Oracle® Fusion Middleware

Administering Oracle WebLogic Server with RESTful Management Services

12*c* (12.1.2)

E28120-02

April 2015

This document describes how SasS application developers can use RESTful APIs provided by WebLogic Server to monitor their applications, and WebLogic Server health status, when they are running behind firewalls in Oracle Cloud.

ORACLE

Oracle Fusion Middleware Administering Oracle WebLogic Server with RESTful Management Services, 12*c* (12.1.2)

E28120-02

Contents

Preface

This preface describes the document accessibility features and conventions used in this guide—*Administering Oracle WebLogic Server with RESTful Management Services*.

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.

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 Introduction and Roadmap

This chapter describes the contents and organization of this guide—*Administering Oracle WebLogic Server with RESTful Management Services*.

This chapter includes the following sections:

- Document Scope and Audience
- Guide to This Document
- Related Documentation
- New and Changed Features in this Release

> **Note:** In the current release of WebLogic Server, RESTful Management Services support only the `GET HTTP` method because they are strictly limited to WebLogic Server monitoring.

1.1 Document Scope and Audience

WebLogic Server provides RESTful Management Services that enable you to monitor WebLogic Server outside of the Oracle Public Cloud Management Console. This document describes how to use the RESTful Management Services.

It is assumed that the reader is familiar with Java EE and general application management concepts. This document emphasizes a hands-on approach to developing a limited but useful set of RESTful monitoring services.

1.2 Guide to This Document

The document is organized as follows:

- Chapter 1, "Introduction and Roadmap," provides a roadmap for this guide and describes the audience for this guide.
- Chapter 2, "Using RESTful Management Services,"provides an overview of using the RESTful Management Service with WebLogic Server. This chapter also describes how to enable the RESTful Management Service in WebLogic Server, the formatting and usage of HTTP URLs, and the supported REST resources when monitoring WebLogic Server with RESTful Management Services.

1.3 Related Documentation

For guidelines on developing other types of management services for WebLogic Server applications, see the following documents:

- *Adding WebLogic Logging Services to Applications Deployed on Oracle WebLogic Server* describes WebLogic support for internationalization and localization of log messages, and shows you how to use the templates and tools provided with WebLogic Server to create or edit message catalogs that are locale-specific.
- *Configuring and Using the Diagnostics Framework for Oracle WebLogic Server* describes how to configure and use the WebLogic Diagnostic Framework to create, collect, analyze, archive, and access diagnostic data generated by a running server and the applications deployed within its containers.

1.4 New and Changed Features in this Release

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

2

Using RESTful Management Services

This chapter describes how to use the RESTful Management Services with WebLogic Server.

This chapter includes the following sections:

- Section 2.1, "Overview"
- Section 2.2, "Getting Started"
- Section 2.3, "Using REST URLs to Monitor WebLogic Server Domains"
- Section 2.4, "Supported REST Resources for Monitoring WebLogic Server"

> **Note:** In the current release of WebLogic Server, RESTful Management Services support only the `GET HTTP` method because they are strictly limited to WebLogic Server monitoring.

2.1 Overview

WebLogic Server RESTful Management Services provides representational state transfer (REST) resources that enable you to monitor WebLogic Server outside of the Oracle Public Cloud Management Console. REST clients can embed fine-grained monitoring information in flexible, customized consoles that can monitor a domain's servers, clusters, applications, and data sources while they are running behind firewalls in the Oracle Public PaaS WebLogic Service. The monitoring resources are hosted by an internal Web application.

Only users that belong to the `Administrators` group or the `Monitors` group are allowed access to the RESTful Management Services. For more information about defining user roles in WebLogic Server, see "Users, Groups, And Security Roles" in *Securing Resources Using Roles and Policies for Oracle WebLogic Server*.

2.1.1 What Can You Monitor With RESTful Management Services?

The following components of WebLogic Server can be monitored using the RESTful Management Services:

- Server – All server instances in a domain or a specified server instance.
- Clusters – All clusters in a domain or a specified cluster, including all server members of the cluster.
- Applications – All applications deployed in a domain or a specified application.
- Data sources – All data sources running in a domain or a specified data source.

2.1.2 What Clients Can Access RESTful Management Services?

You can access the WebLogic Server RESTful Management Services through client applications, such as:

- Web browsers
- cURL
- GNU Wget

You can also use the WebLogic Server RESTful Management Services in clients that are developed in various programming languages, such as:

- JavaScript
- Ruby
- Perl
- Java
- JavaFX

2.2 Getting Started

In order to use RESTful Management Services in a WebLogic Server domain, your WebLogic Administrator must enable them using the WebLogic Server Administration Console. For more information, see "Enable RESTful Management Services" in the *Administration Console Online Help*.

