `String` or simple types, are defined in the MIB.

For more information about the WebLogic Server JMX implementation, see *Developing Manageable Applications Using JMX for Oracle WebLogic Server*.

Object Identifiers

A MIB assigns a unique, immutable number called an **object identifier** (OID) to each managed object that it describes. Each OID consists of a left-to-right sequence of integers. This sequence defines the location of the object in the MIB tree and specifies a unique path through the tree to the object. Each node in the path has both a number and a name associated with it.

The path `.1.3.6.1.4.1` defines the `private.enterprises` OID and each number beneath that node on the tree represents the branches in the tree reserved for a particular vendor, for example, Oracle. The MIB modules are registered at the location `.1.3.6.1.4.1.140` in the tree, and the WebLogic Server MIB module consists of all OIDs below `.1.3.6.1.4.1.140.625`.

- OIDs for Objects and Variables

OIDs for Objects and Variables

The WebLogic Server MIB module uses OIDs to reflect its hierarchical data model. For example, the OID for the `serverRuntimeTable` object is `.1.3.6.1.4.1.140.625.`**`360`**. The OID for the `serverRuntimeState` scalar object, which is contained by the `serverRuntimeTable` object is `.1.3.6.1.4.1.140.625.`**`360.1.60`**.

To identify an object instance (variable), the WebLogic SNMP agent generates and appends an additional set of numbers to the object's OID. For example, the OID for a variable of `serverRuntimeState` would be

```
.1.3.6.1.4.1.140.625.
360.1.60.32.102.100.48.98.101.102.100.99.102.52.98.97.48.48.49.102.57.53.51.50.
100.102.53.55.97.101.52.56.99.99.97.99.
```

The OID is persistent across instantiations of the object.

You can view the contents of `BEA-WEBLOGIC-MIB.asn1` to see the OIDs for managed objects, and the `SnmpWalk` or `SnmpGetNext` commands to see the OIDs for any variable. See WebLogic SNMP Command-Line Utility.

Browsing the MIB

To view the contents of the MIB module for WebLogic Server, Oracle recommends using a MIB browser. WebLogic Server does not provide a MIB browser, but most vendors of SNMP utilities do.

The MIB module is contained in the following location in your WebLogic Server installation directory:

```
WL_HOME/server/lib/BEA-WEBLOGIC-MIB.asn1
```

Monitoring Custom MBeans

You can configure a WebLogic SNMP agent to maintain a runtime MIB module that contains entries for all custom MBeans that have been registered in a WebLogic Server Runtime MBean Server. A **custom MBean** is an MBean that you create and register. You can then use SNMP managers to request information about your custom MBeans. WebLogic Server SNMP agents cannot periodically poll values of custom MBeans and generate notifications if a value crosses a threshold. Instead, SNMP managers must send requests to WebLogic Server SNMP agents. See *Developing Manageable Applications Using JMX for Oracle WebLogic Server*.

- Structure of the Custom MBean MIB Module

Structure of the Custom MBean MIB Module

For each custom MBean type, WebLogic Server adds a table to the MIB module. For each instance of a custom MBean, it adds a table row. For each MBean attribute, it adds a table column.

The table names for standard MBeans are based on the name of the MBean's implementing class; table names for model MBeans are based on the name that is supplied in the MBean's `MBeanInfo` object.

In addition to tables for custom MBean types, the MBean MIB module contains two tables, `federatedMBeanServerDelegateTable` and `mBeanServerDelegateTable`. These MBeans contain information about the MBean server itself (such as the vendor name and version number).

While WebLogic Server does not persist the MIB module as a file or other data structure, the OIDs in the module remain constant across server sessions.

