Oracle® Fusion Middleware

RESTful Management Interface Reference for Oracle WebLogic Server 12.1.3

12*c* (12.1.3)

E49342-02

August 2015

This reference describes a RESTful management API for managing a WebLogic Server 12.1.3 domain.

ORACLE

Oracle Fusion Middleware RESTful Management Interface Reference for Oracle WebLogic Server 12.1.3, 12*c* (12.1.3)

E49342-02

Contents

2 Entities

Preface

This preface describes the document accessibility features and conventions used in this guide, *RESTful Management Interface Reference for Oracle WebLogic Server 12.1.3*.

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 Resources

This section describes the Oracle WebLogic Server 12.1.3 RESTful management resources.

The RESTful management resources provide a comprehensive public interface for configuring, monitoring, and administering the Oracle WebLogic Server platform in all supported environments.

> **Note:** The URL format for REST management APIs has changed in 12.1.3. The URL format introduced in 12.1.2 will continue to work, but is deprecated in this release of WebLogic Server.

Getting Started

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*. Alternatively, using WebLogic Scripting Tool (WLST), set the RestfulManagementService `Enabled` property on the DomainMBean.

REST URL Format to Manage WebLogic Server

The format of the WebLogic Server resource URL is:

`http(s)://`*`host:port`*`/management/wls/{version}/`*`path`*

where:

`host`—the host where WebLogic Server is running

`port`—the HTTP or HTTPS port

{*`version`*}—the version of a REST resource. See "The {version} Specifier in Resource URLs".

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

The {version} Specifier in Resource URLs

The {version} specifier in resource URLs implicitly (`/latest`) or explicitly (`/12.1.3.0`) denotes the version of a REST resource. In future WLS releases, you can specify `/latest` to use the current version of a REST resource or a version number to specify a previous version.

As a best practice, if you are using a browser or a reflexive client that follows links in the resources for navigation and can react to evolving resources, then use `/latest`. If you are writing code that would break as a resource evolves, then specify an explicit version.

Supported Content Types in REST Resources

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

- The data format that can you get back from a resource (for example, through the Accept header) – only JSON.
- The data format that can you send in to a resource (for example, through the Content-Type header) – most support only JSON, however, some also support multi-part forms (for example, the file upload versions of deploying applications and libraries).

Using REST Resource Methods to Manage WebLogic Server

The WebLogic Server REST interfaces support methods for accessing objects in the configuration and monitoring object trees.

Table 1–1 shows the REST methods for administering monitoring and configuration data and the tasks that you can perform with each method.

Table 1–1 REST Resource Methods for Administering Configuration Data

Task	REST Method
Determine the methods and method parameters that an object in the tree supports.	`OPTIONS`
Retrieve data for an object in the tree.	`GET`
Add an object to the tree.	`POST`
Update an object in the tree.	`POST`
Delete an object from the tree.	`DELETE`

Use the following procedures with WebLogic Server REST resources.

Creating

To create a new entity:

1. Invoke `OPTIONS` to get a template object containing default values for the new entity.
2. Change the values of any properties you want to customize on the returned object.
3. `POST` the modified object to create the new entity.

Editing

To change some properties of an entity:

1. Invoke `GET` on the entity's URL to get back an object containing the entity's current values.
2. Change the values of the properties you want to modify on the returned object.
3. `POST` the modified object back to the entity's URL.

Asynchronous Operations

For long running operations, specify the `__detached=true` query parameter (asynchronous invocation). The method runs in detached (background) mode and returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Filtering Results

Specify `_includeFields` and `_excludeFields` query parameters to constrain the fields included in a response. Filtering collection results to only the information that you require may yield faster response times when interacting with large data sets. Include and exclude are mutually exclusive, and if neither is specified, all available information is included.

Uploading Files

Use multi-part form data to pass a stream of bytes from the client to the WLS resource. For example, when the files you want to deploy (or redeploy) are not on the Administration Server file system. Use the `POST accepting "multipart/form-data"` method to upload the files from the client's file system to the Administration Server and then deploy (or redeploy) them.

Navigating the REST Resource Tree Dynamically

The WebLogic Server REST interfaces support navigating the resource tree dynamically. To provide this support, the GET method on each REST resource returns a `links` element containing a list of URIs to the resources above and below it in the resource tree. Thus, you can navigate the resource tree using this technique:

1. Make an HTTP GET request on a known REST resource, specifying a literal, hard-coded value as the URI.
2. Parse the `links` list in the response to identify the appropriate parent or child resource. See "Parsing the links List" for more information.
3. If you are done navigating and the REST resource you identified in Step 2 is the one on which you want to perform operations, use the appropriate HTTP requests to perform the operations, using the URI value you retrieved from the `links` list.

   If you need to continue navigating the resource tree, make an HTTP GET request on the REST resource you identified in Step 2, using the URI value you retrieved from the `links` list. Then, return to Step 2.

Using dynamic navigation of the resource tree can provide a variety of benefits, including:

- More maintainable source code, because it does not include a literal, hard-coded URI value for each HTTP request.
- The ability to perform actions across collections of objects, such as servers and applications, whose names are not known during code development.

Parsing the links List

The `links` element returned as part of the response to an HTTP GET request is a list of subordinate elements, each describing a single resource that is directly related to the current resource. Each of these subordinate elements contains the following properties:

rel

A string value indicating how the linked resource is related to the current resource. The string values are as follows:

parent
The linked resource is the parent of the current resource.

action
The linked resource performs an action on the object represented by the current resource. The value of the `title` property indicates the type of action the linked resource performs.

simple-name
A string without dots (`.`). The linked resource has a relationship to the current resource that is specific to the current resource. For example, the GET method on the `/management/wls/{version}` resource returns `link` list elements with *`simple-name`* values corresponding to the types of objects in a domain:

- `changemanger`
- `datasources`
- `deployments`
- `jobs`
- `servers`
- `targets`

dotted-name
A string containing one dot(`.`). The linked resource is a member of a collection of resources subordinate to the current resource, and another element in the GET response provides more information about the linked resource. An example of such a collection is the individual servers under the `/management/wls/{version}/servers` resource.

The first part of the *`dotted-name`* value is the name of a list element in the GET response that contains more information, and the second part is the name of the property in that list element's subelement that identifies each subelement:

`list-element-name.subelement-property-name`

To discover more information about the linked resource, use its *`dotted-name`* value in conjunction with its `title` value:

1. Parse the *`dotted-name`* value into its two parts: *`list-element-name`* and *`subelement-property-name`*.
2. In the GET response, look for a list element named *`list-element-name`*. In this list, look for a subelement whose *`subelement-property-name`* property value matches the linked resource's `title` property value.

For example, given the *`dotted-name`* value **`items.name`** and the `title` value **`myserver`**, you would look in the GET response for the subelement of the **`items`** list whose **`name`** property value is the string **`myserver`**.

uri
A string value containing the URI to use in an HTTP request to access the linked resource.

title
(Optional) A string value providing additional information when the `rel` property value is `action` or *`dotted-name`*:

- When `rel` property value is `action`, the `title` property value describes the kind of action the linked resource performs.
- When the `rel` property value is *`dotted-name`*, the `title` property value is the value to match in the subelement property of the list element specified by the *`dotted-name`* value.

Standard HTTP Status Codes

The HTTP methods used to manipulate the resources described in this section all return one of the following HTTP status codes:

200 OK

The request was successfully completed. A 200 is returned for successful GET or OPTIONS methods, or a POST method to modify a resource (versus create).

201 Created

The request has been fulfilled and resulted in a new resource being created. A Location header containing the canonical URI for the newly created resource will be returned.

A 201 is returned from a synchronous resource creation or an asynchronous resource creation that completed before the response was returned.

202 Accepted

The request has been accepted for processing, but the processing has not been completed. The request might or might not eventually be acted upon, as it might be disallowed when processing actually takes place.

When specifying an asynchronous (`__detached=true`) resource creation, (such as when deploying an application) or update (such as when redeploying an application), a 202 is returned if the operation is still in progress. If `__detached=false`, a 202 might still be returned if the underlying operation does not finish in a reasonable amount of time.

The response contains a Location header of a job resource that the client should poll to find out when the job has finished. Also, it returns an entity that contains the current state of the job.

400 Bad Request

The request could not be processed because it contains missing or invalid information (such as a validation error on an input field, a missing required value, or such).

401 Unauthorized

The request is not authorized. The authentication credentials included with this request are missing or invalid.

403 Forbidden

Cannot authenticate the user. The user does not possess authorization to perform this request.

404 Not Found

The request specified a URI of a resource that does not exist.

405 Method Not Allowed

The HTTP verb specified in the request (DELETE, GET, HEAD, POST, PUT) is not supported for this request URI.

406 Not Acceptable
The resource identified by this request is not capable of generating a representation corresponding to one of the media types in the Accept header of the request. For example, the client's Accept header asks for XML but the resource can only return JSON.

415 Not Acceptable
The client's ContentType header is wrong (for example. the client tries to send in XML but the resource can only accept JSON).

500 Internal Server Error
The server encountered an unexpected condition which prevented it from fulfilling the request.

503 Service Unavailable
The server is currently unable to handle the request due to temporary overloading or maintenance of the server. The WLS REST web application is not currently running.

/management/wls

The wls resource contains information about the versions of the REST interface supported by the WLS domain.

For information about versions supported by the REST interface, see The {version} Specifier in Resource URLs.

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns information about each supported version of this REST interface.

For information about versions supported by the REST interface, see The {version} Specifier in Resource URLs.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a list Version entities for the supported versions of this REST interface.

This method can return the following links:

- **uri**=/management/wls/{versions} **rel**=versions
- **uri**=/management/wls/12.1.3.0 **rel**=items.version **title**=12.1.3.0
- **uri**=/management/wls/latest **rel**=items.version **title**=latest
- **uri**=/management/wls/12.1.3.0 **rel**=current

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Determining Supported Versions

This example demonstrates how to obtain information about each supported version of the REST resources.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "items.version",
            "uri": "http:\/\/localhost:7001\/management\/wls\/12.1.3.0",
            "title": "12.1.3.0"
        },
        {
            "rel": "items.version",
            "uri": "http:\/\/localhost:7001\/management\/wls\/latest",
```

```
                "title": "latest"
            },
            {
                "rel": "current",
                "uri": "http:\/\/localhost:7001\/management\/wls\/12.1.3.0"
            }
        ],
        "items": [
            {
                "version": "12.1.3.0",
                "lifecycle": "active",
                "isLatest": true
            },
            {
                "version": "latest",
                "lifecycle": "active",
                "isLatest": true
            }
        ]
    }
```

/management/wls/{version}

This resource contains overall information about the WLS domain.

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns overall information about this domain, for example, aggregated server runtime statistics.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a Wls entity.

This method can return the following links:

- **uri**=/management/wls/{version}/jobs **rel**=jobs
- **uri**=/management/wls/{version}/servers **rel**=servers
- **uri**=/management/wls/{version}/deployments **rel**=deployments
- **uri**=/management/wls/{version}/datasources **rel**=datasources
- **uri**=/management/wls/{version}/changeManager **rel**=changeManager
- **uri**=/management/wls/{version}/targets **rel**=targets
- **uri**=/management/wls **rel**=parent

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Getting Information About a Domain

This example obtains the root WLS resource links and summary information for a domain.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest
```

Example Response

```
HTTP/1.1 200 OK
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "parent",
            "uri": "http:\/\/localhost:7001\/management\/wls"
        },
        {
            "rel": "jobs",
```

```
                "uri": "http:\/\/localhost:7001\/management\/wls\/latest\/jobs"
            },
            {
                "rel": "servers",
                "uri": "http:\/\/localhost:7001\/management\/wls\/latest\/servers"
            },
            {
                "rel": "deployments",
                "uri": "http:\/\/localhost:7001\/management\/wls\/latest\/deployments"
            },
            {
                "rel": "datasources",
                "uri": "http:\/\/localhost:7001\/management\/wls\/latest\/datasources"
            },
            {
                "rel": "changeManager",
                "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/changeManager"
            },
            {
                "rel": "targets",
                "uri": "http:\/\/localhost:7001\/management\/wls\/latest\/targets"
            }
        ],
        "item": {
            "name": "mydomain",
            "productionMode": false,
            "activeHttpSessionCount": 0,
            "activeThreadCount": 5,
            "configuredServerCount": 3,
            "activeServerCount": 1,
            "overallServiceHealth": {"state": "ok"}
        }
}
```

/management/wls/{version}/changeManager

The changeManager resource returns information about edit sessions. Use this resource to explicitly manage an edit session:

1. Use the changeManager to start an edit session.
2. Use other resources to make any number of configuration changes, such as configuring data sources and deploying applications.
3. Use the changeManager to either cancel all the changes in the edit session or activate them.

This strategy batches all the changes together which is typically most useful in a production environment when a number of configuration changes need to be made together before they are ready to be exposed to clients.

With implicit edit sessions, instead of using the changeManager, just use other resources to change the configuration (for example, create a data source). Each REST call to change the configuration will start its own edit session, make the change, and then activate the change. This is useful for small tweaks to the configuration as typically performed in development environments.

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns information about the current status of the edit session.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a ChangeManager entity.

This method can return the following links:

- **uri**=/management/wls/{version} **rel**=parent
- **uri**=/management/wls/{version}/changeManager/startEdit **rel**=action **title**=startEdit
- **uri**=/management/wls/{version}/changeManager/cancelEdit **rel**=action **title**=cancelEdit
- **uri**=/management/wls/{version}/changeManager/activate **rel**=action **title**=activate

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing Change Management Information

This example uses the GET method to return change manager information.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/changeManager
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "parent",
            "uri": "http:\/\/localhost:7001\/management\/wls\/latest"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/changeManager\/startEdit",
            "title": "startEdit"
        },
```

```
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/changeManager\/cancelEdit",
            "title": "cancelEdit"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/changeManager\/activate",
            "title": "activate"
        }
    ],
    "item": {"locked": false}
}
```

/management/wls/{version}/changeManager/activate

This resource activates the changes.

The resource supports the following methods:

- POST Method

POST Method

The POST method on this resource activates changes saved during the current editing session but not yet deployed.

Roles

Administrator

Response Body

A success response body includes status messages returned by this operation.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Activating Configuration Changes

This example uses the POST method to activate any outstanding changes in the current session.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST http://localhost:7001/management/wls/latest/changeManager/activate
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"messages": [{
    "message": "The changes are activated successfully.",
    "severity": "SUCCESS"
}]}
```

/management/wls/{version}/changeManager/cancelEdit

This resource cancels the current edit session.

The resource supports the following methods:

- POST Method

POST Method

The POST method on this resource cancels the current edit session, releases the edit lock, and discards saved or unsaved changes.

Roles

Administrator

Response Body

A success response body includes status messages returned by this operation.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Stopping an Edit Session

This example uses the POST method to cancel the current edit session.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST http://localhost:7001/management/wls/latest/changeManager/cancelEdit
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"messages": [{
    "message": "Edit session has been cancelled successfully.",
    "severity": "SUCCESS"
}]}
```

/management/wls/{version}/changeManager/startEdit

This resource starts a configuration edit session.

The resource supports the following methods:

- POST Method

POST Method

The POST method on this resource starts the edit session.

Roles

Administrator

Response Body

A success response body includes status messages returned by this operation.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Starting an Edit Session

This example uses the POST method to begin a new edit session.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST http://localhost:7001/management/wls/latest/changeManager/startEdit
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"messages": [{
    "message": "Started edit session, please be sure to save and activate your
changes once you are done.",
    "severity": "SUCCESS"
}]}
```

/management/wls/{version}/datasources

This resource manages the data sources that run in this WLS domain.

The resource supports the following methods:

- GET Method
- OPTIONS Method
- POST Method

GET Method

The GET method on this resource returns a listing of all the data sources in this WLS domain.

Roles

Administrator, Deployer, Operator, Monitor

Request Query Parameters

This method supports optional query parameters which can be used to constrain the fields included in the response. Constraining the fields to only the information that you require may yield faster response times when interacting with large data sets.

?_excludeFields=*field1*[,*field2*, ...]
(Optional) Specifies that these values are not required and may be omitted in a response.

?_includeFields=*field1*[,*field2*, ...]
(Optional) Specifies that these values are required and must be included in the response.

Include and exclude are mutually exclusive, and if neither is specified all available information is included.

Response Body

The response body returned includes a collection of DataSource entities. It also contains links to their corresponding resources.

This method can return the following links:

- **uri**=/management/wls/{version}/datasources/drivers **rel**=drivers
- **uri**=/management/wls/{version}/datasources/test **rel**=test
- **uri**=/management/wls/{version} **rel**=parent
- **uri**=/management/wls/{version}/datasources/id/{data-source-name} **rel**=items.name **title**=name

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing Data Sources

This example uses the GET method to display information about all configured data sources.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/datasources
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "parent",
            "uri": "http:\/\/localhost:7001\/management\/wls\/latest"
        },
        {
            "rel": "vendors",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors"
        },
        {
            "rel": "test",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/test"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/id\/TestDataSource
",
            "title": "TestDataSource"
        }
    ],
    "items": [{
        "name": "TestDataSource",
        "targets": [
            "myserver",
            "Cluster-0"
        ],
        "jdbcDataSourceParams": {
            "scope": "global",
            "dataSourceList": null,
            "globalTransactionsProtocol": "one phase commit",
            "rowPrefetchSize": 48,
            "streamChunkSize": 256,
            "algorithmType": "failover",
            "connectionPoolFailoverCallbackHandler": null,
            "failoverRequestIfBusy": false,
            "rowPrefetch": false,
            "jndiNames": [
                "jndiName1",
                "jndiName2"
            ],
            "keepConnAfterLocalTx": true,
            "keepConnAfterGlobalTx": false
        },
        "jdbcConnectionPoolParams": {
            "driverInterceptor": "",
            "connectionHarvestMaxCount": 1,
            "connectionHarvestTriggerCount": -1,
            "minCapacity": 1,
            "profileType": 0,
            "connectionLabelingCallback": "",
            "maxCapacity": 15,
```

```
        "highestNumWaiters": 2147483647,
        "loginDelaySeconds": 0,
        "secondsToTrustAnIdlePoolConnection": 10,
        "initialCapacity": 1,
        "shrinkFrequencySeconds": 900,
        "testFrequencySeconds": 120,
        "testTableName": "SQL SELECT 1 FROM SYS.SYSTABLES",
        "testConnectionsOnReserve": false,
        "connectionReserveTimeoutSeconds": 10,
        "connectionCreationRetryFrequencySeconds": 0,
        "inactiveConnectionTimeoutSeconds": 0,
        "statementCacheSize": 10,
        "statementCacheType": "least recently used",
        "statementTimeout": -1,
        "countOfTestFailuresTillFlush": 2,
        "countOfRefreshFailuresTillDisable": 2,
        "profileHarvestFrequencySeconds": 300,
        "initSql": "",
        "fatalErrorCodes": "",
        "removeInfectedConnections": true,
        "jdbcXaDebugLevel": 10,
        "ignoreInUseConnectionsEnabled": true,
        "credentialMappingEnabled": false,
        "pinnedToThread": false,
        "identityBasedConnectionPoolingEnabled": false,
        "wrapTypes": true,
        "wrapJdbc": true
    },
    "jdbcDriverParams": {
        "properties": [
            {
                "name": "portNumber",
                "value": "1527"
            },
            {
                "name": "databaseName",
                "value": "demo;create=true"
            },
            {
                "name": "serverName",
                "value": "localhost"
            }
        ],
        "driverName": "org.apache.derby.jdbc.ClientXADataSource",
        "systemProperties": [],
        "url": "jdbc:derby:\/\/localhost:1527\/demo",
        "useXaDataSourceInterface": true,
        "usePasswordIndirection": false
    },
    "aggregateMetrics": {
        "reserveRequestCount": 0,
        "failedReserveRequestCount": 0,
        "waitingForConnectionTotal": 0,
        "waitingForConnectionSuccessTotal": 0,
        "waitingForConnectionFailureTotal": 0,
        "currCapacityHighCount": 3,
        "state": "Running",
        "prepStmtCacheAccessCount": 0,
        "prepStmtCacheAddCount": 0,
        "prepStmtCacheDeleteCount": 0,
```

```
        "prepStmtCacheCurrentSize": 0,
        "prepStmtCacheHitCount": 0,
        "prepStmtCacheMissCount": 0,
        "currCapacity": 3,
        "numAvailable": 3,
        "highestNumAvailable": 3,
        "numUnavailable": 0,
        "highestNumUnavailable": 0,
        "leakedConnectionCount": 0,
        "failuresToReconnectCount": 0,
        "connectionDelayTime": 238,
        "activeConnectionsCurrentCount": 0,
        "waitingForConnectionCurrentCount": 0,
        "activeConnectionsHighCount": 0,
        "waitingForConnectionHighCount": 0,
        "waitSecondsHighCount": 0,
        "connectionsTotalCount": 3,
        "activeConnectionsAverageCount": 0
    },
    "dataSourceMetrics": [
        {
            "reserveRequestCount": 0,
            "failedReserveRequestCount": 0,
            "waitingForConnectionTotal": 0,
            "waitingForConnectionSuccessTotal": 0,
            "waitingForConnectionFailureTotal": 0,
            "currCapacityHighCount": 1,
            "state": "Running",
            "prepStmtCacheAccessCount": 0,
            "prepStmtCacheAddCount": 0,
            "prepStmtCacheDeleteCount": 0,
            "prepStmtCacheCurrentSize": 0,
            "prepStmtCacheHitCount": 0,
            "prepStmtCacheMissCount": 0,
            "currCapacity": 1,
            "numAvailable": 1,
            "highestNumAvailable": 1,
            "numUnavailable": 0,
            "highestNumUnavailable": 0,
            "leakedConnectionCount": 0,
            "failuresToReconnectCount": 0,
            "connectionDelayTime": 238,
            "activeConnectionsCurrentCount": 0,
            "waitingForConnectionCurrentCount": 0,
            "activeConnectionsHighCount": 0,
            "waitingForConnectionHighCount": 0,
            "waitSecondsHighCount": 0,
            "connectionsTotalCount": 1,
            "activeConnectionsAverageCount": 0,
            "serverName": "Cluster-0-Server-2"
        },
        {
            "reserveRequestCount": 0,
            "failedReserveRequestCount": 0,
            "waitingForConnectionTotal": 0,
            "waitingForConnectionSuccessTotal": 0,
            "waitingForConnectionFailureTotal": 0,
            "currCapacityHighCount": 1,
            "state": "Running",
            "prepStmtCacheAccessCount": 0,
```