2.3 Using REST URLs to Monitor WebLogic Server Domains

In the current release of WebLogic Server, RESTful Management Services support only the `GET HTTP` method because they are strictly limited to WebLogic Server monitoring. Each monitored WebLogic Server resource is accessible through an HTTP uniform resource locator (URL).

2.3.1 REST URL Format to Monitor WebLogic

The format of the WebLogic Server monitoring resource URL is:

`http(s)://`*`host:port`*`/management/tenant-monitoring/`*`path`*

where:

`host` – the host where WebLogic Server is running

`port` – the HTTP or HTTPS port

`path` – the relative path that identifies an individual resource. For example, the path to a server instance would be: `servers/`*`myserver`*

2.3.2 Supported Content Types in Requests to REST Resources

The WebLogic Server RESTful Management Services support the following representation formats:

- JSON
- XML
- HTML

Clients should specify the resource representation through the HTTP header. For example, if clients is using the cURL utility, then it should specify the type through the `-H` option, as follows:

- JSON – specify `-H "Accept application/json"`
- XML – specify `-H "Accept application/xml"`
- HTML – omit the `-H` option

2.3.3 How WebLogic Server Monitoring Resources Are Represented

There are two types of WebLogic Server monitoring resource representations:

- Section 2.3.3.1, "Item Resource Representation"
- Section 2.3.3.2, "Collection Resource Representation"

2.3.3.1 Item Resource Representation

An *item resource* represents an instance of a WebLogic Server entity; for example, a server instance.

2.3.3.1.1 JSON Format Here is an example output of an item resource representation for a server instance using JSON:

```
{
  "body": {
    "item": {
      // attributes for the item, e.g.
      // "name": "adminserver"
      // "state": "RUNNING",
      // …
    }
  },
  "messages": [
    // an array of messages
  ]
}
```

The possible attribute value types are:

- string, enclosed by double quotes ""
- boolean, true or false
- number
- null
- array, enclosed by "[" and "]"
- object, enclosed by "{" and "}"

2.3.3.1.2 XML Format Here is an example output of an item resource representation for a server instance using XML:

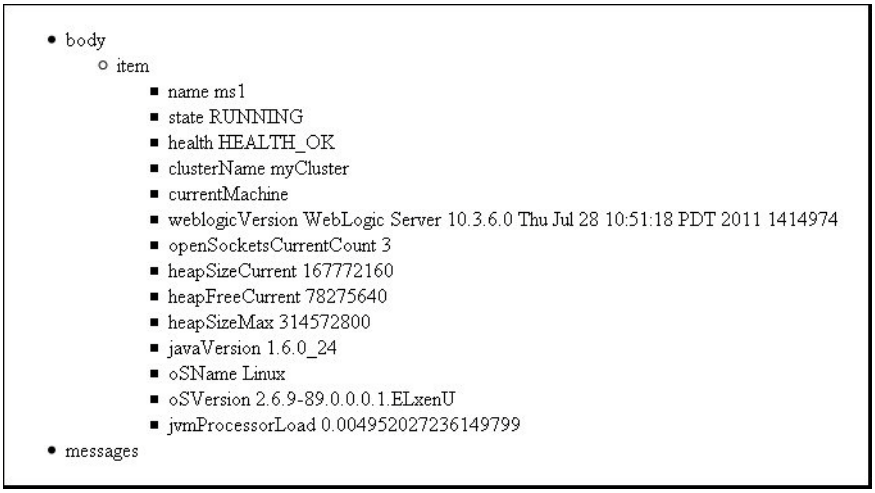
```
<?xml version="1.0" encoding="utf-8"?>
<data>
 <object>
  <property name="body">
   <object>
    <property name="item">
     <object>
```

```
      <!--
      <property name="Name">
       <value type="string">adminserver</value>
      </property>
       // other properties
       -->
     </object>
    </property>
   </object>
  </property>
  <property name="messages">
   <array>
    <!-- message objects -->
   </array>
   </property>
</object>
</data>
```

2.3.3.1.3 HTML Format The HTML resource representation for an item resource is a Web page that displays the item attributes and their values in HTML list format. It should only be used for testing and debugging purposes.