Example 1-1 is an excerpt of the custom MBean MIB module from the Avitek Medical Records Application (MedRec). MedRec is an end-to-end sample Jakarta EE application that is shipped with WebLogic Server. MedRec registers two custom MBeans in the WebLogic Server Runtime MBean Server:

- `AdminReportMBean`, which polls the database every 6 seconds to check for new users to be added to the system. It stores the number of new users in an attribute named `NewUserCount`.
- `RecordSessionEJBMBean`, which contains one attribute, `TotalRx`, to record the number of times that `RecordSessionEJB` writes a prescription to the database.

The MIB module in Example 1-1 contains one table for each of MedRec's custom MBean types, `adminReportTable` and `recordSessionEJBMBeanImplTable`. Note that the table names in the MIB (`adminReportTable` and `recordSessionEJBMBeanImplTable)` match the underlying implementation classes for `AdminReportMBean` and `RecordSessionEJBMBean` (which are standard MBeans).

Example 1-1   MIB Module for Custom MBeans in medrec Server

```
CUSTOM-MBEANS-MIB DEFINITIONS ::=
BEGIN
    IMPORTS
    wls    FROM BEA-WEBLOGIC-MIB
    OBJECT-TYPE, MODULE-IDENTITY    FROM SNMPv2-SMI
    ;
customMBeansMib MODULE-IDENTITY
    LAST-UPDATED "0701101716Z"
    ORGANIZATION "..."
    CONTACT-INFO "...."
    DESCRIPTION "MIB for custom MBeans registered in WLS RuntimeMBeanServer"
    ::= { wls 50  }
customMBeansMibTables OBJECT IDENTIFIER ::= { customMBeansMib 1  }
adminReportTable-Oid OBJECT IDENTIFIER ::= {
  customMBeansMibTables 97 100 109 105 110 82 101 112 111 114
federatedMBeanServerDelegateTable-Oid OBJECT IDENTIFIER ::= {
  customMBeansMibTables 102 101 100 101 114
mBeanServerDelegateTable-Oid OBJECT IDENTIFIER ::= {
  customMBeansMibTables 109 66 101 97 110 83 101 114
recordSessionEJBMBeanImplTable-Oid OBJECT IDENTIFIER ::= {
  customMBeansMibTables 114 101 99 111 114 100
adminReportTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF AdminReportEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "Dynamically created table for type
      com.bea.medrec.admin.AdminReport"
    ::= { adminReportTable-Oid 1  }
adminReportEntry OBJECT-TYPE
    SYNTAX      AdminReportEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "Generated SNMP Table Entry."
    INDEX { adminReportIndex}
    ::= { adminReportTable 1 }
AdminReportEntry ::=
    SEQUENCE
    {
    adminReportIndex     OCTET STRING,
    adminReportObjectName OCTET STRING,
    adminReportNewUserCount OCTET STRING
    }
adminReportIndex OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "Index column"
    ::= { adminReportEntry 1  }
adminReportObjectName OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "ObjectName column"
    ::= { adminReportEntry 2  }
adminReportNewUserCount OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "Attribute exposed for management"
    ::= { adminReportEntry 3  }
... Definitions for federatedMBeanServerDelegateTable and
mBeanServerDelegateTable are omitted from this example ...
recordSessionEJBMBeanImplTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF RecordSessionEJBMBeanImplEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "Dynamically created table for type
      com.bea.medrec.controller.RecordSessionEJBMBeanImpl"
    ::= { recordSessionEJBMBeanImplTable-Oid 1  }
recordSessionEJBMBeanImplEntry OBJECT-TYPE
    SYNTAX      RecordSessionEJBMBeanImplEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "Generated SNMP Table Entry."
    INDEX { recordSessionEJBMBeanImplIndex}
    ::= { recordSessionEJBMBeanImplTable 1 }
RecordSessionEJBMBeanImplEntry ::=
    SEQUENCE
    {
    recordSessionEJBMBeanImplIndex OCTET STRING,
    recordSessionEJBMBeanImplObjectName OCTET STRING,
    recordSessionEJBMBeanImplTotalRx OCTET STRING
    }
recordSessionEJBMBeanImplIndex OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION "Index column"
    ::= { recordSessionEJBMBeanImplEntry 1  }
recordSessionEJBMBeanImplObjectName OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "ObjectName column"
    ::= { recordSessionEJBMBeanImplEntry 2  }
recordSessionEJBMBeanImplTotalRx OBJECT-TYPE
    SYNTAX      OCTET STRING
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION "Attribute exposed for management"
    ::= { recordSessionEJBMBeanImplEntry 3  }
END
```