```
                    "prepStmtCacheAddCount": 0,
                    "prepStmtCacheDeleteCount": 0,
                    "prepStmtCacheCurrentSize": 0,
                    "prepStmtCacheHitCount": 0,
                    "prepStmtCacheMissCount": 0,
                    "currCapacity": 1,
                    "numAvailable": 1,
                    "highestNumAvailable": 1,
                    "numUnavailable": 0,
                    "highestNumUnavailable": 0,
                    "leakedConnectionCount": 0,
                    "failuresToReconnectCount": 0,
                    "connectionDelayTime": 238,
                    "activeConnectionsCurrentCount": 0,
                    "waitingForConnectionCurrentCount": 0,
                    "activeConnectionsHighCount": 0,
                    "waitingForConnectionHighCount": 0,
                    "waitSecondsHighCount": 0,
                    "connectionsTotalCount": 1,
                    "activeConnectionsAverageCount": 0,
                    "serverName": "Cluster-0-Server-1"
                },
                {
                    "reserveRequestCount": 0,
                    "failedReserveRequestCount": 0,
                    "waitingForConnectionTotal": 0,
                    "waitingForConnectionSuccessTotal": 0,
                    "waitingForConnectionFailureTotal": 0,
                    "currCapacityHighCount": 1,
                    "state": "Running",
                    "prepStmtCacheAccessCount": 0,
                    "prepStmtCacheAddCount": 0,
                    "prepStmtCacheDeleteCount": 0,
                    "prepStmtCacheCurrentSize": 0,
                    "prepStmtCacheHitCount": 0,
                    "prepStmtCacheMissCount": 0,
                    "currCapacity": 1,
                    "numAvailable": 1,
                    "highestNumAvailable": 1,
                    "numUnavailable": 0,
                    "highestNumUnavailable": 0,
                    "leakedConnectionCount": 0,
                    "failuresToReconnectCount": 0,
                    "connectionDelayTime": 4,
                    "activeConnectionsCurrentCount": 0,
                    "waitingForConnectionCurrentCount": 0,
                    "activeConnectionsHighCount": 0,
                    "waitingForConnectionHighCount": 0,
                    "waitSecondsHighCount": 0,
                    "connectionsTotalCount": 1,
                    "activeConnectionsAverageCount": 0,
                    "serverName": "myserver"
                }
            ]
        }]
}
```

OPTIONS Method

The OPTIONS method on this resource returns a template entity that has been pre-populated with default values.

If you want to create a data source, call this method to get the template, fill in values, such as the driver class name and database URL, then use it in the POST method to create the data source.

Roles

Administrator

Response Body

The response body returned includes a DataSource template entity that can be customized and then used to create a new data source.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Getting a Template Data Source Entity

This example uses the OPTIONS method to obtain a template for a data source.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X OPTIONS http://localhost:7001/management/wls/latest/datasources
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"item": {
    "name": "DataSource-0",
    "targets": [],
    "jdbcDataSourceParams": {
        "scope": "global",
        "dataSourceList": null,
        "globalTransactionsProtocol": "one phase commit",
        "rowPrefetchSize": 48,
        "streamChunkSize": 256,
        "algorithmType": "failover",
        "connectionPoolFailoverCallbackHandler": null,
        "failoverRequestIfBusy": false,
        "rowPrefetch": false,
        "jndiNames": [],
        "keepConnAfterLocalTx": true,
        "keepConnAfterGlobalTx": false
    },
    "jdbcConnectionPoolParams": {
        "driverInterceptor": "",
```

```
        "connectionHarvestMaxCount": 1,
        "connectionHarvestTriggerCount": -1,
        "minCapacity": 1,
        "profileType": 0,
        "connectionLabelingCallback": "",
        "maxCapacity": 15,
        "highestNumWaiters": 2147483647,
        "loginDelaySeconds": 0,
        "secondsToTrustAnIdlePoolConnection": 10,
        "initialCapacity": 1,
        "shrinkFrequencySeconds": 900,
        "testFrequencySeconds": 120,
        "testTableName": null,
        "testConnectionsOnReserve": false,
        "connectionReserveTimeoutSeconds": 10,
        "connectionCreationRetryFrequencySeconds": 0,
        "inactiveConnectionTimeoutSeconds": 0,
        "statementCacheSize": 10,
        "statementCacheType": "least recently used",
        "statementTimeout": -1,
        "countOfTestFailuresTillFlush": 2,
        "countOfRefreshFailuresTillDisable": 2,
        "profileHarvestFrequencySeconds": 300,
        "initSql": "",
        "fatalErrorCodes": "",
        "removeInfectedConnections": true,
        "jdbcXaDebugLevel": 10,
        "ignoreInUseConnectionsEnabled": true,
        "credentialMappingEnabled": false,
        "pinnedToThread": false,
        "identityBasedConnectionPoolingEnabled": false,
        "wrapTypes": true,
        "wrapJdbc": true
    },
    "jdbcDriverParams": {
        "properties": [],
        "password": null,
        "driverName": null,
        "systemProperties": [],
        "url": null,
        "useXaDataSourceInterface": true,
        "usePasswordIndirection": false
    },
    "aggregateMetrics": {
        "reserveRequestCount": 0,
        "failedReserveRequestCount": 0,
        "waitingForConnectionTotal": 0,
        "waitingForConnectionSuccessTotal": 0,
        "waitingForConnectionFailureTotal": 0,
        "currCapacityHighCount": 0,
        "state": null,
        "prepStmtCacheAccessCount": 0,
        "prepStmtCacheAddCount": 0,
        "prepStmtCacheDeleteCount": 0,
        "prepStmtCacheCurrentSize": 0,
        "prepStmtCacheHitCount": 0,
        "prepStmtCacheMissCount": 0,
        "currCapacity": 0,
        "numAvailable": 0,
        "highestNumAvailable": 0,
```

```
            "numUnavailable": 0,
            "highestNumUnavailable": 0,
            "leakedConnectionCount": 0,
            "failuresToReconnectCount": 0,
            "connectionDelayTime": 0,
            "activeConnectionsCurrentCount": 0,
            "waitingForConnectionCurrentCount": 0,
            "activeConnectionsHighCount": 0,
            "waitingForConnectionHighCount": 0,
            "waitSecondsHighCount": 0,
            "connectionsTotalCount": 0,
            "activeConnectionsAverageCount": 0,
            "serverName": null
        },
        "dataSourceMetrics": []
    }}
```

POST Method

The POST method on this resource creates a new data source.

Roles

Administrator

Request Body

The request body must include a fully populated DataSource entity that describes the new data source. Read only parameter values are ignored.

Response Body

The response body contains a message that states that the data source was created.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Creating a Data Source

This example uses the POST method to create a new data source.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:application/json \
-d "{
    'name': 'TestDataSource',
    'targets': [ 'myserver', 'Cluster-0' ],
    'jdbcDriverParams': {
        'properties': [
             { name: 'portNumber', value: '1527' },
             { name: 'databaseName', value: 'demo;create=true' },
             { name: 'serverName', value: 'localhost' }
        ],
        'password': 'password1',
        'driverName': 'org.apache.derby.jdbc.ClientXADataSource',
        'url': 'jdbc:derby://localhost:1527/demo',
        'systemProperties': [],
        'usePasswordIndirection': false,
        'useXaDataSourceInterface': true
    },
    'jdbcDataSourceParams': {
        'scope': 'global',
        'algorithmType': 'failover',
        'connectionPoolFailoverCallbackHandler': null,
        'failoverRequestIfBusy': false,
        'dataSourceList': null,
        'globalTransactionsProtocol': 'one phase commit',
        'rowPrefetchSize': 48,
        'streamChunkSize': 256,
```

```
        'rowPrefetch': false,
        'keepConnAfterLocalTx': true,
        'keepConnAfterGlobalTx': false,
        'jndiNames': [ 'jndiName1', 'jndiName2' ]
    },
    'jdbcConnectionPoolParams': {
        'profileHarvestFrequencySeconds': 300,
        'driverInterceptor': '',
        'connectionHarvestMaxCount': 1,
        'connectionHarvestTriggerCount': -1,
        'minCapacity': 1,
        'profileType': 0,
        'connectionLabelingCallback': '',
        'maxCapacity': 15,
        'highestNumWaiters': 2147483647,
        'loginDelaySeconds': 0,
        'secondsToTrustAnIdlePoolConnection': 10,
        'initialCapacity': 1,
        'shrinkFrequencySeconds': 900,
        'testFrequencySeconds': 120,
        'testTableName': 'SQL SELECT 1 FROM SYS.SYSTABLES',
        'testConnectionsOnReserve': false,
        'connectionReserveTimeoutSeconds': 10,
        'connectionCreationRetryFrequencySeconds': 0,
        'inactiveConnectionTimeoutSeconds': 0,
        'statementCacheSize': 10,
        'statementCacheType': 'least recently used',
        'statementTimeout': -1,
        'countOfTestFailuresTillFlush': 2,
        'countOfRefreshFailuresTillDisable': 2,
        'fatalErrorCodes': '',
        'initSql': '',
        'ignoreInUseConnectionsEnabled': true,
        'removeInfectedConnections': true,
        'credentialMappingEnabled': false,
        'pinnedToThread': false,
        'identityBasedConnectionPoolingEnabled': false,
        'wrapTypes': true,
        'wrapJdbc': true,
        'jdbcXaDebugLevel': 10
    }
}" \
-X POST http://localhost:7001/management/wls/latest/datasources
```

Example Response

```
HTTP/1.1 201 Created

Location:
http://localhost:7001/management/wls/latest/datasources/id/TestDataSource

Response Body:
{"messages": [{
    "message": "Successfully created 'TestDataSource'.",
    "severity": "SUCCESS"
}]}
```

/management/wls/{version}/datasources/id/{data-source-name}

This resource manages the data source identified by the resource URL.

The resource supports the following methods:

- DELETE Method
- GET Method
- POST Method

DELETE Method

The DELETE method deletes the data source identified by the resource URL.

Roles

Administrator

Response Body

The response body returned contains a message stating that the data source's configuration was deleted.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Deleting a Data Source

This example uses the DELETE method to remove a data source configuration.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X DELETE
http://localhost:7001/management/wls/latest/datasources/id/TestDataSource
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"messages": [{
    "message": "Successfully deleted 'TestDataSource'.",
    "severity": "SUCCESS"
}]}
```

GET Method

The GET method on this resource returns information about the data source identified by the resource URL.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a DataSource entity that contains information about the specified data source.

This method can return the following links:

- **uri**=/management/wls/{version}/datasources **rel**=parent
- **uri**=/management/wls/{version}/datasources/id/{data-source-name}/start **rel**=action **title**=start
- **uri**=/management/wls/{version}/datasources/id/{data-source-name}/shutdown **rel**=action **title**=shutdown
- **uri**=/management/wls/{version}/datasources/id/{data-source-name}/test **rel**=action **title**=test
- **uri**=/management/wls/{version}/datasources/id/{data-source-name}/clearStatementCache **rel**=action **title**=clearStatementCache
- **uri**=/management/wls/{version}/datasources/id/{data-source-name}/reset **rel**=action **title**=reset
- **uri**=/management/wls/{version}/datasources/id/{data-source-name}/shrink **rel**=action **title**=shrink
- **uri**=/management/wls/{version}/datasources/id/{data-source-name}/suspend **rel**=action **title**=suspend
- **uri**=/management/wls/{version}/datasources/id/{data-source-name}/resume **rel**=action **title**=resume

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing a Data Source

This example uses the GET method to find information about a specific data source.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/datasources/id/TestDataSource
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "parent",
            "uri": "http:\/\/localhost:7001\/management\/wls\/latest\/datasources"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/id\/TestDataSource
\/start",
            "title": "start"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/id\/TestDataSource
\/shutdown",
            "title": "shutdown"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/id\/TestDataSource
\/test",
            "title": "test"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/id\/TestDataSource
\/clearStatementCache",
            "title": "clearStatementCache"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/id\/TestDataSource
\/reset",
            "title": "reset"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/id\/TestDataSource
\/shrink",
            "title": "shrink"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/id\/TestDataSource
\/suspend",
            "title": "suspend"
        },
        {
            "rel": "action",
```

```
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/id\/TestDataSource
\/resume",
            "title": "resume"
        }
    ],
    "item": {
        "name": "TestDataSource",
        "targets": [
            "myserver",
            "Cluster-0"
        ],
        "jdbcDataSourceParams": {
            "scope": "global",
            "dataSourceList": null,
            "globalTransactionsProtocol": "one phase commit",
            "rowPrefetchSize": 96,
            "streamChunkSize": 256,
            "algorithmType": "failover",
            "connectionPoolFailoverCallbackHandler": null,
            "failoverRequestIfBusy": false,
            "rowPrefetch": true,
            "jndiNames": [
                "jndiName1",
                "jndiName2",
                "jndiName3"
            ],
            "keepConnAfterLocalTx": true,
            "keepConnAfterGlobalTx": false
        },
        "jdbcConnectionPoolParams": {
            "driverInterceptor": "",
            "connectionHarvestMaxCount": 1,
            "connectionHarvestTriggerCount": -1,
            "minCapacity": 1,
            "profileType": 0,
            "connectionLabelingCallback": "",
            "maxCapacity": 30,
            "highestNumWaiters": 2147483647,
            "loginDelaySeconds": 0,
            "secondsToTrustAnIdlePoolConnection": 10,
            "initialCapacity": 2,
            "shrinkFrequencySeconds": 900,
            "testFrequencySeconds": 120,
            "testTableName": "SQL SELECT 1 FROM SYS.SYSTABLES",
            "testConnectionsOnReserve": false,
            "connectionReserveTimeoutSeconds": 10,
            "connectionCreationRetryFrequencySeconds": 0,
            "inactiveConnectionTimeoutSeconds": 0,
            "statementCacheSize": 10,
            "statementCacheType": "least recently used",
            "statementTimeout": -1,
            "countOfTestFailuresTillFlush": 2,
            "countOfRefreshFailuresTillDisable": 2,
            "profileHarvestFrequencySeconds": 300,
            "initSql": "",
            "fatalErrorCodes": "",
            "removeInfectedConnections": true,
            "jdbcXaDebugLevel": 10,
            "ignoreInUseConnectionsEnabled": true,
```

```
        "credentialMappingEnabled": false,
        "pinnedToThread": false,
        "identityBasedConnectionPoolingEnabled": false,
        "wrapTypes": true,
        "wrapJdbc": true
    },
    "jdbcDriverParams": {
        "properties": [
            {
                "name": "portNumber",
                "value": "1527"
            },
            {
                "name": "databaseName",
                "value": "demo;create=true"
            },
            {
                "name": "serverName",
                "value": "localhost"
            }
        ],
        "driverName": "org.apache.derby.jdbc.ClientXADataSource",
        "systemProperties": [],
        "url": "jdbc:derby:\/\/localhost:1527\/demo",
        "useXaDataSourceInterface": true,
        "usePasswordIndirection": false
    },
    "aggregateMetrics": {
        "reserveRequestCount": 3,
        "failedReserveRequestCount": 0,
        "waitingForConnectionTotal": 0,
        "waitingForConnectionSuccessTotal": 0,
        "waitingForConnectionFailureTotal": 0,
        "currCapacityHighCount": 3,
        "state": "Shutdown",
        "prepStmtCacheAccessCount": 0,
        "prepStmtCacheAddCount": 0,
        "prepStmtCacheDeleteCount": 0,
        "prepStmtCacheCurrentSize": 0,
        "prepStmtCacheHitCount": 0,
        "prepStmtCacheMissCount": 0,
        "currCapacity": 0,
        "numAvailable": 0,
        "highestNumAvailable": 3,
        "numUnavailable": 0,
        "highestNumUnavailable": 3,
        "leakedConnectionCount": 0,
        "failuresToReconnectCount": 0,
        "connectionDelayTime": 238,
        "activeConnectionsCurrentCount": 0,
        "waitingForConnectionCurrentCount": 0,
        "activeConnectionsHighCount": 3,
        "waitingForConnectionHighCount": 0,
        "waitSecondsHighCount": 0,
        "connectionsTotalCount": 3,
        "activeConnectionsAverageCount": 0
    },
    "dataSourceMetrics": [
        {
            "reserveRequestCount": 1,
```

```
                    "failedReserveRequestCount": 0,
                    "waitingForConnectionTotal": 0,
                    "waitingForConnectionSuccessTotal": 0,
                    "waitingForConnectionFailureTotal": 0,
                    "currCapacityHighCount": 1,
                    "state": "Shutdown",
                    "prepStmtCacheAccessCount": 0,
                    "prepStmtCacheAddCount": 0,
                    "prepStmtCacheDeleteCount": 0,
                    "prepStmtCacheCurrentSize": 0,
                    "prepStmtCacheHitCount": 0,
                    "prepStmtCacheMissCount": 0,
                    "currCapacity": 0,
                    "numAvailable": 0,
                    "highestNumAvailable": 1,
                    "numUnavailable": 0,
                    "highestNumUnavailable": 1,
                    "leakedConnectionCount": 0,
                    "failuresToReconnectCount": 0,
                    "connectionDelayTime": 4,
                    "activeConnectionsCurrentCount": 0,
                    "waitingForConnectionCurrentCount": 0,
                    "activeConnectionsHighCount": 1,
                    "waitingForConnectionHighCount": 0,
                    "waitSecondsHighCount": 0,
                    "connectionsTotalCount": 1,
                    "activeConnectionsAverageCount": 0,
                    "serverName": "myserver"
                }
            ]
        }
    }
```

POST Method

The POST method on this resource reconfigures the data source identified by the resource URL.

Roles

Administrator

Request Body

The POST request body must include a fully populated DataSource entity that contains the desired data source configuration. Read only parameter values are ignored.

Response Body

The response body returned contains a message stating that the data source's configuration was updated.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Editing a Data Source

This example uses the POST method to update a data source configuration.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:application/json \
-d "{
    'name': 'TestDataSource',
    'targets': [ 'myserver', 'Cluster-0' ],
    'jdbcDriverParams': {
        'properties': [
             { name: 'portNumber', value: '1527' },
             { name: 'databaseName', value: 'demo;create=true' },
             { name: 'serverName', value: 'localhost' }
        ],
        'password': 'password1',
        'driverName': 'org.apache.derby.jdbc.ClientXADataSource',
        'url': 'jdbc:derby://localhost:1527/demo',
        'systemProperties': [],
        'usePasswordIndirection': false,
        'useXaDataSourceInterface': true
    },
    'jdbcDataSourceParams': {
        'scope': 'global',
        'algorithmType': 'failover',
        'connectionPoolFailoverCallbackHandler': null,
        'failoverRequestIfBusy': false,
        'dataSourceList': null,
```

```
        'globalTransactionsProtocol': 'one phase commit',
        'rowPrefetchSize': 96,
        'streamChunkSize': 256,
        'rowPrefetch': true,
        'keepConnAfterLocalTx': true,
        'keepConnAfterGlobalTx': false,
        'jndiNames': [ 'jndiName1', 'jndiName2', 'jndiName3' ]
    },
    'jdbcConnectionPoolParams': {
        'profileHarvestFrequencySeconds': 300,
        'driverInterceptor': '',
        'connectionHarvestMaxCount': 1,
        'connectionHarvestTriggerCount': -1,
        'minCapacity': 1,
        'profileType': 0,
        'connectionLabelingCallback': '',
        'maxCapacity': 30,
        'highestNumWaiters': 2147483647,
        'loginDelaySeconds': 0,
        'secondsToTrustAnIdlePoolConnection': 10,
        'initialCapacity': 2,
        'shrinkFrequencySeconds': 900,
        'testFrequencySeconds': 120,
        'testTableName': 'SQL SELECT 1 FROM SYS.SYSTABLES',
        'testConnectionsOnReserve': false,
        'connectionReserveTimeoutSeconds': 10,
        'connectionCreationRetryFrequencySeconds': 0,
        'inactiveConnectionTimeoutSeconds': 0,
        'statementCacheSize': 10,
        'statementCacheType': 'least recently used',
        'statementTimeout': -1,
        'countOfTestFailuresTillFlush': 2,
        'countOfRefreshFailuresTillDisable': 2,
        'fatalErrorCodes': '',
        'initSql': '',
        'ignoreInUseConnectionsEnabled': true,
        'removeInfectedConnections': true,
        'credentialMappingEnabled': false,
        'pinnedToThread': false,
        'identityBasedConnectionPoolingEnabled': false,
        'wrapTypes': true,
        'wrapJdbc': true,
        'jdbcXaDebugLevel': 10
    }
}" \
-X POST http://localhost:7001/management/wls/latest/datasources/id/TestDataSource
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"messages": [{
    "message": "Successfully updated 'TestDataSource'.",
    "severity": "SUCCESS"
}]}
```

/management/wls/{version}/datasources/id/{data-source-name}/clearStatementCache

This resource clears the statement cache for each connection in the data source. If statement caching is enabled for a data source, WebLogic Server caches prepared and callable statements that are used in each connection in the data source. Each connection has its own cache, but the caches for each connection are configured and managed as a group. When you clear the statement cache for a data source, you clear the statement cache for all connections in the instance of the data source you select.

When you clear a statement that is currently in use by an application, WebLogic Server removes the statement from the cache, but does not close it. When you clear a statement that is not currently in use, WebLogic Server removes the statement from the cache and closes it.

The resource supports the following methods:

- POST Method

POST Method

The POST method on this resource clears the statement cache for each connection in the data source.

Roles

Administrator, Deployer, Operator

Response Body

A success response body includes status messages returned by this operation.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Clearing a Data Source's Statement Caches

This example uses the POST method to clear the statement caches for a specific data source.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/datasources/id/TestDataSource/clearStatementCache
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"messages": [{
    "message": "Data Source 'TestDataSource' was successfully cleared.",
    "severity": "SUCCESS"
}]}
```

/management/wls/{version}/datasources/id/{data-source-name}/reset

This resource resets a data source. When you reset the database connections in a JDBC data source, WebLogic Server closes and recreates all available database connections in the pool of connections in the data source.

The resource supports the following methods:

- POST Method

POST Method

The POST method on this resource resets the database connections for a data source.

Roles

Administrator, Deployer, Operator

Response Body

A success response body includes status messages returned by this operation.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Resetting a Data Source

This example uses the POST method to reset the database connections for a data source.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/datasources/id/TestDataSource/reset
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"messages": [{
    "message": "Data Source 'TestDataSource' was successfully reset.",
    "severity": "SUCCESS"
}]}
```

/management/wls/{version}/datasources/id/{data-source-name}/resume

This resource resumes a suspended data source. When you resume a data source, WebLogic Server marks the data source as enabled and allows applications to reserve connections from the data source. If you suspended the data source (not forcibly suspended), all connections are preserved exactly as they were before the data source was suspended. Clients that had reserved a connection before the data source was suspended can continue exactly where they left off.