Here is an example output of an item resource representation for a server instance using HTML:

Figure 2–1 Item Resource in HTML Format



2.3.3.2 Collection Resource Representation

A *collection resource* represents a set of WebLogic Server entities of the same type; for example, when representing all servers in a domain. Usually, an item resource exists for each entity inside the collection.

2.3.3.2.1 JSON Format Here is an example output of a collection resource representation for all servers in a domain using JSON:

```
{
 "body": {
  "items": [
    {
      // attributes for item 1
```

```
      // "name": "adminserver"
      // "state": "RUNNING",
      // …
     },
    {
      // attributes for item 2
     },
     …
    {
      // attributes for item n
     }
    ],
   "messages": [
    ]
}
```

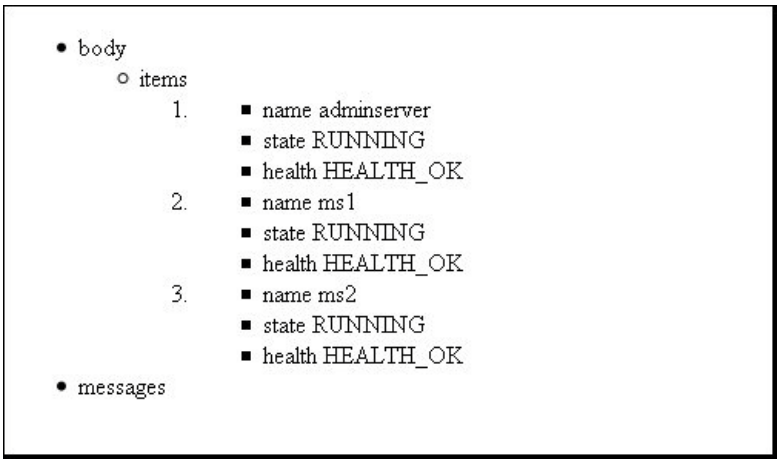
2.3.3.2.2 XML Format Here is an example output of a collection resource representation for all servers in a domain using XML:

```
<?xml version="1.0" encoding="utf-8"?>
<data>
  <object>
    <property name="body">
      <object>
        <property name="items">
          <array>
            <!--
            <object>
              <property name="name">
                <value type="string">adminserver</value>
             </property>
              // other properties
            </object>

             // other items
             -->
          </array>
        </property>
      </object>
    </property>
    <property name="messages">
      <array>
        <!--
        <object>
          <property name="severity">
            <value type="string">WARNING</value>
         </property>
          <property name="message">
            <value type="string">Server ms1 is not running.</value>
         </property>
       </object>

       // other messages
       -->
      </array>
    </property>
  </object>
</data>
```

2.3.3.2.3 HTML Format The HTML resource representation for a collection resource is a web page that displays the attributes and values for all collection items in an HTML list format. It should only be used for testing and debugging purposes.

Here is an example output of an collection resource representation for all servers in a domain using HTML:

Figure 2–2 Collective Resource in HTML Format



2.3.3.3 Compact versus Full Response Format

The collection resource uses a compact response format by default. That is, only a subset of all available monitoring information is returned for each item in the collection.

To get the full response format for each item, you need to add `format=full` in the request parameter. For example:

`http://host:port/management/tenant-monitoring/servers?format=full`

The item resource always uses a full response format that includes all monitoring information available through the REST interfaces. The body item in an Item resource contains exactly the same information as the corresponding item in a collection resource in full format.

2.4 Supported REST Resources for Monitoring WebLogic Server

This section describes the supported REST resources when monitoring WebLogic Server domains with RESTful Management Services.

- Section 2.4.1, "REST Resources for Monitoring All Servers In a Domain,"
- Section 2.4.2, "REST Resources for Monitoring a Specified Server In a Domain,"
- Section 2.4.3, "REST Resources for Monitoring All Clusters In a Domain,"
- Section 2.4.4, "REST Resources for Monitoring a Specified Cluster In a Domain,"
- Section 2.4.5, "REST Resources for Monitoring All Applications Deployed In a Domain,"
- Section 2.4.6, "REST Resources for Monitoring a Specified Application Deployed In a Domain,"
- Section 2.4.7, "REST Resources for Monitoring All Data Sources In a Domain,"

- Section 2.4.8, "REST Resources for Monitoring a Specified Data Source In a Domain,"

2.4.1 REST Resources for Monitoring All Servers In a Domain

The `/servers` URL listed below returns all servers configured in a domain and provides run-time information for each server, including the server state and health.