2 Understanding WebLogic Server Notifications

You can configure a WebLogic Server SNMP agent to detect certain thresholds or conditions within a managed resource and send a report (notification) to one or more SNMP managers. WebLogic Server SNMP agents can generate notifications that conform to the SNMP v1, SNMP v2, or SNMP v3 protocols.
Learn about the notifications that WebLogic Server SNMP agents can generate:

- INFORM Notifications and TRAP Notifications
- Automatically Generated Notifications
- Log Message Notifications
- Monitor Notifications
- Attribute Change Notifications
- OIDs for WebLogic Server Notifications

INFORM Notifications and TRAP Notifications

An SNMP agent that uses the SNMPv2 or SNMPv3 protocol can send one of two types of notifications when a monitored attribute crosses a defined threshold:

- TRAP: The agent sends a TRAP notification once and assumes that the SNMP manager received the message.
- INFORM: The agent sends an INFORM notification and waits for a response from the SNMP manager that indicates the manager has received the message. If the manager does not respond, the agent sends the notification again.

Automatically Generated Notifications

WebLogic Server SNMP agents can automatically generate the notifications described in Table 2-1. Some of these notifications include name–value pairs (variable bindings) to further describe the event.

Table 2-1 Automatically Generated Notifications

Notification	Generated When	Variable Bindings
`coldStart`	The WebLogic Server instance that hosts the SNMP agent starts.	none
`wlsServerStart`	A WebLogic Server instance that was down is now up. An SNMP agent on a Managed Server generates this notification only when its host Managed Server starts. An SNMP agent on an Administration Server generates this notification when any server in the domain starts.	Contains two name–value pairs to identify server start time and the server name.

Table 2-1 (Cont.) Automatically Generated Notifications

Notification	Generated When	Variable Bindings
`wlsServerShutDown`	A server that was up is now down. An SNMP agent on a Managed Server generates this notification only when its host Managed Server stops. An SNMP agent on an Administration Server generates this notification when any server in the domain stops.	Contains two name–value pairs to identify server down time and the server name.

Log Message Notifications

Subsystems and deployable modules (such as applications) on a WebLogic Server instance generate log messages to communicate status or other operational data.

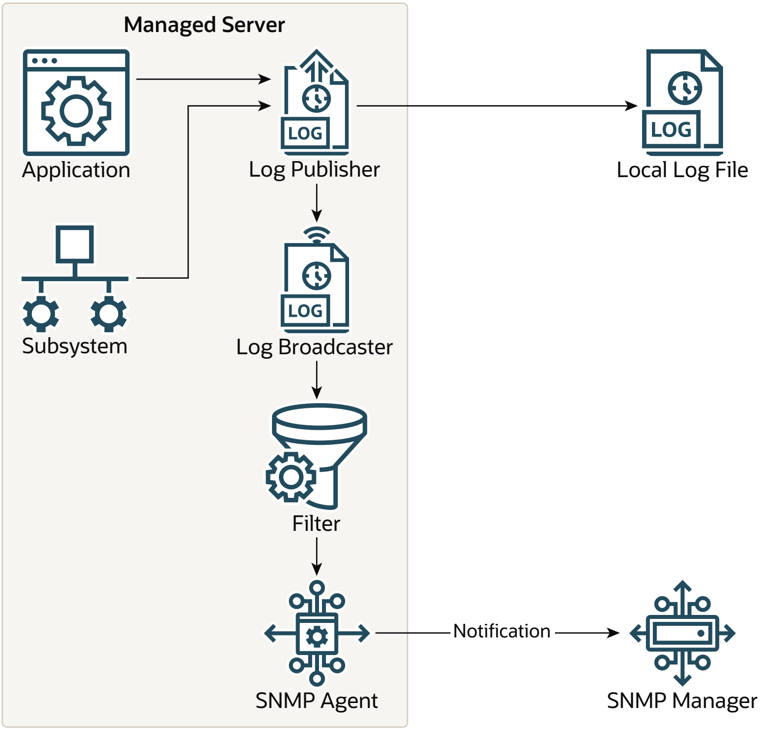
Each server instance saves these messages in a local log file and then broadcasts them as JMX notifications. You can set up a WebLogic Server SNMP agent to listen for all of these JMX notifications or you can set up a filter based on criteria such as:

- Message severity level
- Name of the subsystem that generated the message
- User ID under which the subsystem is running
- Unique message ID
- String within the message text

For example, you can specify that only messages from the Security Service of severity level `ERROR` or higher are sent to an SNMP agent.

When an agent receives a message, it generates an SNMP log notification. See Figure 2-1.

Figure 2-1 Log Message Notifications



- [Variable Bindings in Log Message Notifications](#variable-bindings-in-log-message-notifications)

Variable Bindings in Log Message Notifications

This section describes the name–value pairs that log message notifications pass to SNMP managers in the variable bindings field:

- `trapTime`: Time when the notification is generated.
- `trapServerName`: Name of the server instance on which the log message was generated.
- `trapMachineName`: Name of the machine on which the server instance is running.
- `trapLogThreadId`: Thread ID from the log message.
- `trapLogTransactionId`: Transaction ID, if any, from the log message. Transaction ID is present only for messages logged within the context of a transaction.
- `trapLogUserId`: User ID from the log message. The user ID indicates the security context in which the log message was generated.
- `trapLogSubsystem`: Subsystem that generated the log message.
- `trapLogMsgId`: Log message ID from the log message.
- `trapLogSeverity`: Message severity level from the log message.
- `trapLogMessage`: Text of the log message.

For more information about log messages and the WebLogic Server logging subsystem, see Understanding WebLogic Logging Services in *Configuring Log Files and Filtering Log Messages for Oracle WebLogic Server*.

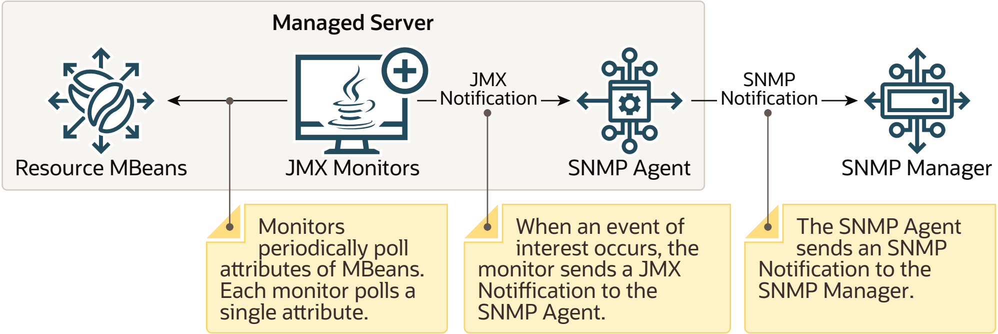
Monitor Notifications

To periodically check the value of WebLogic resources for changes, you set up JMX monitors and configure an SNMP agent to listen for notifications from these monitors.

JMX is a Jakarta EE specification for exposing management data. It is the foundation for the WebLogic Server management system. In the JMX specification, management data and operations are made public through managed beans (MBeans). The managed objects in the WebLogic Server MIB correspond to MBeans and MBean attributes. See Relationship of the MIB Module to the WebLogic Server MBean Data Model.