NOTE: You cannot resume a data source that did not start correctly, for example, if the database server is unavailable.

The resource supports the following methods:

- POST Method

POST Method

The POST method on this resource resumes individual data sources that are in the `Suspended` state.

Roles

Administrator, Deployer, Operator

Response Body

A success response body includes status messages returned by this operation.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Resuming a Suspended Data Source

This example uses the POST method to resume a suspended data source.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/datasources/id/TestDataSource/resume
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"messages": [{
    "message": "Data Source 'TestDataSource' was successfully resumed.",
    "severity": "SUCCESS"
}]}
```

/management/wls/{version}/datasources/id/{data-source-name}/shrink

This resource shrinks the pool of database connections in individual instances of a data source.

The resource supports the following methods:

- POST Method

POST Method

The POST method on this resource shrinks the connection pool in a data source.

Roles

Administrator, Deployer, Operator

Response Body

A success response body includes status messages returned by this operation.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Shrinking The Connection Pool in a Data Source

This example uses the POST method to shrink the connection pool for a specific data source.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/datasources/id/TestDataSource/shrink
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"messages": [{
    "message": "Data Source 'TestDataSource' was successfully shrunken.",
    "severity": "SUCCESS"
}]}
```

/management/wls/{version}/datasources/id/{data-source-name}/shutdown

This resource shuts down a running data source. When you shut down a data source (not forcibly shut down), WebLogic Server closes database connections in the data source and shuts down the data source. If any connections from the data source are currently in use, the operation will fail.

The resource supports the following methods:

- POST Method

POST Method

The POST method on this resource shuts down a data source.

Roles

Administrator, Deployer, Operator

Response Body

A success response body includes status messages returned by this operation.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Shutting Down a Data Source

This example uses the POST method to shut down a data source.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/datasources/id/TestDataSource/shutdown
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"messages": [{
    "message": "Data Source 'TestDataSource' was successfully stopped.",
    "severity": "SUCCESS"
}]}
```

/management/wls/{version}/datasources/id/{data-source-name}/start

This resource starts a stopped data source. Starting a data source re-initializes the data source, creates connections, and transitions the data source to a health state of `Running`.

The resource supports the following methods:

- POST Method

POST Method

The POST method on this resource starts a data source.

Roles

Administrator, Deployer, Operator

Response Body

A success response body includes status messages returned by this operation.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Starting a Data Source

This example uses the POST method to start a data source.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/datasources/id/TestDataSource/start
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"messages": [{
    "message": "Data Source 'TestDataSource' was successfully started.",
    "severity": "SUCCESS"
}]}
```

/management/wls/{version}/datasources/id/{data-source-name}/suspend

This resource suspends a data source. When you suspend a data source (not forcibly suspend), the data source is marked as disabled and applications cannot reserve connections from the pool. Applications that already have a reserved connection from the data source when it is suspended will get an exception when trying to reserve the connection. WebLogic Server preserves all connections in the data source exactly as they were before the data source was suspended.

The resource supports the following methods:

- POST Method

POST Method

The POST method on this resource suspends individual instances of a data source. When you suspend a data source, applications can no longer get a database connection from the data source.

Roles

Administrator, Deployer, Operator

Response Body

A success response body includes status messages returned by this operation.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Suspending a Data Source

This example uses the POST method to suspend a data source.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/datasources/id/TestDataSource/suspend
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"messages": [{
    "message": "Data Source 'TestDataSource' was successfully suspended.",
    "severity": "SUCCESS"
}]}
```

/management/wls/{version}/datasources/id/{data-source-name}/test

This resource tests a data source's database connection.

The resource supports the following methods:

- POST Method

POST Method

The POST method on this resource tests the data source's database connection.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

A success response body includes status messages returned by this operation.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Testing a Data Source

This example uses the POST method to test a data source's connectivity with a database.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/datasources/id/TestDataSource/test
```

Example Response

```
HHTTP/1.1 200 OK

Response Body:
{"messages": [{
    "message": "Data Source 'TestDataSource' is working correctly.",
    "severity": "SUCCESS"
}]}
```

/management/wls/{version}/datasources/test

This resource tests a database connection. It is typically used to verify the parameters that will be used to create a new data source.

The resource supports the following methods:

- OPTIONS Method
- POST Method

OPTIONS Method

The OPTIONS method on this resource returns a template entity that has been pre-populated with default values.

If you want to test a database connection, call this method to get the template, fill in values, such as the driver class name and connection URL, then use it in the POST method to test the connection.

Roles

Administrator, Deployer

Response Body

The response body returned includes a DatabaseDriverAttributes entity.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Getting a Template For Making a Database Connection For a Specific Driver

This example uses the OPTIONS method to obtain a template for a database connection.

Example Request

Example Response

POST Method

The POST method on this resource converts a set of driver-specific attributes into a generic database connection entity, properly formatting the connection URL and list of properties.

NOTE: Since REST resources are not allowed to return cleartext passwords, the returned driver does not have its password filled in, even though one of the driver attributes holds the password. To compensate for this, the client should hold on to the database driver attributes entity, use its `passwordAttributeName` property to find the attribute that holds the password, then set the `password` property on the returned database connection entity to that password.

Roles

Administrator, Deployer

Request Body

The request body must include a fully populated DatabaseDriverAttributes entity. Read only parameter values are ignored.

Response Body

The response body returned includes a DatabaseConnection entity.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Testing a Database Connection For a Specific Database

This example uses the POST method to test a database connection configuration.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:application/json \
-d "{
    url: 'jdbc:derby://localhost:1527/demo',
    driverName: 'org.apache.derby.jdbc.ClientXADataSource',
    password: null,
    properties: [
        { name: 'portNumber', value: '1527' },
        { name: 'databaseName', value: 'demo;create=true' },
        { name: 'serverName', value: 'localhost' }
    ]
}" \
-X POST http://localhost:7001/management/wls/latest/datasources/test
```

Example Response

```
HTTP/1.1 200 OK
```

```
Response Body:
{"item": {
    "cause": [],
    "ok": true
}}
```

/management/wls/{version}/datasources/vendors

This resource lists the available database driver vendors in this WLS domain.

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns a listing of this domain's available database driver vendors.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a collection of DatabaseVendor entities. Note that their `drivers` property is not populated.

This method can return the following links:

- **uri**=/management/wls/{version}/datasources **rel**=parent
- **uri**=/management/wls/{version}/datasources/vendors/id/{vendorName} **rel**=items.name **title**=name

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing Available Database Driver Vendors

This example uses the GET method to discover all available database driver vendors known to a domain.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/datasources/vendors
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "parent",
            "uri": "http:\/\/localhost:7001\/management\/wls\/latest\/datasources"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Adaba
s%20for%20z%2FOS",
            "title": "Adabas for z\/OS"
        },
        {
            "rel": "items.name",
```

```
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/CICS%
2FTS%20for%20z%2FOS",
            "title": "CICS\/TS for z\/OS"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Cache
",
            "title": "Cache"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Cloud
scape",
            "title": "Cloudscape"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/DB2",
            "title": "DB2"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/DB2%2
0for%20i5%2FOS",
            "title": "DB2 for i5\/OS"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/DB2%2
0for%20z%2FOS",
            "title": "DB2 for z\/OS"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Derby
",
            "title": "Derby"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Enter
priseDB",
            "title": "EnterpriseDB"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/First
SQL",
            "title": "FirstSQL"
        },
```

```
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/IMS%2
FDB%20for%20z%2FOS",
            "title": "IMS\/DB for z\/OS"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/IMS%2
FTM%20for%20z%2FOS",
            "title": "IMS\/TM for z\/OS"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Infor
mix",
            "title": "Informix"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Ingre
s",
            "title": "Ingres"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/MS%20
SQL%20Server",
            "title": "MS SQL Server"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/MaxDB
",
            "title": "MaxDB"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/MySQL
",
            "title": "MySQL"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Oracl
e",
            "title": "Oracle"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Oracl
```

```
e%20BI%20Server",
            "title": "Oracle BI Server"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Oracl
e%20TimesTen%20Client%20Connection",
            "title": "Oracle TimesTen Client Connection"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Oracl
e%20TimesTen%20Direct%20Connection",
            "title": "Oracle TimesTen Direct Connection"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Point
Base",
            "title": "PointBase"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Postg
reSQL",
            "title": "PostgreSQL"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Progr
ess",
            "title": "Progress"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Sybas
e",
            "title": "Sybase"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/VSAM%
20for%20z%2FOS",
            "title": "VSAM for z\/OS"
        }
    ],
    "items": [
        {"name": "Adabas for z\/OS"},
        {"name": "CICS\/TS for z\/OS"},
        {"name": "Cache"},
        {"name": "Cloudscape"},
        {"name": "DB2"},
        {"name": "DB2 for i5\/OS"},
```

```
            {"name": "DB2 for z\/OS"},
            {"name": "Derby"},
            {"name": "EnterpriseDB"},
            {"name": "FirstSQL"},
            {"name": "IMS\/DB for z\/OS"},
            {"name": "IMS\/TM for z\/OS"},
            {"name": "Informix"},
            {"name": "Ingres"},
            {"name": "MS SQL Server"},
            {"name": "MaxDB"},
            {"name": "MySQL"},
            {"name": "Oracle"},
            {"name": "Oracle BI Server"},
            {"name": "Oracle TimesTen Client Connection"},
            {"name": "Oracle TimesTen Direct Connection"},
            {"name": "PointBase"},
            {"name": "PostgreSQL"},
            {"name": "Progress"},
            {"name": "Sybase"},
            {"name": "VSAM for z\/OS"}
        ]
}
```

/management/wls/{version}/datasources/vendors/id/{vendor}

This resource describes a database driver vendor.

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns a listing of this vendor's database drivers.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a DatabaseVendor entity.

This method can return the following links:

- **uri**=/management/wls/{version}/datasources/vendors **rel**=parent
- **uri**=/management/wls/{version}/datasources/vendors/id/{vendor}/id/{driverName} **rel**=item.drivers.name **title**=name

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing a Vendor's Database Drivers

This example uses the GET method to discover all of the supported drivers for a database vendor.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/datasources/vendors/id/Derby
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "parent",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors"
        },
        {
            "rel": "item.drivers.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Derby
\/drivers\/id\/Derby's%20Driver%20(Type%204%20XA)%20Versions:Any",
            "title": "Derby's Driver (Type 4 XA) Versions:Any"
        },
        {
            "rel": "item.drivers.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Derby
```

```
\/drivers\/id\/Derby's%20Driver%20(Type%204%20embedded)%20Versions:Any",
            "title": "Derby's Driver (Type 4 embedded) Versions:Any"
        },
        {
            "rel": "item.drivers.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/datasources\/vendors\/id\/Derby
\/drivers\/id\/Derby's%20Driver%20(Type%204)%20Versions:Any",
            "title": "Derby's Driver (Type 4) Versions:Any"
        }
    ],
    "item": {
        "name": "Derby",
        "drivers": [
            {
                "name": "Derby's Driver (Type 4 XA) Versions:Any",
                "type": "Type 4",
                "description": null,
                "driverClassName": "org.apache.derby.jdbc.ClientXADataSource",
                "forXa": true,
                "testSql": "SELECT 1 FROM SYS.SYSTABLES"
            },
            {
                "name": "Derby's Driver (Type 4 embedded) Versions:Any",
                "type": "Type 4 embedded",
                "description": null,
                "driverClassName": "org.apache.derby.jdbc.EmbeddedDriver",
                "forXa": false,
                "testSql": "SELECT 1 FROM SYS.SYSTABLES"
            },
            {
                "name": "Derby's Driver (Type 4) Versions:Any",
                "type": "Type 4",
                "description": null,
                "driverClassName": "org.apache.derby.jdbc.ClientDriver",
                "forXa": false,
                "testSql": "SELECT 1 FROM SYS.SYSTABLES"
            }
        ]
    }
}
```

/management/wls/{version}/datasources/vendors/id/{vendor}/drivers/id/{driver-class-name}

This resource helps construct a database connection that is appropriate to a specific database driver.

The resource supports the following methods:

- OPTIONS Method
- POST Method

OPTIONS Method

The OPTIONS method on this resource returns a template entity that has been pre-populated with default values.

If you want to construct an appropriate database connection for a specific kind of database driver, find the driver class name, then call this method to get a template that lists the attributes that must be filled in for this driver. After filling in the values, use it in the POST method which converts the completed template into a database connection.

Roles

Administrator, Deployer

Response Body

The response body returned includes a DatabaseDriverAttributes entity.

This method can return the following links:

- **uri**=/management/wls/{version}/datasources/vendors/id/{vendor}/drivers/id/{driver-class-name}/connection **rel**=connection

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Getting a Template For Making a Database Connection For a Specific Driver

This example uses the OPTIONS method to obtain a template for a database driver configuration.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X OPTIONS
http://localhost:7001/management/wls/latest/datasources/vendors/id/Derby/drivers/id/Derby's%20Driver%20(Type%204)%20Versions:Any
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"item": {
    "attributes": [
        {
            "name": "DbmsHost",
            "value": null,
            "defaultValue": null,
            "description": null,
            "inUrl": true,
            "required": true
```

```
            },
            {
                "name": "DbmsPort",
                "value": "1527",
                "defaultValue": "1527",
                "description": null,
                "inUrl": true,
                "required": true
            },
            {
                "name": "DbmsName",
                "value": null,
                "defaultValue": null,
                "description": null,
                "inUrl": true,
                "required": true
            },
            {
                "name": "DbmsUsername",
                "value": null,
                "defaultValue": null,
                "description": null,
                "inUrl": false,
                "required": true
            },
            {
                "name": "DbmsPassword",
                "value": null,
                "defaultValue": null,
                "description": null,
                "inUrl": false,
                "required": true
            }
        ],
        "passwordAttributeName": "DbmsPassword"
    }}
```

POST Method

The POST method on this resource converts a set of driver-specific attributes into a generic database connection entity, properly formatting the connection URL and list of properties.

NOTE: Since REST resources are not allowed to return cleartext passwords, the returned driver does not have its password filled in, even though one of the driver attributes holds the password. To compensate for this, the client should hold on to the database driver attributes entity, use its `passwordAttributeName` property to find the attribute that holds the password, then set the `password` property on the returned database connection entity to that password.

Roles

Administrator, Deployer

Request Body

The request body must include a fully populated DatabaseDriverAttributes entity. Read only parameter values are ignored.

Response Body

The response body returned includes a DatabaseConnection entity.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Creating a Database Connection For a Specific Database

This example uses the POST method to construct a database connection configuration.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:application/json \
-d "{
    attributes: [
        { name: 'DbmsHost', value: 'localhost' },
        { name: 'DbmsPort', value: '1527' },
        { name: 'DbmsName', value: 'db1' },
        { name: 'DbmsUsername', value: 'u1' },
        { name: 'DbmsPassword', value: 'p1' }
    ]
}" \
-X POST
http://localhost:7001/management/wls/latest/datasources/vendors/id/Derby/drivers/id/Derby's%20Driver%20(Type%204)%20Versions:Any
```

Example Response

```
HTTP/1.1 200 OK
```

```
Response Body:
{"item": {
    "properties": [
        {
            "name": "user",
            "value": "u1"
        },
        {
            "name": "portNumber",
            "value": "1527"
        },
        {
            "name": "databaseName",
            "value": "db1;create=true"
        },
        {
            "name": "serverName",
            "value": "localhost"
        }
    ],
    "driverName": "org.apache.derby.jdbc.ClientDriver",
    "url":
"jdbc:derby:\/\/localhost:1527\/db1;ServerName=localhost;databaseName=db1"
}}
```

/management/wls/{version}/deployments

This resource lists the deployments that run in this WLS domain.

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns a listing of all the deployments in this WLS domain.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a collection of Deployment entities. It also contains links to their corresponding resources.

This method can return the following links:

- **uri**=/management/wls/{version}/deployments/inspect **rel**=inspect
- **uri**=/management/wls/{version}/deployments/application **rel**=application
- **uri**=/management/wls/{version}/deployments/library **rel**=library
- **uri**=/management/wls/{version} **rel**=parent
- **uri**=/management/wls/{version}/deployments/application/id/{application-name} **rel**=application **title**=name
- **uri**=/management/wls/{version}/deployments/library/id/{library-name} **rel**=library **title**=name

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing Deployments

This example uses the GET method to display all deployments in a domain.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/deployments
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "parent",
            "uri": "http:\/\/localhost:7001\/management\/wls\/latest"
        },
        {
            "rel": "inspect",
            "uri":
```

```
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/inspect"
        },
        {
            "rel": "application",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/application"
        },
        {
            "rel": "library",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/library"
        },
        {
            "rel": "application",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/application\/id\/B
asicApp",
            "title": "BasicApp"
        },
        {
            "rel": "library",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/library\/id\/airli
ne",
            "title": "airline"
        }
    ],
    "items": [
        {
            "name": "BasicApp",
            "state": "active",
            "type": "application",
            "targets": ["myserver"],
            "displayName": "BasicApp",
            "deploymentPath": "\/deployments\/BasicApp\/app\/BasicApp.ear"
        },
        {
            "name": "airline",
            "state": "active",
            "type": "library",
            "targets": [
                "myserver",
                "Cluster-0"
            ],
            "displayName": "airline",
            "deploymentPath": "\/deployments\/airline.war"
        }
    ]
}
```

/management/wls/{version}/deployments/application

This resource manages the applications that are deployed in this WLS domain.

The resource supports the following methods:

- GET Method
- OPTIONS Method
- POST Method

GET Method

The GET method on this resource returns a listing of all the applications in this WLS domain.

Roles

Administrator, Deployer, Operator, Monitor

Request Query Parameters

This method supports optional query parameters which can be used to constrain the fields included in the response. Constraining the fields to only the information that you require may yield faster response times when interacting with large data sets.

?_excludeFields=*field1*[,*field2*, ...]
(optional) Specifies that these values are not required and may be omitted in a response.

?_includeFields=*field1*[,*field2*, ...]
(optional) Specifies that these values are required and must be included in the response.

Include and exclude are mutually exclusive, and if neither is specified all available information is included.

Response Body

The response body returned includes a collection of Application entities. It also contains links to their corresponding resources.

This method can return the following links:

- **uri**=/management/wls/{version}/deployments **rel**=parent
- **uri**=/management/wls/{version}/deployments/application/id/{application-name} **rel**=items.name **title**=name

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing Applications

This example uses the GET method to list all the applications in this WLS domain.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/deployments/application
```

Example Response

```
HTTP/1.1 200 OK
```

```
Response Body:
{
    "links": [
        {
            "rel": "parent",
            "uri": "http:\/\/localhost:7001\/management\/wls\/latest\/deployments"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/application\/id\/f
airShare",
            "title": "fairShare"
        }
    ],
    "items": [{
        "name": "fairShare",
        "state": "active",
        "type": "application",
        "targets": [
            "myserver",
            "Cluster-0"
        ],
        "servlets": [
            {
                "servletName": "SimpleSlowServlet",
                "contextPath": "\/fairShare",
                "aggregateMetrics": {
                    "executionTimeTotal": 0,
                    "invocationTotalCount": 0,
                    "reloadTotalCount": 0,
                    "executionTimeHigh": 0,
                    "executionTimeLow": 0
                },
                "servletMetrics": [
                    {
                        "serverName": "myserver",
                        "executionTimeTotal": 0,
                        "invocationTotalCount": 0,
                        "reloadTotalCount": 0,
                        "executionTimeHigh": 0,
                        "executionTimeLow": 0
                    }
                ]
            },
            {
                "servletName": "SimpleFastServlet",
                "contextPath": "\/fairShare",
                "aggregateMetrics": {
                    "executionTimeTotal": 0,
                    "invocationTotalCount": 0,
                    "reloadTotalCount": 0,
                    "executionTimeHigh": 0,
                    "executionTimeLow": 0
                },
                "servletMetrics": [
                    {
                        "serverName": "myserver",
                        "executionTimeTotal": 0,
                        "invocationTotalCount": 0,
```

```
                "reloadTotalCount": 0,
                "executionTimeHigh": 0,
                "executionTimeLow": 0
            }
        ]
    },
    {
        "servletName": "JspServlet",
        "contextPath": "\/fairShare",
        "aggregateMetrics": {
            "executionTimeTotal": 0,
            "invocationTotalCount": 0,
            "reloadTotalCount": 0,
            "executionTimeHigh": 0,
            "executionTimeLow": 0
        },
        "servletMetrics": [
            {
                "serverName": "myserver",
                "executionTimeTotal": 0,
                "invocationTotalCount": 0,
                "reloadTotalCount": 0,
                "executionTimeHigh": 0,
                "executionTimeLow": 0
            }
        ]
    },
    {
        "servletName": "FileServlet",
        "contextPath": "\/fairShare",
        "aggregateMetrics": {
            "executionTimeTotal": 0,
            "invocationTotalCount": 0,
            "reloadTotalCount": 0,
            "executionTimeHigh": 0,
            "executionTimeLow": 0
        },
        "servletMetrics": [
            {
                "serverName": "myserver",
                "executionTimeTotal": 0,
                "invocationTotalCount": 0,
                "reloadTotalCount": 0,
                "executionTimeHigh": 0,
                "executionTimeLow": 0
            }
        ]
    }
],
"displayName": "fairShare",
"urls": [
    "http:\/\/localhost:7001\/fairShare"
],
"planPath": null,
"applicationType": "war",
"openSessionsCurrentCount": 0,
"sessionsOpenedTotalCount": 0,
"health": {"state": "ok"},
"deploymentPath": "\/deployments\/fairShare.war",
"ejbs": []
```