`http(s)://host:port/management/tenant-monitoring/servers`

Table 2–1 REST Resources for Monitoring All Servers In a Domain

Item Attributes (compact format)	Type	Valid Values
`name`	string	Name of the server.
`state`	string	Server status. Possible states are: ■ RUNNING ■ STARTING ■ STANDBY ■ ADMIN ■ RESUMING ■ SHUTDOWN
`health`	string	Server health state. Possible health states are: ■ HEALTH_OK ■ HEALTH_WARN ■ HEALTH_CRITICAL ■ HEALTH_FAILED ■ HEALTH_OVERLOADED ■ UNKNOWN

Here is an example of a `/servers` URL response in JSON format:

```
{
  "body": {
    "items": [
      {
        "name": "adminserver",
        "state": "RUNNING",
        "health": " HEALTH_OK "
      },
      {
        "name": "ms1",
        "state": "SHUTDOWN",
        "health": ""
      }
     ],
   },
   "messages": [
  ]
 }
```

2.4.2 REST Resources for Monitoring a Specified Server In a Domain

The `/servers/{servername}` URL listed below returns information for a specified server in a domain, including the server state, health, and JVM heap availability.

`http(s)://host:port/management/tenant-monitoring/servers/{servername}`

Table 2–2 REST Resources for Monitoring a Specified Server In a Domain

Item Attributes	Type	Valid Values
`name`	string	Name of the server instance.
`state`	string	Server status. Possible states are: ■ RUNNING ■ STARTING ■ STANDBY ■ ADMIN ■ RESUMING ■ SHUTDOWN
`health`	string	Server health state. Possible health states are: ■ HEALTH_OK ■ HEALTH_WARN ■ HEALTH_CRITICAL ■ HEALTH_FAILED ■ HEALTH_OVERLOADED ■ UNKNOWN
`clusterName`	string	Cluster to which the server belongs.
`currentMachine`	string	Machine on which the server is running.
`weblogicVersion`	string	Version of the WebLogic Server instance.
`openSocketsCurrentCount`	number	Current number of sockets registered for socket muxing on the server.
`heapSizeCurrent`	number	Current size (in bytes) of the JVM heap.
`heapFreeCurrent`	number	Current amount of memory (in bytes) that is available in the JVM heap.
`javaVersion`	string	Java version of the JVM.
`osName`	string	Operating system on which the JVM is running.
`osVersion`	string	Version of the operating system on which the JVM is running.
`jvmProcessorLoad`	number	Average load that the VM is placing on all processors. For example, 1.0 represents 100%.

Here is an example of a `/servers/{servername}` URL response in JSON format:

```
{
  "body": {
    "item": {
      "name": "adminserver",
      "clusterName": null,

      "state": "RUNNING",
```

```
    "currentMachine": "machine-0",
    "weblogicVersion": "WebLogic Server 12.1.1.0.0 Thu May 5 01:17:16 2011 PDT",
    "openSocketsCurrentCount": 2,
    "health": "HEALTH_OK",

    "heapSizeCurrent": 536870912,
    "heapFreeCurrent": 39651944,
    "heapSizeMax": 1073741824,
    "javaVersion": "1.6.0_20",
    "osName": "Linux",
    "osVersion": "2.6.18-238.0.0.0.1.el5xen",

    "jvmProcessorLoad": 0.25,
   }
  },
   messages: [
  ]
}
```

2.4.3 REST Resources for Monitoring All Clusters In a Domain

The `/clusters` URL listed below returns all clusters configured in a domain and provides run-time information for each cluster and for each cluster's member servers, including all the member servers' state and health.

`http(s)://`*`host:port`*`/management/tenant-monitoring/clusters`

Table 2–3 REST Resources for Monitoring All Clusters In a Domain

Item Attributes (compact format)	Type	Valid Values
`name`	string	Name of the cluster.
`servers`	string	Member servers of a cluster: ■ `name` – the member server name. ■ `state` – member server status. Possible states are: ■ RUNNING ■ STARTING ■ STANDBY ■ ADMIN ■ RESUMING ■ SHUTDOWN ■ `health` – member server health state. Possible states are: ■ HEALTH_OK ■ HEALTH_WARN ■ HEALTH_CRITICAL ■ HEALTH_FAILED ■ HEALTH_ OVERLOADED ■ UNKNOWN

Here is an example of a `/clusters` URL response in JSON format:

```
{
  "body": {
    "items": [
      {
        "name": "mycluster1",
        "servers": [
          {
            "name": "ms1",
            "state": "RUNNING",
            "health": "HEALTH_OK"
          },
          {
            "name": "ms2",
            "state": "RUNNING",
            "health": "HEALTH_OVERLOADED"
          }
        ]
      }
    ],
  },
   "messages": [
  ]
 }
```

2.4.4 REST Resources for Monitoring a Specified Cluster In a Domain

The `/clusters/{clustername}` URL listed below returns run-time information for the specified cluster and its member servers, including the member servers' state and health.