JMX monitors poll WebLogic Server MBeans at a specified interval and send notifications to an WebLogic SNMP agent when an event that you specify occurs, such as the crossing of a threshold. The SNMP agent generates a notification and sends it to the SNMP managers. See Figure 2-2.

Figure 2-2 Monitor Notifications


Managed Server
Resource MBeans
JMX Monitors
JMX Notification
SNMP Agent
SNMP Notification
SNMP Manager
Monitors periodically poll attributes of MBeans. Each monitor polls a single attribute.
When an event of interest occurs, the monitor sends a JMX Notiffication to the SNMP Agent.
The SNMP Agent sends an SNMP Notification to the SNMP Manager.

You can configure three types of JMX monitors depending on the data type of the attribute that you want to observe (the *MBean Reference for Oracle WebLogic Server* describes the type of data that an attribute returns):

- **Counter Monitor**

  A counter monitor observes MBean attribute values that are returned as an `Integer` object type.

  You can specify that a notification is generated if an attribute is beyond the bounds of a threshold value. You can also specify that if a value exceeds a threshold, the monitor increases the threshold by an offset value. Each time the observed attribute exceeds the new threshold, the threshold is increased by the offset value, up to a maximum allowable threshold that you specify.

- **Gauge Monitor**

  A gauge monitor observes changes in MBean attributes that are expressed as integers or floating-point.

  You can specify that a notification is generated if an attribute is beyond the bounds of a high or low threshold value.

- **String Monitor**

  A string monitor observes changes in attributes that are expressed as `String` objects.

You can specify that a notification is generated if there is a match between the value and the string you provide, or you can specify that the notification is generated if the value differs from the string you provide.

- Variable Bindings in Monitor Notifications

Variable Bindings in Monitor Notifications

A JMX monitor polls for a specified threshold or condition and the agent generates a monitor notification when the specified threshold is crossed or the specified condition occurs. A WebLogic Server SNMP agent includes the following name–value pairs in the variable bindings of each monitor notification:

- `trapTime`: Time at which the notification was generated.
- `trapServerName`: Local server whose attribute value generated the notification.
- `trapMonitorType`: Either `CounterMonitor`, `StringMonitor`, or `GaugeMonitor`.
- `trapMonitorThreshold`: ASCII representation of the threshold that triggered the notification.
- `trapMonitorValue`: ASCII representation of the value that triggered the notification.
- `trapMBeanName`: Name of the MBean that contained the attribute being monitored.
- `trapMBeanType`: Type of the MBean that contained the attribute being monitored.
- `trapAttributeName`: Name of the attribute whose value triggered the notification.
- `trapConfigName`: Name of the SNMP MBean that defines the monitoring rule or threshold that caused the TRAP notification to be generated.

Attribute Change Notifications

While you can use JMX monitors to periodically poll WebLogic Server resources for changes to attributes that exceed the bounds of specific thresholds, you can also configure an SNMP agent to send a notification immediately after an attribute is changed in any way. For example, you can use a JMX monitor to poll for changes in the current number of active JDBC connections. If the number of active connections exceeds a threshold, the SNMP agent can send a notification.

> **Note:**
>
> Creation of attribute changes for runtime MBeans is not supported. Only attributes of configuration MBeans support attribute change notifications.

- Variable Bindings in Attribute Change Notifications

Variable Bindings in Attribute Change Notifications

An attribute change notification includes the following name–value pairs in the variable bindings:

- `trapTime`: The time at which the notification was generated.
- `trapServerName`: The name of the Administration Server.

- `trapMBeanName`: Name of the MBean that includes the attribute.
- `trapMBeanType`: Type of the MBean that includes the attribute.
- `trapAttributeName`: Name of the configuration attribute that has changed.
- `trapAttributeChangeType`: The value can be either `ADD`, `REMOVE`, or `UPDATE`.
- `trapAttriruteOldVal`: Value of the attribute before the change.
- `trapAttributeNewVal`: Value of the attribute after the change.
- `trapConfigName`: Name of the SNMP MBean that defines the monitoring rule or threshold that caused the TRAP notification to be generated.