```
        }]
    }
```

OPTIONS Method

The OPTIONS method on this resource returns a template entity that has been pre-populated with default values.

If you want to deploy an application, call this method to get the template, fill in values, such as the path to the application, then use it in the POST method to deploy the application.

Roles

Administrator

Response Body

The response body returned includes an Application template entity that can be customized and then used to deploy the application.

Response Codes

This method returns one of the Standard HTTP Status Codes.

POST Method

The resource supports the following methods:

- POST Accepting "application/json"
- POST Accepting "multipart/form-data"

POST Accepting "application/json"

This POST method deploys a new application when the files you want to deploy are already on the Administration Server file system.

Roles

Administrator

Request Query Parameters

This resource launches a job to deploy the application.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Request Body

The request body must include a fully populated Application entity that describes the new application. Read only parameter values are ignored.

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a DeploymentJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/deployment/{job-id} **rel**=job **title**=id

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 201 Created status code and a Location header with a link to the new application resource. If the job timed out, the method returns a 202 Accepted status code and a Location header with a link to the expected new application resource.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code, a link to the job resource, and a Location header with a link to the expected new application resource. In this case, the response does not include a DeploymentJob entity.

Example

Example 1 Deploying an Application

This example uses the POST method to deploy a new application.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:application/json \
-d "{
    name: 'BasicApp',
    deploymentPath: '/deployments/BasicApp/app/BasicApp.ear',
    targets: [ 'myserver' ]
}" \
-X POST http://localhost:7001/management/wls/latest/deployments/application
```

Example Response

```
HTTP/1.1 201 Created

Location:
http://localhost:7001/management/wls/latest/deployments/application/id/BasicApp

Response Body:
{
    "messages": [{
        "message": "Deployed the application 'BasicApp'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/0"
    }],
    "item": {
        "operation": "deploy",
        "status": "completed",
        "beginTime": 1390587114114,
        "endTime": 1390587115796,
        "name": "ADTR-0",
        "id": "0",
        "type": "deployment",
        "targets": [{
            "errors": [],
            "status": "completed",
            "name": "myserver",
            "type": "server"
        }],
        "deploymentName": "BasicApp",
        "description": "[Deployer:149026]deploy application BasicApp on myserver."
    }
}
```

Example 2 Deploying an Application Asynchronously

This example uses the POST method to deploy a new application asynchronously.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:application/json \
-d "{
    name: 'fairShare',
    deploymentPath: '/deployments/fairShare.war'
}" \
-X POST http://localhost:7001/management/wls/latest/deployments/application?__
detached=true
```

Example Response

```
 HTTP/1.1 202 Accepted

Location:
http://localhost:7001/management/wls/latest/deployments/application/id/fairShare

Response Body:
{
    "messages": [{
        "message": "Deploying the application 'fairShare'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/31"
    }]
}
```

POST Accepting "multipart/form-data"

This POST method deploys a new application when the files you want to deploy are not on the Administration Server file system. It uploads the files to deploy from the client's file system to the Administration Server file system and then deploys them.

Note: When using a multi-part form to create, you need to:

1. Specify the file to deploy via the `deployment` form element.
2. Specify the plan to deploy via the `plan` form element.
3. Send in the rest of the template via the `model` form element.

Roles

Administrator

Request Query Parameters

This resource launches a job to upload and deploy the application.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Request Body

The request body includes form data with a Content-Type header of multipart/form-data and the following form fields:

model
(Required) Specifies all the necessary information about the application to deploy except for the deployment archive and its corresponding deployment plan. The Content-Disposition header must have its name parameter set to `model`. The Content-Type header must be `application/json` and the content must be a JSON object containing an Application entity, excluding the `deployment` and `plan` fields.

deployment
(Required) Specifies the deployment archive to be uploaded. The Content-Disposition header must have its name parameter set to `deployment` and must have a filename parameter set to the file name you want the uploaded deployment archive to have on the server. The Content-Type header must be `application/octet-stream` and the content must be the deployment archive data.

plan
(Optional) Specifies the deployment plan to be uploaded. The Content-Disposition header must have its name parameter set to `plan` and must have a filename parameter set to the file name you want the uploaded deployment plan to have on the server. The Content-Type header must be `application/octet-stream` and the content must be the deployment plan data.

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a DeploymentJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/deployment/{job-id} **rel**=job **title**=id

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 201 Created status code and a Location header with a link to the new library resource. If the job timed out, the method returns a 202 Accepted status code and a Location header with a link to the expected new library resource.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code, a link to the job resource, and a Location header with a link to the expected new library resource. In this case, the response does not include a DeploymentJob entity.

Example

Example 1 Uploading and Deploying an Application

This example uses the POST method to upload a deployment archive and deploy it as a new application.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:multipart/form-data \
-F "model={
    name: 'myapp',
    targets: [ 'myserver', 'Cluster-0' ]
}" \
-F "deployment=@/deployments/MyApp/app/MyApp.ear" \
-X POST http://localhost:7001/management/wls/latest/deployments/application
```

Example Response

```
HTTP/1.1 100 Continue
 HTTP/1.1 201 Created

Location:
http://localhost:7001/management/wls/latest/deployments/application/id/myapp

Response Body:
{
    "messages": [{
        "message": "Deployed the application 'myapp'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/27"
    }],
    "item": {
        "operation": "deploy",
        "status": "completed",
        "beginTime": 1390587191114,
        "endTime": 1390587192047,
        "name": "ADTR-27",
        "id": "27",
        "type": "deployment",
        "targets": [
            {
                "errors": [],
                "status": "completed",
                "name": "Cluster-0",
                "type": "cluster"
            },
            {
                "errors": [],
                "status": "completed",
                "name": "myserver",
                "type": "server"
            }
```

```
        ],
        "deploymentName": "myapp",
        "description": "[Deployer:149026]deploy application myapp on
Cluster-0,myserver."
    }
}
```

/management/wls/{version}/deployments/application/id/{application-name}

This resource manages an application.

The resource supports the following methods:

- DELETE Method
- GET Method

DELETE Method

The DELETE method undeploys the application identified by the resource URL.

Roles

Administrator

Request Query Parameters

This resource launches a job to undeploy the application.

_detached
(optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a DeploymentJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/deployment/{job-id} **rel**=job **title**=id

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 200 OK status code. If the job timed out, the method returns a 202 Accepted status code.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code and a link to the job resource. In this case, the response does not include a DeploymentJob entity.

Example

Example 1 Undeploying an Application

This example uses the DELETE method to undeploy an application.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X DELETE
http://localhost:7001/management/wls/latest/deployments/application/id/BasicApp
```

Example Response

```
HTTP/1.1 200 OK
```

```
Response Body:
{
    "messages": [{
        "message": "Undeployed the application 'BasicApp'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/2"
    }],
    "item": {
        "operation": "remove",
        "status": "completed",
        "beginTime": 1390587119309,
        "endTime": 1390587119588,
        "name": "ADTR-2",
        "id": "2",
        "type": "deployment",
        "targets": [{
            "errors": [],
            "status": "completed",
            "name": "myserver",
            "type": "server"
        }],
        "deploymentName": "BasicApp",
        "description": "[Deployer:149026]remove application BasicApp on myserver."
    }
}
```

Example 2 Undeploying an Application Asynchronously

This example uses the DELETE method to undeploy an application asynchronously.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X DELETE
http://localhost:7001/management/wls/latest/deployments/application/id/fairShare?_
_detached=true
```

Example Response

```
HTTP/1.1 202 Accepted

Response Body:
{
    "messages": [{
        "message": "Undeploying the application 'fairShare'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/35"
    }]
}
```

GET Method

The GET method on this resource returns information about the application identified by the resource URL.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes an Application entity that contains information about the specified application.

This method can return the following links:

- **uri**=/management/wls/{version}/deployments/application/id/{application-name}/bindables **rel**=bindables
- **uri**=/management/wls/{version}/deployments/application **rel**=parent
- **uri**=/management/wls/{version}/deployments/application/id/{application-name}/redeploy **rel**=action **title**=redeploy
- **uri**=/management/wls/{version}/deployments/application/id/{application-name}/update **rel**=action **title**=update
- **uri**=/management/wls/{version}/deployments/application/id/{application-name}/start **rel**=action **title**=start
- **uri**=/management/wls/{version}/deployments/application/id/{application-name}/stop **rel**=action **title**=stop

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing an Application

This example uses the GET method to display information about a specific application.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET
http://localhost:7001/management/wls/latest/deployments/application/id/fairShare
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "parent",
            "uri":
```

```
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/application"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/application\/id\/f
airShare\/redeploy",
            "title": "redeploy"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/application\/id\/f
airShare\/update",
            "title": "update"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/application\/id\/f
airShare\/start",
            "title": "start"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/application\/id\/f
airShare\/stop",
            "title": "stop"
        },
        {
            "rel": "bindables",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/application\/id\/f
airShare\/bindables"
        }
    ],
    "item": {
        "name": "fairShare",
        "state": "active",
        "type": "application",
        "targets": [
            "myserver",
            "Cluster-0"
        ],
        "servlets": [
            {
                "servletName": "SimpleSlowServlet",
                "contextPath": "\/fairShare",
                "aggregateMetrics": {
                    "executionTimeTotal": 0,
                    "invocationTotalCount": 0,
                    "reloadTotalCount": 0,
                    "executionTimeHigh": 0,
                    "executionTimeLow": 0
                },
                "servletMetrics": [
                    {
                        "serverName": "myserver",
                        "executionTimeTotal": 0,
```

```
            "invocationTotalCount": 0,
            "reloadTotalCount": 0,
            "executionTimeHigh": 0,
            "executionTimeLow": 0
        }
    ]
},
{
    "servletName": "SimpleFastServlet",
    "contextPath": "\/fairShare",
    "aggregateMetrics": {
        "executionTimeTotal": 0,
        "invocationTotalCount": 0,
        "reloadTotalCount": 0,
        "executionTimeHigh": 0,
        "executionTimeLow": 0
    },
    "servletMetrics": [
        {
            "serverName": "myserver",
            "executionTimeTotal": 0,
            "invocationTotalCount": 0,
            "reloadTotalCount": 0,
            "executionTimeHigh": 0,
            "executionTimeLow": 0
        }
    ]
},
{
    "servletName": "JspServlet",
    "contextPath": "\/fairShare",
    "aggregateMetrics": {
        "executionTimeTotal": 0,
        "invocationTotalCount": 0,
        "reloadTotalCount": 0,
        "executionTimeHigh": 0,
        "executionTimeLow": 0
    },
    "servletMetrics": [
        {
            "serverName": "myserver",
            "executionTimeTotal": 0,
            "invocationTotalCount": 0,
            "reloadTotalCount": 0,
            "executionTimeHigh": 0,
            "executionTimeLow": 0
        }
    ]
},
{
    "servletName": "FileServlet",
    "contextPath": "\/fairShare",
    "aggregateMetrics": {
        "executionTimeTotal": 0,
        "invocationTotalCount": 0,
        "reloadTotalCount": 0,
        "executionTimeHigh": 0,
        "executionTimeLow": 0
    },
    "servletMetrics": [
```

```
                        {
                            "serverName": "myserver",
                            "executionTimeTotal": 0,
                            "invocationTotalCount": 0,
                            "reloadTotalCount": 0,
                            "executionTimeHigh": 0,
                            "executionTimeLow": 0
                        }
                    ]
                }
            ],
            "displayName": "fairShare",
            "urls": [
                "http:\/\/localhost:7001\/fairShare"
            ],
            "planPath": null,
            "applicationType": "war",
            "openSessionsCurrentCount": 0,
            "sessionsOpenedTotalCount": 0,
            "health": {"state": "ok"},
            "deploymentPath": "\/deployments\/fairShare.war",
            "ejbs": []
        }
    }
```

/management/wls/{version}/deployments/application/id/{application-name}/bindables

This resource manages the external resource references for an application.

The resource supports the following methods:

- GET Method
- POST Method

GET Method

The GET method on this resource returns a listing of the external resource references for this application.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes an ApplicationBindables entity that contains information about external resource references for the specified application.

This method can return the following links:

- **uri**=/management/wls/{version}/deployments/application/id/{application-name}
  rel=parent

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing the Application Bindables

This example uses the GET request to display the bindables for an application.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET
http://localhost:7001/management/wls/latest/deployments/application/id/MyApp/binda
bles
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [{
        "rel": "parent",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/application\/id\/M
yApp"
    }],
    "item": {
        "writeable": true,
        "bindables": [
            {
                "name": "JNDIName",
                "jndiName": "gddsBean1",
                "path": [
                    {
                        "name": "ApplicationDescriptor",
```

```
                "type": "DescriptorMBean"
            },
            {
                "name": "gddsBean2",
                "type": "ResourceDescriptionBean"
            }
        ]
    },
    {
        "name": "JNDIName",
        "jndiName": "gddsBean1",
        "path": [
            {
                "name": "ApplicationDescriptor",
                "type": "DescriptorMBean"
            },
            {
                "name": "gddsBean3",
                "type": "ResourceEnvDescriptionBean"
            }
        ]
    },
    {
        "name": "JNDIName",
        "type": "ejb",
        "jndiName": "gddsBean1",
        "path": [
            {
                "name": "ApplicationDescriptor",
                "type": "DescriptorMBean"
            },
            {
                "name": "gddsBean1",
                "type": "EjbReferenceDescriptionBean"
            }
        ]
    },
    {
        "name": "JNDIName",
        "jndiName": "gddsBean",
        "path": [
            {
                "name": "mywar.war",
                "type": "AppDeploymentConfigurationModuleMBean"
            },
            {
                "name": "WebAppDescriptor",
                "type": "DescriptorMBean"
            },
            {
                "name": "gddsBean2",
                "type": "ResourceDescriptionBean"
            }
        ]
    },
    {
        "name": "JNDIName",
        "jndiName": "gddsBean",
        "path": [
            {
```

```
                            "name": "mywar.war",
                            "type": "AppDeploymentConfigurationModuleMBean"
                        },
                        {
                            "name": "WebAppDescriptor",
                            "type": "DescriptorMBean"
                        },
                        {
                            "name": "gddsBean3",
                            "type": "ResourceEnvDescriptionBean"
                        }
                    ]
                },
                {
                    "name": "JNDIName",
                    "type": "ejb",
                    "jndiName": "gddsBean",
                    "path": [
                        {
                            "name": "mywar.war",
                            "type": "AppDeploymentConfigurationModuleMBean"
                        },
                        {
                            "name": "WebAppDescriptor",
                            "type": "DescriptorMBean"
                        },
                        {
                            "name": "gddsBean1",
                            "type": "EjbReferenceDescriptionBean"
                        }
                    ]
                }
            ]
        }
    }
```

POST Method

The POST method updates external resource references for the application. The application must already have a deployment plan before you can call this method.

Roles

Administrator, Deployer

Request Body

The request body must include a fully populated ApplicationBindables entity that describes the bindables for the application.

Response Body

The response body returned contains a message stating that the application bindables were updated.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Updating the Application Bindables

This example uses the POST method to update the bindables for an application.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:application/json \
-d "{
        'bindables': [
            {
                'jndiName': 'JNDINAME1',
                'name': 'JNDIName',
                'type': 'Ejb',
                'path': [
                    {
                        'name': 'ApplicationDescriptor',
                        'type': 'DescriptorMBean'
                    },
                    {
                        'name': 'gddsBean1',
                        'type': 'EjbReferenceDescriptionBean'
                    }
                ]
            },
            {
                'jndiName': 'JNDINAME2',
                'name': 'JNDIName',
                'type': 'JmsConnectionFactory',
                'path': [
                    {
```

```
                'name': 'ApplicationDescriptor',
                'type': 'DescriptorMBean'
            },
            {
                'name': 'gddsBean2',
                'type': 'ResourceDescriptionBean'
            }
        ]
    },
    {
        'jndiName': 'JNDINAME3',
        'name': 'JNDIName',
        'type': 'Resource Adapter',
        'path': [
            {
                'name': 'ApplicationDescriptor',
                'type': 'DescriptorMBean'
            },
            {
                'name': 'gddsBean3',
                'type': 'ResourceEnvDescriptionBean'
            }
        ]
    },
    {
        'jndiName': 'JNDINAME4',
        'name': 'JNDIName',
        'type': 'JmsConnectionFactory',
        'path': [
            {
                'name': 'mywar.war',
                'type': 'AppDeploymentConfigurationModuleMBean'
            },
            {
                'name': 'WebAppDescriptor',
                'type': 'DescriptorMBean'
            },
            {
                'type': 'WeblogicWebAppBean'
            },
            {
                'name': 'gddsBean2',
                'type': 'ResourceDescriptionBean'
            }
        ]
    },
    {
        'jndiName': 'JNDINAME5',
        'name': 'JNDIName',
        'type': 'Resource Adapter',
        'path': [
            {
                'name': 'mywar.war',
                'type': 'AppDeploymentConfigurationModuleMBean'
            },
            {
                'name': 'WebAppDescriptor',
                'type': 'DescriptorMBean'
            },
            {
```

```
                        'type': 'WeblogicWebAppBean'
                    },
                    {
                        'name': 'gddsBean3',
                        'type': 'ResourceEnvDescriptionBean'
                    }
                ]
            },
            {
                'jndiName': 'JNDINAME6',
                'name': 'JNDIName',
                'type': 'Ejb',
                'path': [
                    {
                        'name': 'mywar.war',
                        'type': 'AppDeploymentConfigurationModuleMBean'
                    },
                    {
                        'name': 'WebAppDescriptor',
                        'type': 'DescriptorMBean'
                    },
                    {
                        'type': 'WeblogicWebAppBean'
                    },
                    {
                        'name': 'gddsBean1',
                        'type': 'EjbReferenceDescriptionBean'
                    }
                ]
            }
        ]
}" \
-X POST
http://localhost:7001/management/wls/latest/deployments/application/id/MyApp/binda
bles
```

Example Response

```
HTTP/1.1 100 Continue
 HTTP/1.1 200 OK

Response Body:
{"messages": [{
    "message": "Successfully updated 'MyApp'.",
    "severity": "SUCCESS"
}]}
```

/management/wls/{version}/deployments/application/id/{application-name}/redeploy

This resource redeploys an application.

The resource supports the following methods:

- POST Method

POST Method

The resource supports the following methods:

- POST Accepting "application/json"
- POST Accepting "multipart/form-data"

POST Accepting "application/json"

This POST method redeploys the application identified by the resource URL. Use this method when the file to redeploy is already on the Administration Server file system. For example, you've physically replaced the file that the deployment refers to, and you want the deployment to start using the new file.

You should invoke this method after you have updated the underlying deployment archive or exploded directory.

Roles

Administrator, Deployer

Request Query Parameters

This resource launches a job to redeploy the application.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a DeploymentJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/deployment/{job-id} **rel**=job **title**=id

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 200 OK status code. If the job timed out, the method returns a 202 Accepted status code.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code and a link to the job resource. In this case, the response does not include a DeploymentJob entity.

Example

Example 1 Redeploying an Application

This example uses the POST method to redeploy the specified application.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/deployments/application/id/fairShare/r
edeploy
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "messages": [{
        "message": "Redeployed the application 'fairShare'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/25"
    }],
    "item": {
        "operation": "redeploy",
        "status": "completed",
        "beginTime": 1390587186018,
        "endTime": 1390587186120,
        "name": "ADTR-25",
        "id": "25",
        "type": "deployment",
        "targets": [
            {
                "errors": [],
                "status": "completed",
                "name": "Cluster-0",
                "type": "cluster"
            },
            {
                "errors": [],
                "status": "completed",
                "name": "myserver",
                "type": "server"
            }
        ],
        "deploymentName": "fairShare",
        "description": "[Deployer:149026]redeploy application fairShare on
Cluster-0,myserver."
    }
}
```

Example 2 Redeploying an Application Asynchronously

This example uses the POST method to redeploy the specified application asynchronously.

Example Request

```
curl -v \
```

```
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/deployments/application/id/fairShare/r
edeploy?__detached=true
```

Example Response

```
HTTP/1.1 202 Accepted

Response Body:
{
    "messages": [{
        "message": "Redeploying the application 'fairShare'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/34"
    }]
}
```

POST Accepting "multipart/form-data"

This POST method uploads and redeploys the application. Use this method when the file to redeploy exists on the client's file system. It uploads the file to the Administration Server and then signals the deployment to start using it.

Roles

Administrator

Request Query Parameters

This resource launches a job to upload and redeploy the application.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Request Body

The request body includes form data with a Content-Type header of multipart/form-data and the following form fields:

deployment
(Required) Specifies the application to redeploy. The Content-Disposition header must have its name parameter set to `deployment` and must have a filename parameter set to the file name you want the deployment archive to have on the server. The Content-Type header must be `application/octet-stream` and the content must be the deployment archive data.

plan
(Optional) Specifies the new deployment plan for the application. The Content-Disposition header must have its name parameter set to `plan` and must have a filename parameter set to the file name you want the deployment plan to have on the server. The Content-Type header must be `application/octet-stream` and the content must be the deployment plan data.

archiveVersion
(Optional) Specifies the archive version, as a String. The Content-Disposition header must have its name parameter set to `archiveVersion`. The Content-Type header must be `text/plain`.

planVersion
(Optional) Specifies the plan version, as a String. The Content-Disposition header must have its name parameter set to `planVersion`. The Content-Type header must be `text/plain`.

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a DeploymentJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/deployment/{job-id} **rel**=job **title**=id

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 200 OK status code. If the job timed out, the method returns a 202 Accepted status code.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code and a link to the job resource. In this case, the response does not include a DeploymentJob entity.

Example

Example 1 Uploading and Redeploying an Application

This example uses the POST method to upload an archive and redeploy it as an application.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:multipart/form-data \
-F "deployment=@/deployments/fairShare.war" \
-X POST
http://localhost:7001/management/wls/latest/deployments/application/id/myapp/redep
loy
```

Example Response

```
 HTTP/1.1 200 OK
```

```
Response Body:
{
    "messages": [{
        "message": "Redeployed the application 'myapp'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/29"
    }],
    "item": {
        "operation": "deploy",
        "status": "completed",
        "beginTime": 1390587195811,
        "endTime": 1390587196066,
        "name": "ADTR-29",
        "id": "29",
        "type": "deployment",
        "targets": [
            {
                "errors": [],
                "status": "completed",
                "name": "Cluster-0",
                "type": "cluster"
            },
            {
                "errors": [],
                "status": "completed",
                "name": "myserver",
                "type": "server"
            }
        ],
        "deploymentName": "myapp",
        "description": "[Deployer:149026]deploy application myapp on
Cluster-0,myserver."
    }
}
```

Example 2 Uploading and Redeploying an Application Asynchronously

This example uses the POST method to upload an archive and redeploy it as an application asynchronously.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:multipart/form-data \
-F "deployment=@/deployments/fairShare.war" \
-X POST
http://localhost:7001/management/wls/latest/deployments/library/id/airline/redeplo
y?__detached=true
```

Example Response

```
HTTP/1.1 202 Accepted
```

```
Response Body:
{
    "messages": [{
        "message": "Redeploying the library 'airline'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/13"
    }]
}
```

/management/wls/{version}/deployments/application/id/{application-name}/start

This resource starts an application.