`http(s)://`*`host:port`*`/management/tenant-monitoring/clusters/{clustername}`

Table 2–4 REST Resources for Monitoring a Specified Cluster In a Domain

Item Attributes	Type	Valid Values
`name`	string	Name of the cluster.
`servers`	object or number	Member servers of a cluster: ■ `name` – Name of the member server instance. ■ `state`: member server status: ■ STARTING ■ RUNNING ■ STANDBY ■ ADMIN ■ RESUMING ■ SHUTDOWN ■ `health`: member server health state: ■ HEALTH_OK ■ HEALTH_WARN ■ HEALTH_CRITICAL ■ HEALTH_FAILED ■ HEALTH_ OVERLOADED ■ UNKNOWN ■ `clusterMaster` – (object) True if the server is the cluster master of a cluster which is configured for server migration. ■ `dropOutFrequency` – (number) A relative measure of the number of times that a server has become unreachable from the cluster perspective. ■ `resendRequestsCount` – (number) Number of state-delta messages that had to be re-sent because a receiving server in the cluster missed a message. ■ `fragmentsSentCount` – (number) Total number of message fragments sent from the server into the cluster. This is applicable to both multicast and unicast message types. ■ `fragmentsReceivedCount` – (number) Total number of messages received on the server from the cluster. This is applicable to both multicast and unicast message types.

Here is an example of a `/cluster` URL response in JSON format:

```
{
  "body": {
    "item": {
      "name": "mycluster1",
      "servers": [
        {
          "name": "ms1",
          "state": "RUNNING",
```

```
          "health": "OK",

          "clusterMaster": false,

          "dropOutFrequency": "Never"
          "resendRequestsCount": 0,
          "fragmentsSentCount": 3708,
         "fragmentsReceivedCount": 3631
        },
        {
          "name": "ms2",
          "state": "RUNNING",
          "health": "OK"
          ...
        }
       ]
      }
    },
     "messages": [
    ]
  }
```

2.4.5 REST Resources for Monitoring All Applications Deployed In a Domain

The `/applications` URL listed below returns all applications deployed in the domain and their run-time information, including the application type and their state and health.

`http(s)://`*`host:port`*`/management/tenant-monitoring/applications`

Table 2–5 REST Resources for Monitoring All Applications Deployed In a Domain

Item Attributes (compact format)	Type	Valid Values
`name`	string	Name of the application.

Table 2–5 (Cont.) REST Resources for Monitoring All Applications Deployed In a Domain

Item Attributes (compact format)	Type	Valid Values
`type`	string	Application type. Possible types are: ■ ear ■ war ■ ejb ■ rar ■ jms ■ jdbc
`state`	string	Consolidated application running status. Possible states are: ■ STATE_NEW ■ STATE_FAILED ■ STATE_RETIRED ■ STATE_PREPARED ■ STATE_ADMIN ■ STATE_ACTIVE ■ STATE_UPDATE_PENDING
`health`	string	Consolidated health state of the application. Possible health states are: ■ HEALTH_OK ■ HEALTH_WARN ■ HEALTH_CRITICAL ■ HEALTH_FAILED ■ HEALTH_OVERLOADED ■ UNKNOWN

Here is an example of an `/applications` URL response in JSON format:

```
{
  "body": {
    "items": [
      {
        "name": "appscopedejbs",
        "type": "ear",
         "state": "STATE_ACTIVE",
         "health": " HEALTH_OK"
       },
      {
        "name": "MyWebApp",
        "type": "war",
         "state": "STATE_NEW"
      }
     }
    ],
   },
    "messages": [
   ]
 }
```

2.4.6 REST Resources for Monitoring a Specified Application Deployed In a Domain

The `/applications/{applicationname}` URL returns the run-time information of a specified application, including statistics for entity beans, application-scoped work managers, and data sources.