OIDs for WebLogic Server Notifications

The object identifier (OID) for all WebLogic Server notifications starts with the following WebLogic Server OID:

```
.1.3.6.1.4.1.140.625
```

Table 2-2 describes the subsequent values in OIDs for WebLogic Server notifications.

Table 2-2 OIDs for WebLogic Server Notifications

Value	Generated When
60	A server instance logs a message that matches user-defined criteria for sending a log notification. For example, `.1.3.6.1.4.1.140.625.60`
65	A WebLogic Server instance that was down is now up. An SNMP agent on a Managed Server generates this notification only when its host Managed Server starts. An SNMP agent on an Administration Server generates this notification when any server in the domain starts. This is called a `wlsServerStart` notification. For example, `.1.3.6.1.4.1.140.625.65`
70	A server that was up is now down. An SNMP agent on a Managed Server generates this notification only when its host Managed Server stops. An SNMP agent on an Administration Server generates this notification when any server in the domain stops. This is called a `wlsServerShutDown` notification. For example, `.1.3.6.1.4.1.140.625.70`
75	A user-defined JMX monitor detects the crossing of a threshold or occurrence of an event. For example, `.1.3.6.1.4.1.140.625.75`
80	An attribute selected by the user has changed in value. For example, `.1.3.6.1.4.1.140.625.80`

Some notifications also include variable bindings. To see the OIDs for the variable bindings, see MIB Module for WebLogic Server.

3

Understanding SNMP Proxies

This chapter describes background information about WebLogic Server and SNMP proxy agents.
This chapter includes the following sections:

- SNMP Agent as Proxy for Other Agents
- The Microsoft Windows SNMP Service

SNMP Agent as Proxy for Other Agents

WebLogic SNMP agents can function as primary agents that forward (proxy) requests to other SNMP agents. To use the primary agent functionality of a WebLogic Server agent, you assign branches of the registration tree (OID tree) as the responsibility of other SNMP agents. When an SNMP manager sends a request to a WebLogic SNMP agent, if the OID of the requested object is under the branch of the OID tree assigned to a proxied agent, then the WebLogic SNMP agent forwards the request to the proxied agent.

> **Note:**
>
> You cannot use a WebLogic Server SNMP agent as a proxy for SNMP agents in other WebLogic Server domains. For example, a WebLogic Server agent in domainA cannot proxy requests to a WebLogic Server agent in domainB. This limitation is in effect because all WebLogic Server agents use the same MIB root.
>
> Instead of proxying requests to multiple WebLogic Server domains, you can place all of your server instances in a single domain and send requests directly to each Managed Server. See Narrowing the Scope of a Request.

- Configuring the SNMP Protocols for Proxied Communication
- Specifying Credentials for Proxied Communication
- Choosing Listen Ports for Proxied Agents

Configuring the SNMP Protocols for Proxied Communication

When a WebLogic SNMP agent proxies requests to another agent, the communication process consists of separate events (conversations), each of which can use different SNMP protocols (see Figure 3-1):

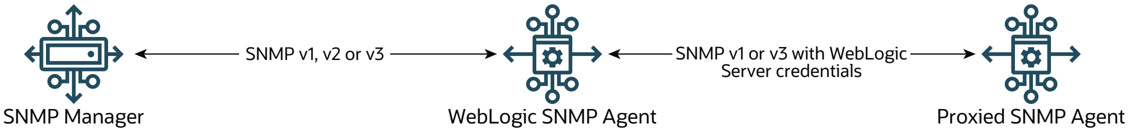
- Communication between the SNMP manager and the WebLogic SNMP agent.

  WebLogic SNMP agents can receive requests that use the SNMP v1, v2, or v3 protocols.
- Communication between the WebLogic SNMP agent and the proxied agent.