The resource supports the following methods:

- POST Method

POST Method

The POST method starts the application identified by the resource URL.

Roles

Administrator, Deployer

Request Query Parameters

This resource launches a job to start the application.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a DeploymentJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/deployment/{job-id} **rel**=job **title**=id

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 200 OK status code. If the job timed out, the method returns a 202 Accepted status code.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code and a link to the job resource. In this case, the response does not include a DeploymentJob entity.

Example

Example 1 Starting an Application Synchronously

This example uses the POST method to start an application.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/deployments/application/id/fairShare/s
tart
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "messages": [{
        "message": "Started the application 'fairShare'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/24"
    }],
    "item": {
        "operation": "start",
        "status": "completed",
        "beginTime": 1390587183386,
        "endTime": 1390587183443,
        "name": "ADTR-24",
        "id": "24",
        "type": "deployment",
        "targets": [
            {
                "errors": [],
                "status": "completed",
                "name": "Cluster-0",
                "type": "cluster"
            },
            {
                "errors": [],
                "status": "completed",
                "name": "myserver",
                "type": "server"
            }
        ],
        "deploymentName": "fairShare",
        "description": "[Deployer:149026]start application fairShare on
Cluster-0,myserver."
    }
}
```

Example 2 Starting an Application Asynchronously

This example uses the POST method to start an application asynchronously.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/deployments/application/id/fairShare/s
tart?__detached=true
```

Example Response

```
HTTP/1.1 202 Accepted

Response Body:
{
```

```
    "messages": [{
        "message": "Starting the application 'fairShare'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/33"
    }]
}
```

/management/wls/{version}/deployments/application/id/{application-name}/stop

This resource stops an application.

The resource supports the following methods:

- POST Method

POST Method

The POST method stops the application identified by the resource URL.

Roles

Administrator, Deployer

Request Query Parameters

This resource launches a job to stop the application.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a DeploymentJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/deployment/{job-id} **rel**=job **title**=id

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 200 OK status code. If the job timed out, the method returns a 202 Accepted status code.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code and a link to the job resource. In this case, the response does not include a DeploymentJob entity.

Example

Example 1 Stopping an Application

This example uses the POST method to stop an application.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/deployments/application/id/fairShare/stop
```

Example Response

```
HTTP/1.1 200 OK
```

```
Response Body:
{
    "messages": [{
        "message": "Stopped the application 'fairShare'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/23"
    }],
    "item": {
        "operation": "stop",
        "status": "completed",
        "beginTime": 1390587180998,
        "endTime": 1390587181033,
        "name": "ADTR-23",
        "id": "23",
        "type": "deployment",
        "targets": [
            {
                "errors": [],
                "status": "completed",
                "name": "Cluster-0",
                "type": "cluster"
            },
            {
                "errors": [],
                "status": "completed",
                "name": "myserver",
                "type": "server"
            }
        ],
        "deploymentName": "fairShare",
        "description": "[Deployer:149026]stop application fairShare on
Cluster-0,myserver."
    }
}
```

Example 2 Stopping an Application Asynchronously

This example uses the POST method to stop an application asynchronously.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/deployments/application/id/fairShare/s
top?__detached=true
```

Example Response

```
HTTP/1.1 202 Accepted

Response Body:
{
    "messages": [{
```

```
            "message": "Stopping the application 'fairShare'.",
            "severity": "SUCCESS"
        }],
        "links": [{
            "rel": "job",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/32"
        }]
}
```

/management/wls/{version}/deployments/application/id/{application-name}/update

This resource updates an application. Use this resource when you want to update the dynamic elements in a deployment plan without redeploying the application; no sessions will be lost. Use the redeploy resource to update the actual deployment. It will also update all changes in a deployment plan, but sessions will be lost.

The resource supports the following methods:

- POST Method

POST Method

The POST method updates the application identified by the resource URL. You should invoke this method after you have updated the underlying deployment plan.

Roles

Administrator, Deployer

Request Query Parameters

This resource launches a job to update the application.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a DeploymentJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/deployment/{job-id} **rel**=job **title**=id

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 200 OK status code. If the job timed out, the method returns a 202 Accepted status code.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code and a link to the job resource. In this case, the response does not include a DeploymentJob entity.

Example

Example 1 Updating an Application

This example uses the POST method to update an application with deployment plan changes.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/deployments/application/id/BasicApp/update
```

Example Response

```
HTTP/1.1 400 Bad Request

Response Body:
{
    "messages": [{
        "message": "Failed to update the application 'BasicApp'.",
        "severity": "FAILURE"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/37"
    }],
    "item": {
        "operation": "update",
        "error": "\n The application BasicApp cannot have the resource
\/deployments\/BasicApp\/plan\/Plan.xml updated dynamically. Either:\n1.) The
resource does not exist. \n or \n2) The resource cannot be changed dynamically.
\nPlease ensure the resource uri is correct, and redeploy the entire application
for this change to take effect.",
        "status": "failed",
        "beginTime": 1390587229032,
        "endTime": 1390587229322,
        "name": "ADTR-37",
        "id": "37",
        "type": "deployment",
        "targets": [{
            "errors": ["\n The application BasicApp cannot have the resource
\/deployments\/BasicApp\/plan\/Plan.xml updated dynamically. Either:\n1.) The
resource does not exist. \n or \n2) The resource cannot be changed dynamically.
\nPlease ensure the resource uri is correct, and redeploy the entire application
for this change to take effect."],
            "status": "failed",
            "name": "myserver",
            "type": "server"
        }],
        "deploymentName": "BasicApp",
        "description": "[Deployer:149026]update application BasicApp on myserver."
    }
}
```

/management/wls/{version}/deployments/inspect

This resource inspects a deployment (archive or exploded directory) and returns information about it. This is useful for finding out information about the deployment before it has been deployed, for example, whether it's a library or an application.

The resource supports the following methods:

- OPTIONS Method
- POST Method

OPTIONS Method

The OPTIONS method on this resource returns a template entity that has been pre-populated with default values.

If you want to inspect a deployment, call this method to get the template, fill in the values, such as the deploymentPath, then use it in the POST method to inspect the deployment.

Roles

Administrator, Deployer

Response Body

The response body returned includes a DeploymentReference template entity that can be customized and then used to inspect the deployment.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Getting a Template Deployment Reference Entity

This example uses the OPTIONS method to obtain a template for deployment inspection.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X OPTIONS http://localhost:7001/management/wls/latest/deployments/inspect
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"item": {
    "planPath": null,
    "deploymentPath": null
}}
```

POST Method

The POST method on this resource examines a deployment and returns information about it.

Roles

Administrator

Request Body

The request body must include a fully populated DeploymentReference entity. Read only parameter values are ignored.

Response Body

The response body returned includes a DeploymentDescription entity.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Inspecting a Versioned Library

This example uses the POST method to inspect a library prior to deployment.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:application/json \
-d "{
    deploymentPath: '/deployments/aquarium_sv2_ivfab.ear'
}" \
-X POST http://localhost:7001/management/wls/latest/deployments/inspect
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [{
        "rel": "parent",
        "uri": "http:\/\/localhost:7001\/management\/wls\/latest\/deployments"
    }],
    "item": {
        "deploymentType": "library",
        "name": "aquarium_sv2_ivfab",
        "specVersion": "2.0",
        "implVersion": "fabulous",
        "deploymentPath": "\/deployments\/aquarium_sv2_ivfab.ear"
    }
}
```

/management/wls/{version}/deployments/library

This resource manages the libraries that are deployed in this WLS domain.

The resource supports the following methods:

- GET Method
- OPTIONS Method
- POST Method

GET Method

The GET method on this resource returns a listing of all the libraries in this WLS domain.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a collection of Library entities. It also contains links to their corresponding resources.

This method can return the following links:

- **uri**=/management/wls/{version}/deployments **rel**=parent
- **uri**=/management/wls/{version}/deployments/library/id/{library-name} **rel**=items.name **title**=name

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing Libraries

This example uses the GET method to return a summary of all libraries.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/deployments/library
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "parent",
            "uri": "http:\/\/localhost:7001\/management\/wls\/latest\/deployments"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/library\/id\/airli
ne",
            "title": "airline"
        }
    ],
    "items": [{
        "name": "airline",
```

```
            "state": "active",
            "type": "library",
            "targets": ["myserver"],
            "displayName": "airline",
            "deploymentPath": "\/deployments\/airline.war"
        }]
}
```

OPTIONS Method

The OPTIONS method on this resource returns a template entity that has been pre-populated with default values.

If you want to deploy a library, call this method to get the template, fill in values, such as the path to the library, then use it in the POST method to deploy the library.

Roles

Administrator

Response Body

The response body returned includes a Library template entity that can be customized and then used to deploy the library.

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Getting a Template Library Entity

This example uses the OPTIONS method to obtain a template for a library deployment.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X OPTIONS http://localhost:7001/management/wls/latest/deployments/library
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{"item": {
    "name": null,
    "state": null,
    "type": "library",
    "targets": null,
    "displayName": null,
    "specVersion": null,
    "implVersion": null,
    "deploymentPath": null,
    "referencingApplications": null,
    "referencingLibraries": null
}}
```

POST Method

The resource supports the following methods:

- POST Accepting "application/json"
- POST Accepting "multipart/form-data"

POST Accepting "application/json"

This POST method deploys a new library.

Roles

Administrator

Request Query Parameters

This resource launches a job to deploy the library.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Request Body

The request body must include a fully populated Library entity that describes the new library. Read only parameter values are ignored.

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a DeploymentJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/deployment/{job-id} **rel**=job **title**=name

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 201 Created status code and a Location header with a link to the new library resource. If the job timed out, the method returns a 202 Accepted status code and a Location header with a link to the expected new library resource.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code, a link to the job resource, and a Location header with a link to the expected new library resource. In this case, the response does not include a DeploymentJob entity.

Example

Example 1 Deploying a Library

This example uses the POST method to deploy a new library.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:application/json \
-d "{
    name: 'airline',
    deploymentPath: '/deployments/airline.war',
    targets: [ 'myserver', 'Cluster-0' ]
}" \
-X POST http://localhost:7001/management/wls/latest/deployments/library
```

Example Response

```
HTTP/1.1 201 Created

Location:
http://localhost:7001/management/wls/latest/deployments/library/id/airline

Response Body:
{
    "messages": [{
        "message": "Deployed the library 'airline'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/1"
    }],
    "item": {
        "operation": "deploy",
        "status": "completed",
        "beginTime": 1390587116296,
        "endTime": 1390587116676,
        "name": "ADTR-1",
        "id": "1",
        "type": "deployment",
        "targets": [
            {
                "errors": [],
                "status": "completed",
                "name": "Cluster-0",
                "type": "cluster"
            },
            {
                "errors": [],
                "status": "completed",
                "name": "myserver",
                "type": "server"
            }
        ],
        "deploymentName": "airline",
```

```
        "description": "[Deployer:149117]deploy library airline on
Cluster-0,myserver."
    }
}
```

Example 2 Deploying an Library Asynchronously

This example uses the POST method to deploy a new library asynchronously.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:application/json \
-d "{
    name: 'airline',
    deploymentPath: '/deployments/airline.war',
    targets: [ 'myserver', 'Cluster-0' ]
}" \
-X POST http://localhost:7001/management/wls/latest/deployments/library
```

Example Response

```
HTTP/1.1 201 Created

Location:
http://localhost:7001/management/wls/latest/deployments/library/id/airline

Response Body:
{
    "messages": [{
        "message": "Deployed the library 'airline'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/1"
    }],
    "item": {
        "operation": "deploy",
        "status": "completed",
        "beginTime": 1390587116296,
        "endTime": 1390587116676,
        "name": "ADTR-1",
        "id": "1",
        "type": "deployment",
        "targets": [
            {
                "errors": [],
                "status": "completed",
                "name": "Cluster-0",
                "type": "cluster"
            },
            {
                "errors": [],
                "status": "completed",
                "name": "myserver",
                "type": "server"
```

```
            }
        ],
        "deploymentName": "airline",
        "description": "[Deployer:149117]deploy library airline on
Cluster-0,myserver."
    }
}
```

POST Accepting "multipart/form-data"

This POST method uploads and deploys a new library.

Note: When using a multi-part form to create, you need to:

1. Specify the file to deploy via the `deployment` form element.
2. Send in the rest of the template via the `model` form element.

Roles

Administrator

Request Query Parameters

This resource launches a job to upload and deploy the library.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Request Body

The request body includes form data with a Content-Type header of multipart/form-data and the following form fields:

model
(Required) Specifies all the necessary information about the library to deploy. The Content-Disposition header must have its name parameter set to `model`. The Content-Type header must be `application/json` and the content must be a JSON object containing a Library entity excluding the `deployment` field.

deployment
(Required) Specifies the library to be deployed. The Content-Disposition header must have its name parameter set to `deployment` and must have a filename parameter set to the file name you want the library to have on the server. The Content-Type header must be `application/octet-stream` and the content must be the library data.

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a DeploymentJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/deployment/{job-id} **rel**=job **title**=id

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 201 Created status code and a Location header with a link to the new library resource. If the job timed out, the method returns a 202 Accepted status code and a Location header with a link to the expected new library resource.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code, a link to the job resource, and a Location header with a link to the expected new library resource. In this case, the response does not include a DeploymentJob entity.

Example

Example 1 Uploading and Deploying a Library

This example uses the POST method to upload a deployment archive and deploy it as a new library.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:multipart/form-data \
-F "model={
    name: 'airline',
    targets: [ 'myserver' ]
}" \
-F "deployment=@/deployments/airline.war" \
-X POST http://localhost:7001/management/wls/latest/deployments/library
```

Example Response

```
 HTTP/1.1 201 Created

Location:
http://localhost:7001/management/wls/latest/deployments/library/id/airline

Response Body:
{
    "messages": [{
        "message": "Deployed the library 'airline'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/7"
    }],
    "item": {
        "operation": "deploy",
        "status": "completed",
        "beginTime": 1390587130807,
        "endTime": 1390587131056,
        "name": "ADTR-7",
        "id": "7",
        "type": "deployment",
```

```
        "targets": [{
            "errors": [],
            "status": "completed",
            "name": "myserver",
            "type": "server"
        }],
        "deploymentName": "airline",
        "description": "[Deployer:149117]deploy library airline on myserver."
    }
}
```

Example 2 Uploading and Deploying a Library Asynchronously

This example uses the POST method to upload a deployment archive and deploy it as a new library.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:multipart/form-data \
-F "model={
    name: 'airline',
    targets: [ 'myserver' ]
}" \
-F "deployment=@/deployments/airline.war" \
-X POST http://localhost:7001/management/wls/latest/deployments/library?__
detached=true
```

Example Response

```
HTTP/1.1 202 Accepted

Location:
http://localhost:7001/management/wls/latest/deployments/library/id/airline

Response Body:
{
    "messages": [{
        "message": "Deploying the library 'airline'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/11"
    }]
}
```

/management/wls/{version}/deployments/library/id/{library-name}

This resource manages a library.

The resource supports the following methods:

- DELETE Method
- GET Method

DELETE Method

The DELETE method undeploys the library identified by the resource URL.

Roles

Administrator

Request Query Parameters

This resource launches a job to undeploy the library.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a DeploymentJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/deployment/{job-id} **rel**=job **title**=id

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 200 OK status code. If the job timed out, the method returns a 202 Accepted status code.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code and a link to the job resource. In this case, the response does not include a DeploymentJob entity.

Example

Example 1 Undeploying a Library

This example uses the DELETE method to undeploy a library.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X DELETE
http://localhost:7001/management/wls/latest/deployments/library/id/airline
```

Example Response

```
HTTP/1.1 200 OK
```

```
Response Body:
{
    "messages": [{
        "message": "Undeployed the library 'airline'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/3"
    }],
    "item": {
        "operation": "remove",
        "status": "completed",
        "beginTime": 1390587121460,
        "endTime": 1390587121632,
        "name": "ADTR-3",
        "id": "3",
        "type": "deployment",
        "targets": [
            {
                "errors": [],
                "status": "completed",
                "name": "Cluster-0",
                "type": "cluster"
            },
            {
                "errors": [],
                "status": "completed",
                "name": "myserver",
                "type": "server"
            }
        ],
        "deploymentName": "airline",
        "description": "[Deployer:149117]remove library airline on
Cluster-0,myserver."
    }
}
```

GET Method

The GET method on this resource returns information about the library identified by the resource URL.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a Library entity that contains information about the specified library.

This method can return the following links:

- **uri**=/management/wls/{version}/deployments/library/id/{library-name}/undeploy **rel**=undeploy
- **uri**=/management/wls/{version}/deployments/library **rel**=parent

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing a Library

This example uses the GET method to find information about a specific library.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/deployments/library/id/airline
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "parent",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/library"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/deployments\/library\/id\/airli
ne\/redeploy",
            "title": "redeploy"
        }
    ],
    "item": {
```

```
        "name": "airline",
        "state": "active",
        "type": "library",
        "targets": ["myserver"],
        "displayName": "airline",
        "deploymentPath": "\/deployments\/airline.war"
    }
}
```

/management/wls/{version}/deployments/library/id/{library-name}/redeploy

This resource redeploys a library.

The resource supports the following methods:

- POST Method

POST Method

The resource supports the following methods:

- POST Accepting "application/json"
- POST Accepting "multipart/form-data"

POST Accepting "application/json"

This POST method redeploys the library identified by the resource URL. You should invoke this method after you have updated the underlying deployment archive or exploded directory.

Roles

Administrator, Deployer

Request Query Parameters

This resource launches a job to redeploy the library.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a DeploymentJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/deployment/{job-id} **rel**=job **title**=id

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 200 OK status code. If the job timed out, the method returns a 202 Accepted status code.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code and a link to the job resource. In this case, the response does not include a DeploymentJob entity.

Example

Example 1 Redeploying a Library

This example uses the POST method to redeploy the specified library.

Example Request

```
curl -v \
```

```
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/deployments/library/id/airline/redeplo
y
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "messages": [{
        "message": "Redeployed the library 'airline'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/5"
    }],
    "item": {
        "operation": "redeploy",
        "status": "completed",
        "beginTime": 1390587126314,
        "endTime": 1390587126328,
        "name": "ADTR-5",
        "id": "5",
        "type": "deployment",
        "targets": [{
            "errors": [],
            "status": "completed",
            "name": "myserver",
            "type": "server"
        }],
        "deploymentName": "airline",
        "description": "[Deployer:149117]redeploy library airline on myserver."
    }
}
```

POST Accepting "multipart/form-data"

This POST method uploads and redeploys the library identified by the resource URL. You should invoke this method after you have updated the underlying deployment archive or exploded directory.

Roles

Administrator, Deployer

Request Query Parameters

This resource launches a job to upload and redeploy the library.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Request Body

The request body includes form data with a Content-Type header of multipart/form-data and the following form field:

deployment
(Required) Specifies the library to be redeployed. The Content-Disposition header must have its name parameter set to `deployment` and must have a filename parameter set to the file name you want the library to have on the server. The Content-Type header must be `application/octet-stream` and the content must be the library data.

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a DeploymentJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/deployment/{job-id} **rel**=job **title**=id

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 200 OK status code. If the job timed out, the method returns a 202 Accepted status code.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code and a link to the job resource. In this case, the response does not include a DeploymentJob entity.

Example

Example 1 Uploading and Redeploying a Library

This example uses the POST method to upload an archive and redeploy it as a library.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:multipart/form-data \
-F "deployment=@/deployments/fairShare.war" \
-X POST
http://localhost:7001/management/wls/latest/deployments/library/id/airline/redeplo
y
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
```

```
    "messages": [{
        "message": "Redeployed the library 'airline'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/9"
    }],
    "item": {
        "operation": "deploy",
        "status": "completed",
        "beginTime": 1390587135085,
        "endTime": 1390587135333,
        "name": "ADTR-9",
        "id": "9",
        "type": "deployment",
        "targets": [{
            "errors": [],
            "status": "completed",
            "name": "myserver",
            "type": "server"
        }],
        "deploymentName": "airline",
        "description": "[Deployer:149117]deploy library airline on myserver."
    }
}
```

Example 2 Uploading and Redeploying a Library Asynchronously

This example uses the POST method to upload an archive and redeploy it as a library.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-H Content-Type:multipart/form-data \
-F "deployment=@/deployments/fairShare.war" \
-X POST
http://localhost:7001/management/wls/latest/deployments/library/id/airline/redeplo
y?__detached=true
```

Example Response

```
HTTP/1.1 202 Accepted

Response Body:
{
    "messages": [{
        "message": "Redeploying the library 'airline'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment\/id\/13"
    }]
}
```

/management/wls/{version}/jobs

This resource lists the jobs running in this WLS domain.

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns a listing of all the jobs in this WLS domain.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a collection of Job entities. It also contains links to their corresponding resources.

This method can return the following links:

- **uri**=/management/wls/{version}/jobs/server **rel**=server
- **uri**=/management/wls/{version}/jobs/deployment **rel**=deployment
- **uri**=/management/wls/{version} **rel**=parent
- **uri**=/management/wls/{version}/jobs/{job-type}/id/{job-id} **rel**=items.id **title**=id

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing All Jobs

This example uses the GET method to obtain information about all jobs.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/jobs
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "parent",
            "uri": "http:\/\/localhost:7001\/management\/wls\/latest"
        },
        {
            "rel": "server",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/server"
        },
        {
            "rel": "deployment",
            "uri":
```

```
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment"
        },
        {
            "rel": "items.id",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/server\/id\/Cluster-0-Ser
ver-1:_0_start",
            "title": "Cluster-0-Server-1:_0_start"
        }
    ],
    "items": [{
        "status": "completed",
        "beginTime": 1390586983126,
        "endTime": 1390586998781,
        "name": "_0_start",
        "id": "Cluster-0-Server-1:_0_start",
        "type": "server",
        "description": "Starting Cluster-0-Server-1 server ..."
    }]
}
```

/management/wls/{version}/jobs/deployment

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns information about each job that is related to a deployment operation.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a DeploymentJob entity.