`http(s)://host:port/management/tenant-monitoring/applications/{application name}`

Table 2–6 REST Resources for Monitoring a Specified Application Deployed In a Domain

Item Attributes	Type	Valid Values
`name`	string	Name of the application.
`type`	string	Application type. Possible types are: ■ ear ■ war ■ ejb ■ rar ■ jms ■ jdbc
`state`	string	Consolidated application running status. Possible states are: ■ STATE_NEW ■ STATE_FAILED ■ STATE_RETIRED ■ STATE_PREPARED ■ STATE_ADMIN ■ STATE_ACTIVE ■ STATE_UPDATE_PENDING
`health`	string	Consolidated health state of the application. Possible health states are: ■ HEALTH_OK ■ HEALTH_WARN ■ HEALTH_CRITICAL ■ HEALTH_FAILED ■ HEALTH_OVERLOADED ■ UNKNOWN

Table 2–6 (Cont.) REST Resources for Monitoring a Specified Application Deployed In a Domain

Item Attributes	Type	Valid Values
`targetStates`	object	Application running status for each target: ■ `target` – Name of the target. ■ `state` – Application running status for the specified target. Possible states are: ■ STATE_NEW ■ STATE_FAILED ■ STATE_RETIRED ■ STATE_PREPARED ■ STATE_ADMIN ■ STATE_ACTIVE ■ STATE_UPDATE_ PENDING
`dataSources`	object	Run-time information for the embedded data sources: ■ `name` – Name of the data source instance. ■ `server` – Name of the server on which the data source instance is running. ■ `state` – Current state of the data source instance. Possible states are: ■ Running ■ Suspend ■ Shutdown ■ Overloaded ■ Unknown
`workManagers`	object or number	Statistics for the Work Managers that are configured for an application: ■ `name` – (string) Name of the Work Manager. ■ `server` – (string) Name of the server the Work Manager is associated with. ■ `pendingRequests` – (number) Number of pending requests waiting in the queue. ■ `completedRequests` – (number) Number of requests that have been processed.

Table 2–6 (Cont.) REST Resources for Monitoring a Specified Application Deployed In a Domain

Item Attributes	Type	Valid Values
`minThreadsConstraints`	object or number	Statistics for the minimum thread constraints that are configured for an application: ■ `name` – (string) Name of the minimum thread constraint. ■ `server` – (string) Name of the sever the minimum thread constraint is associated with. ■ `pendingRequests` – (number) Number of pending requests waiting for an available thread. ■ `completedRequests` – (number) Number of completed requests that have been processed. ■ `executingRequests` – (number) Number of requests that are currently executing. ■ `outOfOrderExecutionCount` – (number) Number of requests executed out of turn to satisfy the constraint. ■ `mustRunCount` – (number) Number of requests that must be executed to satisfy the constraint. ■ `maxWaitTime` – (number) Maximum amount of time a request had to wait for a thread. Only requests whose execution is needed to satisfy the constraint are considered. ■ `currentWaitTime` – (number) Last measured amount of time a request had to wait for a thread. Only requests whose execution is needed to satisfy the constraint are considered.
`maxThreadsConstraints`	object or number	Statistics for maximum thread constraints that are configured for an application: ■ `name` – (string) Name of the maximum thread constraint. ■ `server` – (string) Name of the server the maximum thread constraint is associated with. ■ `executingRequests` – (number) Number of requests that are currently executing. ■ `deferredRequests` – (number) Number of requests that are denied a thread for execution because the constraint is exceeded.

Table 2–6 (Cont.) REST Resources for Monitoring a Specified Application Deployed In a Domain

Item Attributes	Type	Valid Values
requestClasses	object or number	Statistics for the request classes that are configured for an application: ■ name – (string) Name of the request class. ■ server – (string) Name of the server the request is associated with. ■ requestClassType – (string) Type of request class, which can either be "fairshare", "responsetime" or 'context" ■ completedCount – (number) Total number of completed requests since the server was started. ■ totalThreadUse – (number) Total amount of thread use time in millisecond used by the request class since the server was started. ■ pendingRequestCount – (number) Number of pending requests waiting for a thread to become available. ■ virtualTimeIncrement – (number) Current priority of the request class. The priority is relative to other request class priorities. The priority is frequently, dynamically calculated, and so can change.