  Regardless of the protocol used between a manager and a WebLogic SNMP agent, you can specify whether a WebLogic SNMP agent forwards SNMP v1 or v3 requests to proxied agents.

If you specify a security name when you configure a proxy, then the WebLogic SNMP agent will always use SNMP v3 to communicate with the proxied agent. If you do not specify a security name, the WebLogic SNMP agent will always use SNMP v1.

Figure 3-1 Separate Conversations for Proxied SNMP Communication



Specifying Credentials for Proxied Communication

When the WebLogic SNMP agent receives a request from a manager, it always authenticates and performs other security operations on the request before determining if it should process the request itself or forward it to a proxied agent. Therefore, regardless of the protocol or the credentials that are used for communication between the SNMP manager and the WebLogic SNMP agent, the WebLogic SNMP agent supplies its own credentials for its conversation with the proxied agent.

Choosing Listen Ports for Proxied Agents

By default a WebLogic SNMP agent listens for management requests on port 161. If a WebLogic SNMP agent is to proxy for other SNMP agents, then those other agents must be configured to listen for SNMP management requests on some other port.

The Microsoft Windows SNMP Service

While a WebLogic Server SNMP agent can be a proxy for other SNMP agents, it cannot be configured as a subagent of the Microsoft Windows SNMP agent service.

Using Microsoft Extension Agent API, the Microsoft Windows SNMP agent service can be a proxy for other SNMP agents. However, WebLogic Server does not support this feature and cannot use the Windows SNMP agent as a proxy.

4
WebLogic SNMP Command-Line Utility

This chapter describes the SNMP command-line utility, which offers many of the same features as an SNMP manager. You can use this utility to test and troubleshoot the configuration of your SNMP agents in a WebLogic Server domain.
This chapter includes the following sections:

- Required Environment for the SNMP Command Line Utility
- Syntax and Commands for the SNMP Command-Line Utility
- Examples

Required Environment for the SNMP Command Line Utility

To set up your environment for the WebLogic Server SNMP command line utility:

1. Install and configure the WebLogic Server software, as described in the Installation Guide.
2. Create an SNMP agent and trap destination in a WebLogic Server domain.
3. Open a command prompt (shell) and invoke the following script:

   `WL_HOME\server\bin\setWLSEnv.sh` (or `setWLSEnv.cmd` on Windows)

   where *`WL_HOME`* is the directory in which you installed WebLogic Server.

   The script adds a supported JDK to the shell's `PATH` environment variable and adds WebLogic Server classes to the `CLASSPATH` variable.

Syntax and Commands for the SNMP Command-Line Utility

Invoke the SNMP command-line utility using the following syntax:

```
java weblogic.diagnostics.snmp.cmdline.Manager command [-?]
```

In the preceding syntax, *`command`* represents one of the commands described in Table 4-1. Use the optional argument, `-?`, to obtain usage information about the specified command.