This method can return the following links:

- **uri**=/management/wls/{version}/jobs **rel**=parent
- **uri**=/management/wls/{version}/jobs/deployment/id/{job-id} **rel**=deployment **title**=name

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing All Deployment Jobs

This example uses the GET method to obtain information about all deployment jobs.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/jobs/deployment
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [{
        "rel": "parent",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment"
    }],
    "items": [{
        "operation": "deploy",
        "status": "in progress",
        "beginTime": 1390587139546,
        "name": "ADTR-11",
        "id": "11",
        "type": "deployment",
        "targets": [{
            "errors": [],
            "status": "in progress",
            "name": "myserver",
            "type": "server"
```

```
            }],
            "deploymentName": "airline",
            "description": "[Deployer:149117]deploy library airline on myserver."
        }]
    }
```

/management/wls/{version}/jobs/deployment/id/{job-id}

This resource returns information about a deployment-related job.

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns information about the deployment-related job identified by the resource URL.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a DeploymentJob entity that contains information about the specified deployment-related job.

This method can return the following links:

- **uri**=/management/wls/{version}/jobs/deployment **rel**=parent

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing a Deployment-Related Job

This example uses the GET method to obtain information about a deployment job.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/jobs/deployment/id/11
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [{
        "rel": "parent",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/deployment"
    }],
    "item": {
        "operation": "deploy",
        "status": "in progress",
        "beginTime": 1390587139546,
        "name": "ADTR-11",
        "id": "11",
        "type": "deployment",
        "targets": [{
            "errors": [],
            "status": "in progress",
            "name": "myserver",
            "type": "server"
        }],
```

```
            "deploymentName": "airline",
            "description": "[Deployer:149117]deploy library airline on myserver."
        }
    }
```

/management/wls/{version}/jobs/server

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns information about each job that is related to a server lifecycle operation.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a ServerJob entity.

This method can return the following links:

- **uri**=/management/wls/{version}/jobs **rel**=parent
- **uri**=/management/wls/{version}/jobs/server/id/{job-id} **rel**=server **title**=name

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Getting All Server Jobs

This example uses the GET method to obtain information about all server lifecycle jobs.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/jobs/server
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "parent",
            "uri": "http:\/\/localhost:7001\/management\/wls\/latest\/jobs"
        },
        {
            "rel": "server",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/server\/id\/Cluster-0-Ser
ver-1:_0_start",
            "title": "Cluster-0-Server-1:_0_start"
        }
    ],
    "items": [{
        "operation": "start",
        "status": "completed",
        "beginTime": 1390586983126,
```

```
            "endTime": 1390586998781,
            "name": "_0_start",
            "id": "Cluster-0-Server-1:_0_start",
            "type": "server",
            "description": "Starting Cluster-0-Server-1 server ...",
            "serverName": "Cluster-0-Server-1"
        }]
    }
```

/management/wls/{version}/jobs/server/id/{job-id}

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns information about a specific server lifecycle job.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a ServerJob entity.

This method can return the following links:

- **uri**=/management/wls/{version}/jobs/server **rel**=parent

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Getting Server Job Information

This example uses the GET method to obtain information about a server lifecycle job.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET
http://localhost:7001/management/wls/latest/jobs/server/id/Cluster-0-Server-1:_8_
start
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [{
        "rel": "parent",
        "uri": "http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/server"
    }],
    "item": {
        "operation": "start",
        "status": "in progress",
        "beginTime": 1390587038967,
        "name": "_8_start",
        "id": "Cluster-0-Server-1:_8_start",
        "type": "server",
        "description": "Starting Cluster-0-Server-1 server ...",
        "serverName": "Cluster-0-Server-1"
    }
}
```

/management/wls/{version}/servers

This resource enumerates the servers in a WLS domain and provides summary metrics and state for each server.

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns a listing of this domain's servers.

Roles

Administrator, Deployer, Operator, Monitor

Request Query Parameters

This method supports optional query parameters which can be used to constrain the fields included in the response. Constraining the fields to only the information that you require may yield faster response times when interacting with large data sets.

?_excludeFields=*field1*[,*field2*, ...]
(Optional) Specifies that these values are not required and may be omitted in a response.

?_includeFields=*field1*[,*field2*, ...]
(Optional) Specifies that these values are required and must be included in the response.

Include and exclude are mutually exclusive, and if neither is specified all available information is included.

Response Body

The response body returned includes a collection of Server entities. It also contains links to their corresponding resources.

This method can return the following links:

- **uri**=/management/wls/{version} **rel**=parent
- **uri**=/management/wls/{version}/servers/id/{server-name} **rel**=items.name **title**=name

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing Servers

This example uses the GET method to return a list of all servers in this domain.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/servers
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
```

```
{
    "links": [
        {
            "rel": "parent",
            "uri": "http:\/\/localhost:7001\/management\/wls\/latest"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/servers\/id\/myserver",
            "title": "myserver"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/servers\/id\/Cluster-0-Server-1
",
            "title": "Cluster-0-Server-1"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/servers\/id\/Cluster-0-Server-2
",
            "title": "Cluster-0-Server-2"
        }
    ],
    "items": [
        {
            "name": "myserver",
            "state": "running",
            "heapFreeCurrent": 136785464,
            "heapSizeCurrent": 423231488,
            "activeHttpSessionCount": 0,
            "activeThreadCount": 5,
            "health": {"state": "ok"},
            "usedPhysicalMemory": 3502743552,
            "jvmProcessorLoad": 0
        },
        {
            "name": "Cluster-0-Server-1",
            "state": "shutdown"
        },
        {
            "name": "Cluster-0-Server-2",
            "state": "shutdown"
        }
    ]
}
```

/management/wls/{version}/servers/id/{server-name}

This resource gets information about a specific server.

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns information about the server identified by the resource URL.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a Server entity that contains information about the specified server.

This method can return the following links:

- **uri**=/management/wls/{version}/servers/id/{server-name}/logs **rel**=logs
- **uri**=/management/wls/{version}/servers **rel**=parent
- **uri**=/management/wls/{version}/servers/id/{server-name}/start **rel**=action **title**=start
- **uri**=/management/wls/{version}/servers/id/{server-name}/restart **rel**=action **title**=restart
- **uri**=/management/wls/{version}/servers/id/{server-name}/suspend **rel**=action **title**=suspend
- **uri**=/management/wls/{version}/servers/id/{server-name}/resume **rel**=action **title**=resume
- **uri**=/management/wls/{version}/servers/id/{server-name}/shutdown **rel**=action **title**=shutdown

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing a Specific Server

This example uses the GET method to obtain information about a specific server.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/servers/id/Cluster-0-Server-1
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "parent",
```

```
            "uri": "http:\/\/localhost:7001\/management\/wls\/latest\/servers"
        },
        {
            "rel": "logs",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/servers\/id\/Cluster-0-Server-1
\/logs"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/servers\/id\/Cluster-0-Server-1
\/start",
            "title": "start"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/servers\/id\/Cluster-0-Server-1
\/restart",
            "title": "restart"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/servers\/id\/Cluster-0-Server-1
\/suspend",
            "title": "suspend"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/servers\/id\/Cluster-0-Server-1
\/resume",
            "title": "resume"
        },
        {
            "rel": "action",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/servers\/id\/Cluster-0-Server-1
\/shutdown",
            "title": "shutdown"
        }
    ],
    "item": {
        "name": "Cluster-0-Server-1",
        "state": "running",
        "heapFreeCurrent": 231060176,
        "heapSizeCurrent": 320798720,
        "activeHttpSessionCount": 0,
        "activeThreadCount": 6,
        "health": {"state": "ok"},
        "usedPhysicalMemory": 3925577728,
        "jvmProcessorLoad": 0
    }
}
```

/management/wls/{version}/servers/id/{server-name}/logs

This resource returns a list of logs which are maintained for this server.

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns the server's log file list.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body returned includes a collection of LogReference entities. It also contains links to their corresponding resources.

This method can return the following links:

- **uri**=/management/wls/{version}/servers/id/{server-name} **rel**=parent
- **uri**=/management/wls/{version}/servers/id/{server-name}/logs/id/{log-name} **rel**=items.name **title**=name

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Displaying a Server's Logs

This example uses the GET method to find all of the available logs for a server.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET
http://localhost:7001/management/wls/latest/servers/id/Cluster-0-Server-1/logs
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [
        {
            "rel": "parent",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/servers\/id\/Cluster-0-Server-1
"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/servers\/id\/Cluster-0-Server-1
\/logs\/id\/HTTPAccessLog",
            "title": "HTTPAccessLog"
        },
        {
            "rel": "items.name",
```

```
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/servers\/id\/Cluster-0-Server-1
\/logs\/id\/DataSourceLog",
            "title": "DataSourceLog"
        },
        {
            "rel": "items.name",
            "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/servers\/id\/Cluster-0-Server-1
\/logs\/id\/ServerLog",
            "title": "ServerLog"
        }
    ],
    "items": [
        {"name": "HTTPAccessLog"},
        {"name": "DataSourceLog"},
        {"name": "ServerLog"}
    ]
}
```

/management/wls/{version}/servers/id/{server-name}/logs/id/{log-name}

This resource displays the entries in a server's log.

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns the server's log.

Roles

Administrator, Deployer, Operator, Monitor

Request Query Parameters

This method supports the following query parameters which can be used to filter and qualify the results included in the response.

maxResults=*long*
(Optional) Specifies the maximum number of entries to return. If maxResults is not specified, or is less than one, then all of the entries in the log are returned. Otherwise, only maxResults entries are returned. If there are more than maxResults matching entries, the last maxResults entries (based on the order of the entries in the underlying log) are returned.

startTime=*mm/dd/yy hh:mm:ss ms*
(Optional) Specifies the earliest log entry to return. Note that you need to encode the query parameter to turn it into a valid URL; a space becomes %20, a '/' (forward slash) becomes %2F, and a ':' (colon) becomes %3A. For example, `management/wls/latest/servers/id/myserver/logs/id/mylog?startTime=11/21/14 12:06:53` would become `management/wls/latest/servers/id/myserver/logs/id/mylog?startTime=11%2F21%2F14%2012%3A06%3A53`.

endTime=*mm/dd/yy hh:mm:ss ms*
(Optional) Specifies the latest log entry to return. Note that you need to encode the query parameter to turn it into a valid URL; a space becomes %20, a '/' (forward slash) becomes %2F, and a ':' (colon) becomes %3A. For example, `management/wls/latest/servers/id/myserver/logs/id/mylog?endTime=11/21/14 02:23:15` would become `management/wls/latest/servers/id/myserver/logs/id/mylog?endTime=11%2F21%2F14%2002%3A23%3A15`.

query=*wldf expression*
(Optional) Specifies a WebLogic Diagnostics Framework (WLDF) query expression which can be used to further filter and qualify the log entries returned.

Response Body

The response body returned includes a collection of LogEntry entities. The entities are in the same order as the corresponding entries in the underlying log.

This method can return the following links:

- **uri**=/management/wls/{version}/servers/id/{server-name}/logs **rel**=parent

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Displaying a Server's Log

This example uses the GET method to return the last five entries in the server log.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET
http://localhost:7001/management/wls/latest/servers/id/Cluster-0-Server-1/logs/id/
ServerLog?maxResults=5
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [{
        "rel": "parent",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/servers\/id\/Cluster-0-Server-1
\/logs"
    }],
    "items": [
        {
            "timeStamp": "01\/24\/14 13:09:58 604",
            "message": "The server \"myserver\" connected to this server.",
            "severity": "Info",
            "subSystem": "Server",
            "userId": "WLS Kernel",
            "contextId": "",
            "recordId": "90",
            "msgId": "BEA-002635"
        },
        {
            "timeStamp": "01\/24\/14 13:09:58 613",
            "message": "Starting \"async\" replication service with remote cluster
address \"null\"",
            "severity": "Notice",
            "subSystem": "Cluster",
            "userId": "WLS Kernel",
            "contextId": "",
            "recordId": "91",
            "msgId": "BEA-000162"
        },
        {
            "timeStamp": "01\/24\/14 13:09:58 670",
            "message": "The server started in RUNNING mode.",
            "severity": "Notice",
            "subSystem": "WebLogicServer",
            "userId": "WLS Kernel",
            "contextId": "",
            "recordId": "92",
            "msgId": "BEA-000360"
        },
        {
```

```
            "timeStamp": "01\/24\/14 13:09:59 507",
            "message": "Server state changed to RUNNING.",
            "severity": "Notice",
            "subSystem": "WebLogicServer",
            "userId": "WLS Kernel",
            "contextId": "",
            "recordId": "93",
            "msgId": "BEA-000365"
        },
        {
            "timeStamp": "01\/24\/14 13:09:59 512",
            "message": "Instantiated an instance of
org.hibernate.validator.engine.resolver.JPATraversableResolver.",
            "severity": "Info",
            "subSystem":
"org.hibernate.validator.engine.resolver.DefaultTraversableResolver",
            "userId": "WLS Kernel",
            "contextId": "",
            "recordId": "94",
            "msgId": "BEA-000000"
        }
    ]
}
```

/management/wls/{version}/servers/id/{server-name}/restart

This resource restarts a specific server.

The resource supports the following methods:

- POST Method

POST Method

The POST method restarts the server identified by the resource URL.

Roles

Administrator

Request Query Parameters

This resource launches a job to restart the server.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a ServerJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/server/{job-id} **rel**=job **title**=name

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 200 OK status code. If the job timed out, the method returns a 202 Accepted status code.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code and a link to the job resource. In this case, the response does not include a ServerJob entity.

Example

Example 1 Restarting a Server

This example uses the POST method to restart a server synchronously.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/servers/id/Cluster-0-Server-1/restart
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
```

```
{
    "messages": [{
        "message": "Restarted the server 'Cluster-0-Server-1'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/server\/id\/Cluster-0-Ser
ver-1:_4_start"
    }],
    "item": {
        "operation": "start",
        "status": "completed",
        "beginTime": 1390587009914,
        "endTime": 1390587022047,
        "name": "_4_start",
        "id": "Cluster-0-Server-1:_4_start",
        "type": "server",
        "description": "Starting Cluster-0-Server-1 server ...",
        "serverName": "Cluster-0-Server-1"
    }
}
```

Example 2 Restarting a Server Asynchronously

This example uses the POST method to restart a server in detached mode.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/servers/id/Cluster-0-Server-1/restart?
__detached=true
```

Example Response

```
HTTP/1.1 202 Accepted

Response Body:
{
    "messages": [{
        "message": "Restarting the server 'Cluster-0-Server-1'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/server\/id\/Cluster-0-Ser
ver-1:_12_start"
    }]
}
```

/management/wls/{version}/servers/id/{server-name}/resume

This resource resumes a server.

The resource supports the following methods:

- POST Method

POST Method

The POST method resumes the server instance that is suspended or in the ADMIN state.

Roles

Administrator

Request Query Parameters

This resource launches a job to restart the server.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a ServerJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/server/{job-id} **rel**=job **title**=name

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 200 OK status code. If the job timed out, the method returns a 202 Accepted status code.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code and a link to the job resource. In this case, the response does not include a ServerJob entity.

Example

Example 1 Resuming a Server

This example uses the POST method to resume a server synchronously.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/servers/id/Cluster-0-Server-1/resume
```

Example Response

```
HTTP/1.1 200 OK
```

```
Response Body:
{
    "messages": [{
        "message": "Resumed the server 'Cluster-0-Server-1'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/server\/id\/Cluster-0-Ser
ver-1:_2_resume"
    }],
    "item": {
        "operation": "resume",
        "status": "completed",
        "beginTime": 1390587004735,
        "endTime": 1390587004747,
        "name": "_2_resume",
        "id": "Cluster-0-Server-1:_2_resume",
        "type": "server",
        "description": "Resuming Cluster-0-Server-1 server ...",
        "serverName": "Cluster-0-Server-1"
    }
}
```

Example 2 Resuming a Server Asynchronously

This example uses the POST method to resume a server in detached mode.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/servers/id/Cluster-0-Server-1/resume?_
_detached=true
```

Example Response

```
HTTP/1.1 202 Accepted

Response Body:
{
    "messages": [{
        "message": "Resuming the server 'Cluster-0-Server-1'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/server\/id\/Cluster-0-Ser
ver-1:_10_resume"
    }]
}
```

/management/wls/{version}/servers/id/{server-name}/shutdown

This resource performs a graceful shut down on a specific server.

The resource supports the following methods:

- POST Method

POST Method

The POST method shuts down the server identified by the resource URL.

Roles

Administrator, Deployer, Operator

Request Query Parameters

This resource launches a job to shut down the server.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously. If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

force
(Optional) Specifies whether to gracefully shut down the server (false, the default) or force shut down of the server (true).

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a ServerJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/server/{job-id} **rel**=job **title**=name

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 200 OK status code. If the job timed out, the method returns a 202 Accepted status code.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code and a link to the job resource. In this case, the response does not include a ServerJob entity.

Example

Example 1 Shutting Down a Server

This example uses the POST method to shut down a server synchronously.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/servers/id/Cluster-0-Server-1/shutdown
```

Example Response

```
HTTP/1.1 200 OK
```

```
Response Body:
{
    "messages": [{
        "message": "Shutdown the server 'Cluster-0-Server-1'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/server\/id\/Cluster-0-Ser
ver-1:_7_shutdown"
    }],
    "item": {
        "operation": "shutdown",
        "status": "completed",
        "beginTime": 1390587036811,
        "endTime": 1390587036940,
        "name": "_7_shutdown",
        "id": "Cluster-0-Server-1:_7_shutdown",
        "type": "server",
        "description": "Shutting down Cluster-0-Server-1 server ...",
        "serverName": "Cluster-0-Server-1"
    }
}
```

Example 2 Shutting Down a Server Asynchronously

This example uses the POST method to shut down a server in detached mode.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/servers/id/Cluster-0-Server-1/shutdown
?__detached=true
```

Example Response

```
HTTP/1.1 202 Accepted

Response Body:
{
    "messages": [{
        "message": "Shutting down the server 'Cluster-0-Server-1'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/server\/id\/Cluster-0-Ser
ver-1:_11_shutdown"
    }]
}
```

/management/wls/{version}/servers/id/{server-name}/start

This resource starts a specific server.

The resource supports the following methods:

- POST Method

POST Method

The POST method starts the server identified by the resource URL.

Roles

Administrator

Request Query Parameters

This resource launches a job to start the server.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a ServerJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/server/{job-id} **rel**=job **title**=name

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 200 OK status code. If the job timed out, the method returns a 202 Accepted status code.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code and a link to the job resource. In this case, the response does not include a ServerJob entity.

Example

Example 1 Starting a Server

This example uses the POST method to start a server synchronously.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/servers/id/Cluster-0-Server-1/start
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
```

```
{
    "messages": [{
        "message": "Started the server 'Cluster-0-Server-1'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/server\/id\/Cluster-0-Ser
ver-1:_0_start"
    }],
    "item": {
        "operation": "start",
        "status": "completed",
        "beginTime": 1390586983126,
        "endTime": 1390586998781,
        "name": "_0_start",
        "id": "Cluster-0-Server-1:_0_start",
        "type": "server",
        "description": "Starting Cluster-0-Server-1 server ...",
        "serverName": "Cluster-0-Server-1"
    }
}
```

Example 2 Starting a Server Asynchronously

This example uses the POST method to start a server in detached mode.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/servers/id/Cluster-0-Server-1/start?__
detached=true
```

Example Response

```
HTTP/1.1 202 Accepted

Response Body:
{
    "messages": [{
        "message": "Starting the server 'Cluster-0-Server-1'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/server\/id\/Cluster-0-Ser
ver-1:_8_start"
    }]
}
```

/management/wls/{version}/servers/id/{server-name}/suspend

This resource suspends a server.

The resource supports the following methods:

- POST Method

POST Method

The POST method moves the server identified by the resource URL from the RUNNING state to the ADMIN state.

Roles

Administrator

Request Query Parameters

This resource launches a job to suspend the server.

_detached
(Optional) Specifies whether this method runs in detached (background) mode or synchronously.

If the __detached=true query parameter is specified (detached invocation), then this method returns immediately after launching the job. Otherwise, it waits for the job to complete, fail, or timeout (synchronous invocation).

Response Body

If the method was invoked synchronously and successfully launched the job, the response body includes a ServerJob entity containing the state of the job (completed, failed, or still running in the case of a timeout) as well as a link to the job resource.

This method can return the following links:

- **uri**=/wls/jobs/server/{job-id} **rel**=job **title**=name

Response Codes

If the method was invoked synchronously and the job successfully completed, the method returns a 200 OK status code. If the job timed out, the method returns a 202 Accepted status code.

If the method was invoked detached and successfully launched the job, the method returns a 202 Accepted status code and a link to the job resource. In this case, the response does not include a ServerJob entity.

Example

Example 1 Suspending a Server

This example uses the POST method to suspend a server synchronously.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/servers/id/Cluster-0-Server-1/suspend
```

Example Response

```
HTTP/1.1 200 OK
```

```
Response Body:
{
    "messages": [{
        "message": "Suspended the server 'Cluster-0-Server-1'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/server\/id\/Cluster-0-Ser
ver-1:_1_suspendWithTimeout"
    }],
    "item": {
        "operation": "suspending",
        "status": "completed",
        "beginTime": 1390587002462,
        "endTime": 1390587002489,
        "name": "_1_suspendWithTimeout",
        "id": "Cluster-0-Server-1:_1_suspendWithTimeout",
        "type": "server",
        "description": "suspending Cluster-0-Server-1 server ...",
        "serverName": "Cluster-0-Server-1"
    }
}
```

Example 2 Suspending a Server Asynchronously

This example uses the POST method to suspend a server in detached mode.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X POST
http://localhost:7001/management/wls/latest/servers/id/Cluster-0-Server-1/suspend?
__detached=true
```

Example Response

```
HTTP/1.1 202 Accepted

Response Body:
{
    "messages": [{
        "message": "Suspending the server 'Cluster-0-Server-1'.",
        "severity": "SUCCESS"
    }],
    "links": [{
        "rel": "job",
        "uri":
"http:\/\/localhost:7001\/management\/wls\/latest\/jobs\/server\/id\/Cluster-0-Ser
ver-1:_9_suspendWithTimeout"
    }]
}
```

/management/wls/{version}/targets

This resource lists all servers and clusters in this WLS domain.

The resource supports the following methods:

- GET Method

GET Method

The GET method on this resource returns all servers and clusters in this WLS domain.

Roles

Administrator, Deployer, Operator, Monitor

Response Body

The response body contains a collection of Target entities.

This method can return the following links:

- **uri**=/management/wls/{version} **rel**=parent

Response Codes

This method returns one of the Standard HTTP Status Codes.

Example

Example 1 Viewing WLS Targets

This example uses the GET method to return a list of all targets in this domain.