Here is an example of an /applications/{applicationname} URL response in JSON format:

```
{
  "body": {
    "item": {
      "name": "appscopedejbs",
      "type": "ear",
      "health": " HEALTH_OK ",
      "state": "STATE_ACTIVE",
      "targetStates": [
        {
          "target": "ms1",
          "state": "STATE_ACTIVE",
        },
        {
          "name": "ms2",
          "state": "STATE_ACTIVE",
        }
      ] ,
      "dataSources": [
      ],
     "entities": [
      ],
      "workManagers": [
        {
          "name": "default",
          "server": "ms1",
          "pendingRequests": 0,
          "completedRequests": 0
        }
      ],
      "minThreadsConstraints": [
        {
```

```
      "name": "minThreadsConstraints-0",
      "server": "ms1",
      "completedRequests"; 0,
      "pendingRequests"; 0,
      "executingRequests": 0,
      "outOfOrderExecutionCount":0,
      "mustRunCount": 0,
      "maxWaitTime": 0,
      "currentWaitTime": 0
    }
  ],
  "maxThreadsConstraints": [
    {
      "name": "maxThreadsConstraints-0",
      "server": "ms1",
      "executingRequests": 0,
      "deferredRequests": 0
    }
  ],
  "requestClasses": [
    {
      "name": "requestClasses-0",
      "server": "ms1",
      "requestClassType"; "fairshare",
      "completedCount"; 0,
      "totalThreadUse": 0,
      "pendingRequestCount":0,
      "virtualTimeIncrement": 0
     }
    ]
   }
  },
  "messages": [
  ]
}
```

2.4.7 REST Resources for Monitoring All Data Sources In a Domain

The `/datasources` URL listed below returns all generic and GridLink JDBC data sources configured in the domain, and provides run-time information for each data source.

`http(s)://`*`host:port`*`/management/tenant-monitoring/datasources`

Table 2–7 REST Resources for Monitoring All Data Sources In a Domain

Item Attributes (compact format)	Type	Valid Values
`name`	string	Name of the data source.
`type`	string	Data source type; either `Generic` or `GridLink`.
`instances`	string	Run-time information for the data source instances: ■ `server` – The server instance on which the data source instance is running. ■ `state` – The current state of the data source instance. Possible states are: ■ Running ■ Suspended ■ Shutdown ■ Overloaded ■ Unknown

Here is an example of a `/datasources` URL response in JSON format:

```
{
  "body": {
    "items": [
      {
        "name": "genericDS",
        "type": "Generic"
        "instances": [
          {
            "server": "ms1",
            "state": "Running"
          },
          {
            "server": "ms2",
            "state": "Suspended"
          }
        ]
      },
      {
        "name": "gridlinkDS",
        "type": "GridLink",
        "instances": [
          {
            "server": "ms1",
            "state": "Running"
          },
         ]
       }
     ],
  },
  "messages": [
   ]
  }
```

2.4.8 REST Resources for Monitoring a Specified Data Source In a Domain

The `/datasources/{datasourcename}` URL listed below returns run-time information for the specified data source, including Oracle RAC statistics for GridLink data sources.

`http(s)://`*`host:port`*`/management/tenant-monitoring/datasources/{datasourcename}`

Table 2–8 REST Resources for Monitoring a Specified Data Source In a Domain

Item Attributes (compact format)	Type	Valid Values
`name`	string	Name of the data source.
`type`	string	Data source type; either `Generic` or `GridLink`.
`instances`	object or number	Run-time information for the data source instances: ■ `server` – (string) Name of server on which the data source instance is running. ■ `state` – (string) Current state of the data source instance. Possible states are: ■ Running ■ Suspended ■ Shutdown ■ Overloaded ■ Unknown ■ `enabled` – (boolean) True if the data source is enabled. ■ `versionJDBCDriver` – (string) Driver class name of the JDBC driver used to create database connections. ■ `activeConnectionsAverageCount` – (number) Average number of active connections in the data source. ■ `activeConnectionsCurrentCount` – (number) Number of connections currently in use by applications. ■ `activeConnectionsHighCount` – (number) Highest number of active database connections in the instance of the data source since the data source was instantiated. ■ `connectionDelayTime` – (number) Average amount of time, in milliseconds, that it takes to create a physical connection to the database. ■ `connectionsTotalCount` – (number) Cumulative total number of database connections created in the data source since the data source was deployed. ■ `currCapacity` – (number) Current count of JDBC connections in the connection pool in the data source. ■ `currCapacityHighCount` – (number) Highest number of database connections available or in use (current capacity) in the data source since it was deployed ■ `failedReserveRequestCount` – (number) Cumulative running count of requests for a connection from the data source that could not be fulfilled. ■ `failuresToReconnectCount` – (number) Number of failed attempts to refresh a database connection.