Table 4-1 SNMP Commands

Command	Description
`SnmpBulkWalk [`*`startingOID`*`] [`*`endingOID`*`]`	Returns a collection of MIB object **instances** (MIB variables) by repeatedly invoking `SnmpGetNext` in a pattern that you specify. Starting from the first OID that you specify on the command line, the command invokes the `SnmpGetNext` command as many times as needed to retrieve all MIB variables below the object. The command increments the OID that you specified on the command line and repeats the pattern described in the previous paragraph. Optionally specify the `-Bm` argument, which groups multiple `SnmpGetNext` invocations in each request that it sends to the SNMP agent. For example, `-Bm 3` causes this command to issue 3 `SnmpGetNext` invocations in each request until it reaches the last OID in the sequence.
`SnmpInform`	Constructs an INFORM notification and distributes it to an SNMP manager or trap monitor.
`SnmpGet`	Retrieves the value of one or more MIB variables. This command does not accept OIDs for managed objects. You can specify an optional interval at which this command repeatedly retrieves the value of the specified variable.
`SnmpGetAll <`*`columnOID`*`> [<`*`columnOID`*`>...]`	Walks the table columns that you specify and groups the column values for each row in the output. All specified columns must be from the same table.
`SnmpGetBulk [`*`OIDs`*`]`	Returns a collection of MIB variables by repeatedly invoking `SnmpGetNext` in a pattern that you specify. Optionally include the `-Bn` and `-Bm` arguments and one or more OIDs to specify the pattern. The `-Bn` (non-repeaters) argument specifies the number of OID arguments on the command line for which the command will **not** repeatedly perform `SnmpGetNext` operations. For these OIDs, the command performs an `SnmpGetNext` operation once and then moves on to the next OID argument on the command line. Consider using `-Bn` and non-repeater OIDs for scalar objects. The `-Bm` (maximum repetitions) argument specifies how many times the command will perform `SnmpGetNext` operations for all other OIDs on the command line. Consider using `-Bm` and associated OIDs for tabular objects.
`SnmpGetNext`	Returns managed objects or MIB variables. If you specify a tabular object, this command returns the first child managed object. If you specify a scalar object, this command returns the first object variable. Instead of the recursive listing that the `SnmpWalk` command provides, this command returns the description of only one managed object or variable whose OID is the next in sequence. You could string together a series of `SnmpGetNext` commands to achieve the same result as the `SnmpWalk` command.
`SnmpTrap`	Constructs a TRAP notification and distributes it to an SNMP manager or trap monitor.
`SnmpTrapLogger`	Starts a process that listens for notifications. Writes each notification that it receives to a log file.

Table 4-1 (Cont.) SNMP Commands

Command	Description
`SnmpTrapMonitor`	Starts a process that listens for notifications. Prints each notification that it receives to standard out.
`SnmpWalk`	Returns all managed objects or variables that are below a specified node in the MIB or within a specified range. If you specify the OID for a tabular object, the command returns all of its object variables along with all related (child) objects and variables.

Examples

The examples in this section assume that you have created an SNMP agent on the Administration Server and that you have not modified the default values for the agent. (For example, the agent listens on UDP port 161 and uses `public` as its community name.)

These examples use the following options for loading the MIB module, which enable you to refer to managed objects by their display names instead of OIDs:

- `-m` — Specifies the WebLogic Server MIB module, `BEA-WEBLOGIC-MIB`
- `-M` — Specifies either the directory to, or the resource classpath of, the compiled MIB. In these examples, the resource classpath is used.

To see the display names for all WebLogic Server managed objects, see MIB Module for WebLogic Server.

- The following example displays usage information for the `SnmpWalk` command:

  ```
  java weblogic.diagnostics.snmp.cmdline.Manager SnmpWalk -?
  ```

- The following example retrieves the names of all applications that are deployed in a domain.

  ```
  java weblogic.diagnostics.snmp.cmdline.Manager SnmpWalk
  -m BEA-WEBLOGIC-MIB -M /weblogic/diagnostics/snmp/mib
  applicationRuntimeObjectName
  ```

- The following command retrieves all variable bindings for all applications in the domain. To make the network communication more efficient, it groups 3 `SnmpGetBulk` commands in each request:

  ```
  java weblogic.diagnostics.snmp.cmdline.Manager SnmpBulkWalk -v2 -Bm 3
  -m BEA-WEBLOGIC-MIB -M /weblogic/diagnostics/snmp/mib
  applicationRuntimeTable
  ```

- The following command retrieves the name of the first application that is deployed in the domain. Then it retrieves the name and Bytes Pending Count for the first two JMS servers in the domain:

  ```
  java weblogic.diagnostics.snmp.cmdline.Manager SnmpGetBulk -v2
  -Bn 1 -Bm 2 applicationRuntimeObjectName
  jmsServerRuntimeObjectName jmsServerRuntimeBytesPendingCount
  ```