Example Request

```
curl -v \
--user username:password \
-H X-Requested-By:MyClient \
-H Accept:application/json \
-X GET http://localhost:7001/management/wls/latest/targets
```

Example Response

```
HTTP/1.1 200 OK

Response Body:
{
    "links": [{
        "rel": "parent",
        "uri": "http:\/\/localhost:7001\/management\/wls\/latest"
    }],
    "items": [
        {
            "name": "myserver",
            "type": "server"
        },
        {
            "name": "Cluster-0",
            "type": "cluster"
        }
    ]
}
```

2
Entities

The following sections describe the data models on which the REST resources operate for WebLogic Server 12.1.3. These data models describe the information exchanged in REST resources.

The descriptions of these data models in this section include a description of the constraints that apply to each data field.

Constraints on Data Fields

Confidential
Specifies that this value is encrypted and will not be returned by a `GET` method. It may be specified in a `POST` for update.

Immutable
Specifies that the contents of this field can be written once, during creation, and may not be modified thereafter.

Not Null
Specifies that a value must be specified for this field.

Read Only
Specifies that this value may be read by a `GET`, but is ignored during a `POST`.

Application

The Application entity includes configuration and monitoring information for an application deployment.

Note that an Application entity includes all the Deployment properties in addition to the following properties.

The properties of the Application entity are as follows:

planPath
(Optional) The pathname (relative to the Administration Server) of a file containing a deployment plan.

Type: String

Constraints: Immutable

archiveVersion
The archive version (from the deployment manifest file).

Type: String

Constraints: Read Only

planVersion
The deployment plan version (specified during deployment).

Type: String

Constraints: Immutable

applicationType
The application type: `ear`, `war`, `ejb`, or `gar`.

Type: String

Constraints: Read Only

health
The health of this application, if running. This health status is aggregated across all of the servers running this application and will denote the most severe health condition encountered.

Type: Health

Constraints: Read Only

urls
A list of the URLs for each context root. Can be used to ping the application.

Type: List<String>

Constraints: Read Only

openSessionsCurrentCount
The maximum total number of open sessions in this server. The count starts at zero each time the server is activated. Note that this is an optimization method for a highly useful statistic that could be implemented less efficiently using change notification.

Type: int

Constraints: Read Only

sessionsOpenedTotalCount
A count of the total number of sessions opened.

Type: int

Constraints: Read Only

servlets
The runtime information for all servlets in this application.

Type: List<Servlet>

Constraints: Read Only

ejbs
The runtime information for all EJBs in this application.

Type: List<Ejb>

Constraints: Read Only

ApplicationBindable

The ApplicationBindable entity describes information about a descriptor element that can be bound to a resource and overridden through a deployment plan.

The properties of the ApplicationBindable entity are as follows:

name
The name of this bindable descriptor value.

Type: String

Constraints: Read Only

path
The path in the deployment used to locate and identify this descriptor value.

Type: List<ApplicationDescriptorScope>

Constraints: Read Only

type
The type of resource that is expected to be bound to this descriptor value, if constrained: `jms destination`, `jms connection factory`, `resource adapter`, or `ejb`.

Type: String

Constraints: Read Only

jndiName
The JNDI name of the resource that should be bound to this application.

Type: String

ApplicationBindables

The ApplicationBindables entity is a collection of ApplicationBindable descriptor values in an application. Each entry represents a reference to a resource that can be specified and overridden in a deployment plan for an application.

The properties of the ApplicationBindables entity are as follows:

writeable
Writeable indicates that this application configuration supports overwriting deployment plan descriptor values. If a deployment is not supported by a deployment plan, then values may not be overwritten.

Type: boolean

Constraints: Read Only

bindables
Bindables is the collection of ApplicationBindable descriptor values.

Type: List<ApplicationBindable>

Constraints: Read Only

ApplicationDescriptorScope

The ApplicationDescriptorScope entity identifies a path or scope within a deployment descriptor tree.

The properties of the ApplicationDescriptorScope entity are as follows:

name
The name for this scope, intended to be human readable.

Type: String

Constraints: Read Only

path
The path of this scope, which identifies the deployment descriptor path.

Type: String

Constraints: Read Only

type
The type of this descriptor scope.

Type: String

Constraints: Read Only

ChangeManager

The ChangeManager entity includes properties that are required for change management.

The properties of the ChangeManager entity are as follows:

locked
`True` if a configuration session is currently active.

Type: boolean

Constraints: Read Only

lockOwner
The user name of the owner of the current configuration lock.

Type: String

Constraints: Read Only

hasChanges
`True` if there are any changes that have been saved but not activated in the current and previous edit sessions.

Type: boolean

Constraints: Read Only

DatabaseConnection

A DatabaseConnection entity describes the key properties necessary to establish a connection to a database. This is the basis for a DataSource entity, and may be used to validate that a database connection can be established successfully.

The properties of the DatabaseConnection entity are as follows:

url
The URL used to connect to the database.

Type: String

driverName
The driver name for this connection.

Type: String

properties
The properties for this connection.

Type: List<Property>

password
The password for this connection.

Type: String

Constraints: Confidential

DatabaseConnectionStatus

A DatabaseConnectionStatus entity describes the results that occur when a database connection is attempted.

The properties of the DatabaseConnectionStatus entity are as follows:

ok
The connection status. `True` if this connection was successful.

Type: boolean

Constraints: Read Only

cause
If issues were uncovered during this attempt, then `cause` can be used to retrieve human readable strings which describe these issues.

Type: List<String>

Constraints: Read Only

DatabaseDriver

The DatabaseDriver entity includes properties that are required to interact with a specific database driver.

The properties of the DatabaseDriver entity are as follows:

name
The name of this driver.

Type: String

Constraints: Read Only

driverClassName
The driver class name.

Type: String

Constraints: Read Only

type
The type of this driver, unconstrained, but typically `Thin`, `OCI`, `Type 2`, or `Type 4`.

Type: String

Constraints: Read Only

testSql
A test SQL statement that should be used when interacting with this driver.

Type: String

Constraints: Read Only

description
A human readable description for this driver.

Type: String

Constraints: Read Only

forXa
Whether this driver supports XA connections.

Type: boolean

Constraints: Read Only

DatabaseDriverAttribute

The DatabaseDriverAttribute entity represents a driver-specific attribute that is required to support connections with this driver.

The properties of the DatabaseDriverAttribute entity are as follows:

name
The name of this attribute.

Type: String

Constraints: Read Only

description
A human readable description string for this attribute.

Type: String

Constraints: Read Only

defaultValue
The default value for this attribute, if any.

Type: String

Constraints: Read Only

inUrl
If `true`, this attribute will be formatted and included in the URL string used to interact with the driver.

Type: boolean

Constraints: Read Only

required
If `true`, this attribute is required. Otherwise, attributes are optional and may be omitted.

Type: boolean

Constraints: Read Only

value
The user specified value for this attribute.

Type: String

DatabaseDriverAttributes

The DatabaseDriverAttributes entity represents a list of attributes that are associated with a database driver. These attributes represent driver-specific properties that are required to support connections with this driver.

The properties of the DatabaseDriverAttributes entity are as follows:

passwordAttributeName
The name of the password attribute, which requires special handling in clients to avoid surfacing confidential information.

Type: String

Constraints: Read Only

attributes
The list of attributes for this driver.

Type: List<DatabaseDriverAttribute>

DatabaseVendor

The DatabaseVendor entity describes a vendor of database drivers. Each vendor will have one or more drivers that can be selected and applied.

The properties of the DatabaseVendor entity are as follows:

name
The name of this database driver vendor.

Type: String

Constraints: Read Only

drivers
A list of drivers available from this vendor.

Type: List<DatabaseDriver>

Constraints: Read Only

DataSource

The DataSource entity represents the configuration and runtime metrics for a data source resource, used by applications to interact with a database.

The properties of the DataSource entity are as follows:

name
The name of this data source.

Type: String

Constraints: Immutable, Not Null

jdbcDataSourceParams
The general parameters of this data source. Basic configuration parameters for this data source are specified using these parameters.

Type: JdbcDataSourceParams

Constraints: Read Only

jdbcConnectionPoolParams
The connection pool parameters of this data source. Configuration parameters for this data source's connection pool are specified using these parameters.

Type: JdbcConnectionPoolParams

Constraints: Read Only

jdbcDriverParams
The driver parameters for this data source. Configuration parameters for the JDBC driver used by this data source are specified using these parameters.

Type: JdbcDriverParams

Constraints: Read Only

aggregateMetrics
Aggregate metrics calculated across all running instances of this data source.

Type: DataSourceMetrics

Constraints: Read Only

dataSourceMetrics
The list of per server runtime metrics for this data source.

Type: List<DataSourceMetrics>

Constraints: Read Only

targets
The list of the targets for this data source.

Type: List<String>

Constraints: Read Only

DataSourceMetrics

The DataSourceMetrics entity includes attributes that can be used to monitor a data source.

The properties of the DataSourceMetrics entity are as follows:

serverName
The name of the server where this data source instance runs.

Type: String

Constraints: Read Only

state

The state of this data source: `running`, `suspended`, `shutdown`, `overloaded`, or `unknown`.

Type: String

Constraints: Read Only

leakedConnectionCount

The number of leaked connections. A leaked connection is a connection that was reserved from the data source but was not returned to the data source by calling `close()`.

Type: int

Constraints: Read Only

failuresToReconnectCount

The number of times that the data source attempted to refresh a database connection and failed. Failures may occur when the database is unavailable or when the network connection to the database is interrupted.

Type: int

Constraints: Read Only

connectionDelayTime

The average amount of time, in milliseconds, that it takes to create a physical connection to the database. The value is calculated as summary of all times to connect divided by the total number of connections.

Type: int

Constraints: Read Only

prepStmtCacheAccessCount

The cumulative, running count of the number of times that the statement cache was accessed.

Type: long

Constraints: Read Only

prepStmtCacheAddCount

The cumulative, running count of the number of statements added to the statement cache. Each connection in the connection pool has its own cache of statements. This number is the sum of the number of statements added to the caches for all connections in the connection pool.

Type: long

Constraints: Read Only

prepStmtCacheDeleteCount

The cumulative, running count of statements discarded from the cache. Each connection in the connection pool has its own cache of statements. This number is the sum of the number of statements that were discarded from the caches for all connections in the connection pool.

Type: long

Constraints: Read Only

prepStmtCacheCurrentSize
The number of prepared and callable statements currently cached in the statement cache. Each connection in the connection pool has its own cache of statements. This number is the sum of the number of statements in the caches for all connections in the connection pool.

Type: int

Constraints: Read Only

prepStmtCacheHitCount
The cumulative, running count of the number of times that statements from the cache were used.

Type: int

Constraints: Read Only

prepStmtCacheMissCount
The number of times that a statement request could not be satisfied with a statement from the cache.

Type: int

Constraints: Read Only

activeConnectionsCurrentCount
The number of connections currently in use by applications.

Type: int

Constraints: Read Only

waitingForConnectionCurrentCount
The number of connection requests waiting for a database connection.

Type: int

Constraints: Read Only

activeConnectionsHighCount
The highest number of active database connections in this instance of the data source since the data source was instantiated. Active connections are connections in use by an application.

Type: int

Constraints: Read Only

activeConnectionsAverageCount
The average number of active connections in this instance of the data source. Active connections are connections in use by an application. This value is only valid if the resource is configured to allow shrinking.

Type: int

Constraints: Read Only

reserveRequestCount
The cumulative, running count of requests for a connection from this data source.

Type: long

Constraints: Read Only

failedReserveRequestCount

The cumulative, running count of requests for a connection from this data source that could not be fulfilled.

Type: long

Constraints: Read Only

waitingForConnectionHighCount

Highest number of application requests concurrently waiting for a connection from this instance of the data source.

Type: int

Constraints: Read Only

waitingForConnectionTotal

The cumulative, running count of requests for a connection from this 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.

Type: long

Constraints: Read Only

waitingForConnectionSuccessTotal

The cumulative, running count of requests for a connection from this data source that had to wait before getting a connection and eventually succeeded in getting a connection.

Type: long

Constraints: Read Only

waitingForConnectionFailureTotal

The cumulative, running count of requests for a connection from this data source that had to wait before getting a connection and eventually failed to get a connection. Waiting connection requests can fail for a variety of reasons, including waiting for longer than the ConnectionReserveTimeoutSeconds.

Type: long

Constraints: Read Only

waitSecondsHighCount

The highest number of seconds that an application waited for a connection (the longest connection reserve wait time) from this instance of the connection pool since the connection pool was instantiated. This value is updated when a completed getConnection request takes longer to return a connection than any previous request.

Type: int

Constraints: Read Only

connectionsTotalCount

The cumulative total number of database connections created in this data source since the data source was deployed.

Type: int

Constraints: Read Only

currCapacity
The current count of JDBC connections in the connection pool in the data source.

Type: int

Constraints: Read Only

currCapacityHighCount
The highest number of database connections available or in use (current capacity) in this instance of the data source since the data source was deployed.

Type: int

Constraints: Read Only

numAvailable
The number of database connections that are currently idle and available to be used by applications in this instance of the data source.

Type: int

Constraints: Read Only

highestNumAvailable
The highest number of database connections that were idle and available to be used by an application at any time in this instance of the data source since the data source was deployed.

Type: int

Constraints: Read Only

numUnavailable
The number of connections currently in use by applications or being tested in this instance of the data source.

Type: int

Constraints: Read Only

highestNumUnavailable
The highest number of database connections that were in use by applications or being tested by the system in this instance of the data source since the data source was deployed.

Type: int

Constraints: Read Only

Deployment

The Deployment entity includes essential information that is common to all deployment types.

The properties of the Deployment entity are as follows:

name
The name of this deployment.

Type: String

Constraints: Immutable

displayName
A human readable name for this deployment, with formatted version information.

Type: String

Constraints: Read Only

deploymentPath
The pathname (relative to the Administration Server) of a file containing the deployment archive or a directory containing an exploded deployment.

Type: String

Constraints: Immutable

type
The type of this deployment: `application` or `library`.

Type: String

Constraints: Read Only

state
The state of this deployment: `active`, `admin`, `update pending`, `prepared`, `failed`, `retired`, or `new`.

Type: String

Constraints: Read Only

targets
The list of the targets for this deployment.

Type: List<String>

Constraints: Read Only

DeploymentDescription

The DeploymentDescription entity provides access to the configuration of a deployment.

The properties of the DeploymentDescription entity are as follows:

deploymentPath
The pathname (relative to the Administration Server) of a file containing the deployment archive or a directory containing an exploded deployment.

Type: String

Constraints: Read Only

planPath
(Optional) The pathname (relative to the Administration Server) of a file containing a deployment plan.

Type: String

Constraints: Read Only

deploymentType
The type of deployment: `library`, `application`, or `unknown`.

Type: String

Constraints: Read Only

name
The name of this deployment, if available. (This is the suggested name for a new deployment.)

Type: String

Constraints: Read Only

archiveVersion
The archive version if the deployment is an application and specifies its archive version.

Type: String

Constraints: Read Only

planVersion
The plan version if the deployment is an application and specifies its plan version.

Type: String

Constraints: Read Only

specVersion
The specification version if the deployment is a library and specifies its specification version.

Type: String

Constraints: Read Only

implVersion
The implementation version if the deployment is a library and specifies its implementation version.

Type: String

Constraints: Read Only

DeploymentJob

The DeploymentJob entity includes the attributes of a Job entity, and adds the following attributes that are unique to a deployment operation.

The properties of the DeploymentJob entity are as follows:

status
The status of this deployment job: `initialized`, `in progress` `completed`, `failed`, or `deferred`.

Type: String

Constraints: Read Only

operation
The deployment operation that this job is tracking: `activate`, `prepare`, `deactivate`, `remove`, `unprepare`, `distribute`, `start`, `stop`, `redeploy`, `update`, `deploy`, or `undeploy`.

Type: String

Constraints: Read Only

deploymentName
The name of the deployment affected by this job.

Type: String

Constraints: Read Only

targets
The target specific status information associated with this job.

Type: List<DeploymentJobTarget>

Constraints: Read Only

DeploymentJobTarget

The DeploymentJobTarget entity provides access to target-specific status information for a deployment job.

The properties of the DeploymentJobTarget entity are as follows:

name
The name of this target.

Type: String

Constraints: Read Only

type
The type of this target: `unknown`, `server`, `cluster` `JMS server`, `virtual host`, or `SAF agent`.

Type: String

Constraints: Read Only

status
The status of the deployment operation on this target: `initialized`, `in progress` `completed`, `failed`, or `unavailable`.

Type: String

Constraints: Read Only

errors
A list of error messages that describe issues that occurred during this deployment job, if any.

Type: List<String>

Constraints: Read Only

DeploymentReference

A DeploymentReference entity is used to identify a deployment for inspection or deployment.

The properties of the DeploymentReference entity are as follows:

deploymentPath
The pathname (relative to the Administration Server) of a file containing the deployment archive or a directory containing an exploded deployment.

Type: String

Constraints: Not Null

planPath
(Optional) The pathname (relative to the Administration Server) of a file containing a deployment plan.

Type: String

Ejb

The Ejb entity includes information for each EJB.

The properties of the Ejb entity are as follows:

ejbName
The name for this EJB as defined in the `javax.ejb.EJB` annotation, or the ejb-name when using the `ejb-jar.xml` deployment descriptor.

Type: String

Constraints: Read Only

type
The type of this EJB: `entity`, `stateless`, `stateful`, `singleton`, or `message driven`.

Type: String

Constraints: Read Only

moduleName
The name of this EJB module.

Type: String

Constraints: Read Only

aggregateMetrics
Aggregate metrics calculated across all running instances of this EJB.

Type: EjbMetrics

Constraints: Read Only

ejbMetrics
The list of per server runtime metrics for this EJB.

Type: List<EjbMetrics>

Constraints: Read Only

EjbCacheMetrics

The EjbCacheMetrics entity includes runtime cache monitoring information for each EJB.

The properties of the EjbCacheMetrics entity are as follows:

cachedBeansCurrentCount
A count of the total number of beans from this EJB Home currently in the EJB cache.

Type: int

Constraints: Read Only

cacheAccessCount
A count of the total number of attempts to access a bean from the cache.

Type: long

Constraints: Read Only

cacheMissCount
A count of the total number of times an attempt to access a bean from the cache failed.

Type: long

Constraints: Read Only

activationCount
A count of the total number of beans from this EJB Home that have been activated.

Type: long

Constraints: Read Only

passivationCount
A count of the total number of beans from this EJB Home that have been passivated.

Type: long

Constraints: Read Only

EjbLockingMetrics

The EjbLockingMetrics entity includes runtime locking monitoring information for each EJB.

The properties of the EjbLockingMetrics entity are as follows:

lockEntriesCurrentCount
A count of the number of beans currently locked.

Type: int

Constraints: Read Only

lockManagerAccessCount
The total number of attempts to obtain a lock on a bean. This includes attempts to obtain a lock on a bean that is already locked on behalf of the client.

Type: long

Constraints: Read Only

waiterTotalCount
The total number of threads that have waited for a lock on a bean.

Type: long

Constraints: Read Only

waiterCurrentCount
The current number of threads that have waited for a lock on a bean.

Type: int

Constraints: Read Only

timeoutTotalCount
The current number of threads that have timed out waiting for a lock on a bean.

Type: long

Constraints: Read Only

EjbMetrics

The EjbMetrics entity includes runtime monitoring information for an EJB instance.

The properties of the EjbMetrics entity are as follows:

serverName
The name of the server where this EJB instance runs.

Type: String

Constraints: Read Only

transactionMetrics
The EJB transaction metrics, if applicable.

Type: EjbTransactionMetrics

Constraints: Read Only

timerMetrics
The EJB timer metrics, if applicable.

Type: EjbTimerMetrics

Constraints: Read Only

poolMetrics
The EJB pool metrics, if applicable.

Type: EjbPoolMetrics

Constraints: Read Only

cacheMetrics
The EJB caching metrics, if applicable.

Type: EjbCacheMetrics

Constraints: Read Only

lockingMetrics
The EJB locking metrics, if applicable.

Type: EjbLockingMetrics

Constraints: Read Only

EjbPoolMetrics

The EjbPoolMetrics entity includes runtime pool monitoring information for each EJB.

The properties of the EjbPoolMetrics entity are as follows:

accessTotalCount
A count of the total number of times an attempt was made to get an instance from the free pool.

Type: long

Constraints: Read Only

missTotalCount
A count of the total number of times a failed attempt was made to get an instance from the free pool. An attempt to get a bean from the pool will fail if there are no available instances in the pool.

Type: long

Constraints: Read Only

destroyedTotalCount
A count of the total number of times a bean instance from this pool was destroyed due to a non-application exception being thrown from it.

Type: long

Constraints: Read Only

pooledBeansCurrentCount
A count of the current number of available bean instances in the free pool.

Type: int

Constraints: Read Only

beansInUseCurrentCount
A count of the number of bean instances currently being used from the free pool.

Type: int

Constraints: Read Only

waiterCurrentCount
A count of the total number of threads currently waiting for an available bean instance from the free pool.

Type: int

Constraints: Read Only

timeoutTotalCount
A count of the total number of threads that have timed out waiting for an available bean instance from the free pool.

Type: long

Constraints: Read Only

EjbTimerMetrics

The EjbTimerMetrics entity includes runtime timer monitoring information for each EJB.

The properties of the EjbTimerMetrics entity are as follows:

timeoutCount
The total number of successful timeout notifications that have been made for this EJB.

Type: long

Constraints: Read Only

cancelledTimerCount
The total number of timers that have been explicitly cancelled for this EJB.

Type: long

Constraints: Read Only

activeTimerCount
The current number of active timers for this EJB.

Type: int

Constraints: Read Only

disabledTimerCount
The current number of timers temporarily disabled for this EJB.

Type: int

Constraints: Read Only

EjbTransactionMetrics

The EjbTransactionMetrics entity includes runtime transaction monitoring information for each EJB.

The properties of the EjbTransactionMetrics entity are as follows:

transactionsCommittedTotalCount
A count of the total number of transactions that have been committed for this EJB.

Type: long

Constraints: Read Only

transactionsRolledBackTotalCount
A count of the total number of transactions that have been rolled back for this EJB.

Type: long

Constraints: Read Only

transactionsTimedOutTotalCount
A count of the total number of transactions that have timed out for this EJB.

Type: long

Constraints: Read Only

Health

The Health entity represents the health of a WebLogic domain.

The properties of the Health entity are as follows:

state
The domain health state: `ok`, `warn`, `critical`, `failed`, `overloaded`, or `unknown`.