Table 2–8 (Cont.) REST Resources for Monitoring a Specified Data Source In a Domain

Item Attributes (compact format)	Type	Valid Values
`instances` *(Cont.)*	object or number	■ `highestNumAvailable` – (number) Highest number of database connections that were idle and available to be used by an application at any time in the data source since it was deployed ■ `leakedConnectionCount` – (number) Number of leaked connections. ■ `numAvailable` – (number) Number of database connections that are currently idle and available to be used by applications in the data source. ■ `numUnavailable` – (number) Number of connections currently in use for the pool ■ `prepStmtCacheAccessCount` – (number) Cumulative running count of the number of times that the statement cache was accessed. ■ `prepStmtCacheAddCount` – (number) Cumulative running count of the number of statements added to the statement cache. ■ `prepStmtCacheCurrentSize` – (number) Number of prepared and callable statements currently cached in the statement cache. ■ `prepStmtCacheDeleteCount` – (number) Cumulative running count of statements discarded from the cache. ■ `prepStmtCacheHitCount` – (number) Cumulative running count of the number of times that statements from the cache were used. ■ `prepStmtCacheMissCount` – (number) Number of times that a statement request could not be satisfied with a statement from the cache. ■ `reserveRequestCount` – (number) Cumulative running count of requests for a connection from the data source. ■ `waitSecondsHighCount` – (number) Highest number of seconds that an application waited for a connection (the longest connection reserve wait time) from the connection pool since the connection pool was instantiated. ■ `waitingForConnectionCurrentCount` – (number) Number of connection requests waiting for a database connection. ■ `waitingForConnectionFailureTotal` – (number) Cumulative running count of requests for a connection from the data source that had to wait before getting a connection and eventually failed to get a connection. ■ `waitingForConnectionHighCount` – (number) Highest number of application requests concurrently waiting for a connection from the data source.

Table 2–8 (Cont.) REST Resources for Monitoring a Specified Data Source In a Domain

Item Attributes (compact format)	Type	Valid Values
`instances` *(Cont.)*	object or number	■ `waitingForConnectionSuccessTotal` – (number) Cumulative running count of requests for a connection from the data source that had to wait before getting a connection and eventually succeeded in getting a connection. ■ `waitingForConnectionTotal` – (number) Cumulative running count of requests for a connection from the data source that had to wait before getting a connection, including those that eventually got a connection and those that did not get a connection. ■ `successfulRclbBasedBorrowCount` – (number) Number of reserve requests for which existing connections were found that satisfied the run-time connection load balancing policy. ■ `failedRCLBBasedBorrowCount` – (number) Number of reserve requests for which a connection was not found based on the run-time connection load balancing policy. ■ `successfulAffinityBasedBorrowCount` – (number) Number of reserve requests for which an existing connection was found that satisfied the affinity policy. ■ `failedAffinityBasedBorrowCount` – (number) Number of reserve requests for which an existing connection for the affinity policy was not found. ■ `racInstances` – Oracle RAC instance statistics: ■ `instanceName` – (string) Name of the Oracle RAC instance. ■ `state` – (string) Current state of the Oracle RAC instance within the data source: Running, Suspended, Shutdown, Overloaded, and Unknown. ■ `enabled` – (boolean) True if the Oracle RAC instance is enabled. ■ `signature` – (string) Signature that uniquely identifies the Oracle RAC instance. ■ `currentWeight` – (number) Current weight of the Oracle RAC instance. ■ `activeConnectionsCurrentCount` – (number) Number of connections currently in use by applications. ■ `reserveRequestCount` – (number) Cumulative running count of

Table 2–8 (Cont.) REST Resources for Monitoring a Specified Data Source In a Domain

Item Attributes (compact format)	Type	Valid Values
`instances` *(Cont.)*	object or number	■ `racInstances` (Cont.): ■ `connectionsTotalCount` – (number) Cumulative total number of database connections created in the Oracle RAC instance since the data source was deployed. ■ `currCapacity` – (number) Current count of JDBC connections in the connection pool in the data source for the Oracle RAC instance. ■ `numAvailable` – (number) Number of database connections currently available (not in use) in the data source for the Oracle RAC instance. ■ `numUnavailable` – (number) Number of database connections that are currently unavailable (in use or being tested by the system) in the Oracle RAC instance.

Here is an example of a `/datasources/{datasourcename}` URL response in JSON format:

```
{
  "body": {
   "item": {
     "name": "gridlinkDS",
     "instances": [
     {
       "server": "ms1",
       "state": 'Running",
       "enabled": true,
       "activeConnectionsAverageCount": 5,
       …
       "racInstances": [
         {
           "instanceName": "gridlinkDS-0",
           "state": "Running",
           …
         },
         {
```

```
            "instanceName": "gridlinkDS-1",
            "state": "Suspended",
            …
          }
         ]
        }

     }
    },
    "messages": [
   ]
}
```