Type: String

Constraints: Read Only

reasons
The reasons for the health state of this domain.

Type: String

Constraints: Read Only

JdbcConnectionPoolParams

The JdbcConnectionPoolParams entity includes parameters for a data source's connection pool.

The properties of the JdbcConnectionPoolParams entity are as follows:

initialCapacity

The number of physical connections to create when creating the connection pool in the data source. If unable to create this number of connections, creation of the data source will fail.

Type: int

maxCapacity

The maximum number of physical connections that this connection pool can contain.

Type: int

minCapacity

The minimum number of physical connections that this connection pool can contain after it is initialized.

Type: int

shrinkFrequencySeconds

The number of seconds to wait before shrinking a connection pool that has incrementally increased to meet demand. When set to zero (0), shrinking is disabled.

Type: int

highestNumWaiters

The maximum number of connection requests that can concurrently block threads while waiting to reserve a connection from the data source's connection pool.

Type: int

connectionCreationRetryFrequencySeconds

If you do not set this value, data source creation fails if the database is unavailable. If set and if the database is unavailable when the data source is created, WebLogic Server will attempt to create connections in the pool again after the number of seconds you specify, and will continue to attempt to create the connections until it succeeds. When set to zero (0), connection retry is disabled.

Type: int

connectionReserveTimeoutSeconds

The number of seconds after which a call to reserve a connection from the connection pool will timeout. When set to zero (0), a call will never timeout. When set to -1, a call will timeout immediately.

Type: int

testFrequencySeconds

The number of seconds a WebLogic Server instance waits between attempts when testing unused connections. (Requires that you specify a Test Table Name.)

Connections that fail the test are closed and reopened to re-establish a valid physical connection. If the test fails again, the connection is closed.

In the context of multi data sources, this attribute controls the frequency at which WebLogic Server checks the health of data sources it had previously marked as unhealthy.

When set to zero (0), the feature is disabled.

Type: int

testConnectionsOnReserve

Enables WebLogic Server to test a connection before giving it to a client. (Requires that you specify a Test Table Name.)

The test adds a small delay in serving the client's request for a connection from the pool, but ensures that the client receives a viable connection.

Type: boolean

profileHarvestFrequencySeconds

The number of seconds between when WebLogic Server harvests profile data. When set to zero (0), harvesting of data is disabled.

Type: int

ignoreInUseConnectionsEnabled

Enables the data source to be shutdown even if connections obtained from the pool are still in use.

Type: boolean

inactiveConnectionTimeoutSeconds

The number of inactive seconds on a reserved connection before WebLogic Server reclaims the connection and releases it back into the connection pool.

You can use the Inactive Connection Timeout feature to reclaim leaked connections - connections that were not explicitly closed by the application. Note that this feature is not intended to be used in place of properly closing connections.

When set to zero (0), the feature is disabled.

Type: int

testTableName

The name of the database table to use when testing physical database connections. This name is required when you specify a Test Frequency and enable Test Reserved Connections.

The default SQL code used to test a connection is `select count(*) from TestTableName`. Most database servers optimize this SQL to avoid a table scan, but it is still a good idea to set the Test Table Name to the name of a table that is known to have few rows, or even no rows.

If the Test Table Name begins with `SQL`, then the rest of the string following that leading token will be taken as a literal SQL statement that will be used to test connections instead of the standard query. For example: `SQL BEGIN; Null; END;`

For an Oracle database, you can reduce the overhead of connection testing by setting Test Table Name to `SQL PINGDATABASE` which uses the `pingDatabase()` method to test the Oracle connection.

Type: String

loginDelaySeconds

The number of seconds to delay before creating each physical database connection. This delay supports database servers that cannot handle multiple connection requests in rapid succession.

The delay takes place both during initial data source creation and during the lifetime of the data source whenever a physical database connection is created.

Type: int

initSql

SQL statement to execute that will initialize newly created physical database connections. Start the statement with `SQL` followed by a space.

If the Init SQL value begins with `SQL`, then the rest of the string following that leading token will be taken as a literal SQL statement that will be used to initialize database connections. If the Init SQL value does not begin with `SQL`, the value will be treated as the name of a table and the following SQL statement will be used to initialize connections: `select count(*) from InitSQL`.

The table `InitSQL` must exist and be accessible to the database user for the connection. Most database servers optimize this SQL to avoid a table scan, but it is still a good idea to set InitSQL to the name of a table that is known to have few rows, or even no rows.

Type: String

Constraints: Not Null

statementCacheSize

The number of prepared and callable statements stored in the cache. (This may increase server performance.)

WebLogic Server can reuse statements in the cache without reloading the statements, which can increase server performance. Each connection in the connection pool has its own cache of statements.

Setting the size of the statement cache to zero (0) turns off statement caching.

Type: int

statementCacheType

The algorithm used for maintaining the prepared statements stored in the statement cache:

- `least recently used` - when a new prepared or callable statement is used, the least recently used statement is replaced in the cache.
- `fixed` - the first fixed number of prepared and callable statements are cached.

Type: String

removeInfectedConnections

Specifies whether a connection will be removed from the connection pool after the application uses the underlying vendor connection object.

If you disable removing infected connections, you must make sure that the database connection is suitable for reuse by other applications.

When set to `true` (the default), the physical connection is not returned to the connection pool after the application closes the logical connection. Instead, the physical connection is closed and recreated.

When set to `false`, when the application closes the logical connection, the physical connection is returned to the connection pool and can be reused by the application or by another application.

Type: boolean

secondsToTrustAnIdlePoolConnection

The number of seconds within a connection that WebLogic Server trusts that the connection is still viable and will skip the connection test, either before delivering it to an application or during the periodic connection testing process.

This option is an optimization that minimizes the performance impact of connection testing, especially during heavy traffic.

Type: int

statementTimeout

The time after which a statement currently being executed will time out.

Statement timeout relies on underlying JDBC driver support. WebLogic Server passes the time specified to the JDBC driver using the `java.sql.Statement.setQueryTimeout()` method. If your JDBC driver does not support this method, it may throw an exception and the timeout value is ignored.

A value of -1 disables this feature. A value of zero (0) means that statements will not time out.

Type: int

profileType

The type of profile data to be collected for the JDBC subsystem.

Type: int

jdbcXaDebugLevel

The level of JDBC debugging for XA drivers, where larger values in the range provide more debugging information.

Type: int

credentialMappingEnabled

Enables Set Client ID on connection for the data source. When an application requests a database connection, WebLogic Server sets a light-weight client ID on the database connection.

By default, it uses the credential mapping to map WebLogic Server user IDs to database user IDs. However, if use-database-credentials is set to `true`, then the credential mapping is not done and the ID is used directly as a database user ID.

It is currently supported for IBM DB2 driver and Oracle thin driver. Support for this feature will be dropped in a future Oracle thin driver release. Oracle recommends using proxy authentication instead of this feature.

Type: boolean

driverInterceptor

The absolute name of the application class used to intercept method calls to the JDBC driver. The application specified must implement the `weblogic.jdbc.extensions.DriverInterceptor` interface.

WebLogic Server will invoke the `preInvokeCallback()`, `postInvokeExceptionCallback()`, and `postInvokeCallback()` methods of the

registered application before and after invoking any method inside the JDBC driver. You can use this feature to profile JDBC driver usage and monitor:

- Methods being executed
- Any exceptions thrown
- Time spent inside the driver executing methods

Type: String

Constraints: Not Null

pinnedToThread

Enables an option to improve performance by enabling execute threads to keep a pooled database connection even after the application closes the logical connection. When enabled:

- WebLogic Server pins a database connection from the connection pool to an execution thread the first time an application uses the thread to reserve a connection. When the application finishes using the connection and calls `connection.close()`, WebLogic Server keeps the connection with the execute thread and does not return it to the connection pool. When an application subsequently requests a connection using the same execute thread, WebLogic Server provides the connection already reserved by the thread.
- There is no locking contention on the connection pool that occurs when multiple threads attempt to reserve a connection at the same time. There is no contention for threads that attempt to reserve the same connection from a limited number of database connections.
- If an application concurrently reserves more than one connection from the connection pool using the same execute thread, WebLogic Server creates additional database connections and pins them to the thread.

Type: boolean

identityBasedConnectionPoolingEnabled

Enables identity-based-connection-pooling for the data source. When an application requests a database connection, WebLogic Server picks or creates a physical connection with the requested DBMS identity based on a map of WebLogic user IDs and database IDs. You must also specify the map of WebLogic Server user IDs to database user IDs (credential mapping).

Type: boolean

wrapTypes

By default, data type objects for Array, Blob, Clob, NClob, Ref, SQLXML, and Struct, plus ParameterMetaData and ResultSetMetaData objects are wrapped with a WebLogic wrapper. This allows features like debugging and connection usage to be done by the server. The wrapping can be turned off by setting this value to `false`. This improves performance, in some cases significantly, and allows the application to use the native driver objects directly.

Type: boolean

fatalErrorCodes

A comma-separated list of error codes that are treated as fatal errors. These errors include deployment errors that cause a server to fail to boot and connection errors that prevent a connection from being put back in the connection pool.

This optional property is used to define fatal error codes, that when specified as the exception code within a SQLException (retrieved by `sqlException.getErrorCode()`), indicate that a fatal error has occurred and the connection is no longer usable. For Oracle databases the following fatal error codes are predefined within WLS and do not need to be placed in the configuration file:

- 3113: `end-of-file on communication channel`
- 3114: `not connected to ORACLE`
- 1033: `ORACLE initialization or shutdown in progress`
- 1034: `ORACLE not available`
- 1089: `immediate shutdown in progress - no operations are permitted`
- 1090: `shutdown in progress - connection is not permitted`
- 17002: `I/O exception`

Type: String

Constraints: Not Null

connectionLabelingCallback

The class name of the connection labeling callback. This is automatically passed to `registerConnectionLabelingCallback` when the data source is created. The class must implement `oracle.ucp.ConnectionLabelingCallback`.

Type: String

Constraints: Not Null

connectionHarvestMaxCount

The maximum number of connections that may be harvested when the connection harvesting occurs. The range of valid values is `1` to `MaxCapacity`.

Type: int

connectionHarvestTriggerCount

The number of available connections (trigger value) used to determine when connection harvesting occurs.

- Harvesting occurs when the number of available connections is below the trigger value for a connection pool.
- The range of valid values is `-1` to `MaxCapacity`.
- The default value is `-1`.
- Setting the value to `-1` disables connection harvesting.

Type: int

countOfTestFailuresTillFlush

The number of test failures allowed before WebLogic Server closes all connections in a connection pool to minimize the delay caused by further database testing.

Type: int

countOfRefreshFailuresTillDisable

The number of refresh failures allowed before WebLogic Server closes all connections in a connection pool to minimize the delay caused by further database testing.

Type: int

wrapJdbc
By default, SQL objects for CallableStatement, PreparedStatement, ResultSet, Statement, and DatabaseMetaData are wrapped with a WebLogic wrapper. Wrapping allows features like debugging and connection usage to be performed by the server.

When `false`, wrapping is disabled. This improves performance, in some cases significantly, and allows for the application to use the native driver objects directly. A value of `false` also disables data type wrapping.

Type: boolean

JdbcDataSourceParams

The JdbcDataSourceParams entity includes the basic configuration settings for a data source.

The properties of the JdbcDataSourceParams entity are as follows:

jndiNames
The JNDI path to where this data source is bound. By default, the JNDI name is the name of the data source. Applications that look up the JNDI path will get a `javax.sql.DataSource` instance that corresponds to this data source.

Type: List<String>

scope
The scoping of the data source: `global` or `application`

Type: String

rowPrefetch
Enables multiple rows to be "prefetched" (that is, sent from the server to the client) in one server access.

Type: boolean

rowPrefetchSize
If row prefetching is enabled, specifies the number of result set rows to prefetch for a client.

Type: int

streamChunkSize
The data chunk size for streaming data types.

Type: int

algorithmType
The algorithm that determines the connection request processing for the multi data source: `failover` or `load balancing`

Type: String

dataSourceList
The list of data sources to which the multi data source will route connection requests. The order of data sources in the list determines the failover order.

Type: String

connectionPoolFailoverCallbackHandler
The name of the application class to handle the callback sent when a multi data source is ready to failover or fail back connection requests to another data source within the multi data source. The name must be the absolute name of an application class that implements the `weblogic.jdbc.extensions.ConnectionPoolFailoverCallback` interface.

Type: String

failoverRequestIfBusy
For multi data sources with the `failover` algorithm, enables the multi data source to failover connection requests to the next data source if all connections in the current data source are in use.

Type: boolean

globalTransactionsProtocol
The transaction protocol (global transaction processing behavior) for the data source: `two phase commit`, `logging last resource`, `emulate two phase commit`, `one phase commit`, or `none`

Type: String

keepConnAfterLocalTx
Enables WebLogic Server to keep the physical database connection associated with the logical connection when committing a local transaction instead of releasing it and getting another physical connection when needed. Setting this property to `true` may require additional connections to be configured on the database.

Type: boolean

keepConnAfterGlobalTx
Enables WebLogic Server to keep the physical database connection associated with the logical connection when committing a global transaction instead releasing it and getting another physical connection when needed. Setting this property to `true` may require additional connections to be configured on the database.

Type: boolean

JdbcDriverParams

A JdbcDriverParams entity includes the driver parameters for the JDBC driver used by a data source.

The properties of the JdbcDriverParams entity are as follows:

useXaDataSourceInterface
Specifies that WebLogic Server should use the XA interface of the JDBC driver. If the JDBC driver class used to create database connections implements both XA and non-XA versions of a JDBC driver, you can set this attribute to indicate that WebLogic Server should treat the JDBC driver as an XA driver or as a non-XA driver.

Type: boolean

usePasswordIndirection
Specifies whether the database credentials are to be obtained from the credential mapper using the "user" property as the key. When `true`, the credentials are obtained from the credential mapper. When `false`, the database user name and password are obtained from the "user" property and Password element, respectively.

Type: boolean

systemProperties
The list of properties passed to the JDBC driver when creating physical database connections.

To enable driver-level features, add the driver property and its value to the Properties list. WebLogic Server sets driver-level properties in the Properties list on the driver's `ConnectionPoolDataSource` object.

NOTE: For Security reasons, when WebLogic Server is running in production mode, you cannot specify database passwords in this properties list. Data source deployment will fail if a password is specified in the properties list. To override this security check, use the command line argument `weblogic.management.allowClearTextPasswords` when starting the server.

Type: List<Property>

Job

The Job entity includes information about an asynchronous task.

The properties of the Job entity are as follows:

id
A unique identity for the job within its specific job type. Not meant to be displayed. Useful for constructing URLs and for querying for additional job information.

Type: String

Constraints: Read Only

name
The name associated with this job.

Type: String

Constraints: Read Only

type
The type of job: `server` or `deployment`

Type: String

Constraints: Read Only

status
The status of this job.

Type: String

Constraints: Read Only

description
A description associated with this job.

Type: String

Constraints: Read Only

beginTime
The time at which this job was started.

Type: long

Constraints: Read Only

endTime
The time at which this job was completed. Zero (0) if running.

Type: long

Constraints: Read Only

error
An error message for this job, if any.

Type: String

Constraints: Read Only

Library

A Library entity includes information about a library deployment, including the libraries and applications which are dependent on this library.

Note that a Library entity includes all the Deployment properties in addition to the following properties.

The properties of the Library entity are as follows:

specVersion
The library specification version (from the deployment manifest file).

Type: String

Constraints: Read Only

implVersion
The library implementation version (from the deployment manifest file).

Type: String

Constraints: Read Only

referencingApplications
The list of application names that reference this library.

Type: List<String>

Constraints: Read Only

referencingLibraries
The list of other library names that reference this library.

Type: List<String>

Constraints: Read Only

LogEntry

The LogEntry entity describes a specific log entry in a log file.

The fields recorded in a log file in WebLogic Server vary considerably according to the log type: data source, HTTP access, or server log. As such, the information returned in a LogEntry entity will also vary depending on the log file being viewed.

The properties of the LogEntry entity are as follows:

recordId
The unique identifier for this log entry in this log.

Type: String

Constraints: Read Only

timeStamp
The date and time of this log entry. Not available for resource adapter logs.

Type: String

Constraints: Read Only

severity
The severity of this condition, if applicable to this log type.

Type: String

Constraints: Read Only

subSystem
The subsystem associated with this condition, if applicable to this log type.

Type: String

Constraints: Read Only

msgId
The message ID of this condition, if applicable to this log type.

Type: String

Constraints: Read Only

message
The message associated with this condition, if applicable to this log type.

Type: String

Constraints: Read Only

host
The host of this condition, if applicable to this log type.

Type: String

Constraints: Read Only

remoteUser
The remote user associated with this condition, if applicable to this log type.

Type: String

Constraints: Read Only

authUser
The authorized user associated with this condition, if applicable to this log type.

Type: String

Constraints: Read Only

request
The request associated with this condition, if applicable to this log type.

Type: String

Constraints: Read Only

status
The status of this condition, if applicable to this log type.

Type: String

Constraints: Read Only

userId
The user ID associated with this condition, if applicable to this log type.

Type: String

Constraints: Read Only

contextId
The context ID associated with this condition, if applicable to this log type.

Type: String

Constraints: Read Only

datasource
The data source associated with this condition, if applicable to this log type.

Type: String

Constraints: Read Only

profileType
The profile type associated with this condition, if applicable to this log type.

Type: String

Constraints: Read Only

user
The user associated with this condition, if applicable to this log type.

Type: String

Constraints: Read Only

profileInformation
The profile information associated with this condition, if applicable to this log type.

Type: String

Constraints: Read Only

LogReference

The LogReference entity describes a specific, named log.

The properties of the LogReference entity are as follows:

name
The name of the log.

Type: String

Constraints: Read Only

Server

The Server entity describes the configuration, state, and metrics associated with a WebLogic Server instance.

The properties of the Server entity are as follows:

name
The name of this server instance.

Type: String

Constraints: Read Only

state
The current state of this server instance:

- `shutdown`
- `starting`
- `running`
- `standby`
- `admin`
- `suspending`
- `force suspending`
- `resuming`
- `shutting down`
- `force shutting down`
- `failed`
- `unknown`
- `discovered`
- `failed restarting`
- `activate later`
- `failed not restartable`
- `failed migratable`

Type: String

Constraints: Read Only

health
The health of this server instance, if running. This health status is aggregated across all of the components of a server and will denote the most severe health condition encountered.

Type: Health

Constraints: Read Only

jvmProcessorLoad
The average load that the VM is placing on all processors, in percent: 0% to 100%.

Type: int

Constraints: Read Only

usedPhysicalMemory
The amount (in bytes) of physical memory that is currently being used by this server.

Type: long

Constraints: Read Only

heapSizeCurrent
The current size (in bytes) of the JVM heap used by this server.

Type: long

Constraints: Read Only

heapFreeCurrent
The current amount of memory (in bytes) that is available in the JVM heap.

Type: long

Constraints: Read Only

activeHttpSessionCount
The number of current active HTTP sessions for all servlets hosted by this server.

Type: int

Constraints: Read Only

activeThreadCount
The number of active threads being used by all applications and resources on this server.

Type: int

Constraints: Read Only

ServerJob

The ServerJob entity includes the attributes of a Job entity, and adds additional attributes that are unique to a server life cycle operation.

The properties of the ServerJob entity are as follows:

status
The server job status: `in progress`, `completed`, or `failed`.

Type: String

Constraints: Read Only

serverName
The name of the server affected by this job.

Type: String

Constraints: Read Only

operation
The server life cycle state transitions tracked by this job:

- `start`
- `shutdown`

- `start in standby`
- `resume`
- `force shutdown`
- `suspending`
- `force suspend`

Type: String

Constraints: Read Only

Servlet

The Servlet entity includes information for each servlet.

The properties of the Servlet entity are as follows:

servletName
The name of this instance of a servlet.

Type: String

Constraints: Read Only

moduleName
The module associated with this servlet.

Type: String

Constraints: Read Only

contextPath
The context path for this servlet.

Type: String

Constraints: Read Only

aggregateMetrics
Aggregate metrics calculated across all running instances of this servlet.

Type: ServletMetrics

Constraints: Read Only

servletMetrics
The list of per server runtime metrics for this servlet.

Type: List<ServletMetrics>

Constraints: Read Only

ServletMetrics

The ServletMetrics entity includes runtime monitoring information for a servlet instance.

The properties of the ServletMetrics entity are as follows:

serverName
The name of the server where this servlet instance runs.

Type: String

Constraints: Read Only

reloadTotalCount
A total count of the number of times this servlet has been reloaded.

Type: int

Constraints: Read Only

invocationTotalCount
A total count of the number of times this servlet has been invoked.

Type: int

Constraints: Read Only

executionTimeTotal
The total amount of time all invocations of the servlet have executed since created.

Type: long

Constraints: Read Only

executionTimeHigh
The amount of time the single longest invocation of the servlet has executed since created.

Type: int

Constraints: Read Only

executionTimeLow
The amount of time the single shortest invocation of the servlet has executed since created. Note that for the CounterMonitor, the difference option must be used.

Type: int

Constraints: Read Only

Target

The Target entity represents a WLS target.

The properties of the Target entity are as follows:

name
The name of this target.

Type: String

Constraints: Read Only

type
The type of this target: `server` or `cluster`.

Type: String

Constraints: Read Only

Version

A Version entity describes a version of the WLS resources.

The properties of the Version entity are as follows:

version
The name of this version.

Type: String

Constraints: Read Only

isLatest
`True` if this is the default version.

Type: boolean

Constraints: Read Only

state
The life cycle of this version: `active` or `deprecated`

Type: String

Constraints: Read Only

lifecycle
Type: String

Wls

The Wls entity provides access to the overall state and condition of the WebLogic Server domain.

The properties of the Wls entity are as follows:

name
The name of this domain.

Type: String

Constraints: Read Only

productionMode
`True` if this domain is running in production mode. `False` if the domain is running in development mode.

Type: boolean

Constraints: Read Only

activeHttpSessionCount
The total number of active HTTP sessions in all applications and all servers in this domain.

Type: int

Constraints: Read Only

activeThreadCount
The total number of active threads currently in use on all servers.

Type: int

Constraints: Read Only

configuredServerCount
The number of potential WebLogic Server instances.

Type: int

Constraints: Read Only

activeServerCount
The number of currently active WebLogic Server instances.

Type: int

Constraints: Read Only

overallServiceHealth
The health of this domain. This health status is aggregated across all of the running servers and will denote the most severe health condition encountered.

Type: Health

Constraints: Read Only