Oracle ILOM Web Service REST API

F48379-03
October 2022

Oracle ILOM Web Service REST API,

F48379-03

Primary Author: Cheryl Smith

Contents

1 Using This Documentation

- **Overview** – Provides getting started information for accessing and using the Oracle Integrated Lights Out Manager (ILOM) Web Service Representational State Transfer (REST) application programming interface (API).
- **Audience** – Web application architects, developers, or administrators who need to perform the tasks or learn about the concepts in this guide.
- **Required knowledge** – Experience with web development, specifically HTTP and JSON, and REST API programming skills.

Product Documentation Library

Documentation and resources for this product and related products are available at Systems Management and Diagnostics Documentation.

Feedback

Provide feedback about this documentation at Oracle Feedback.

2 Getting Started With the Oracle ILOM REST API

The Oracle ILOM REST API is a secure RESTful web service that enables you to programmatically access and manage resources through the use of HTTPS. The Oracle ILOM resources are acted upon through a set of simple, well-defined operations. These are GET, DELETE, POST, and PATCH. The JSON data format is the primary media-type used for transfering the data representation to and from Oracle ILOM.

The topics in this section provide fundamental information about how to use the Oracle ILOM REST API.

- Prerequisites
- Access and Authentication
- Supported Operations
- Common Request Header Fields
- Server Responses to REST API Requests
- REST API Error Response
- Unsupported Oracle ILOM Server Capabilities

Prerequisites

Before using the Oracle ILOM REST API, consider the following information:

- **Required Oracle ILOM Firmware**

  The Oracle ILOM REST API is available on servers running Oracle ILOM firmware version 4.0 or later.

- **Required SSL Certificate**

  Oracle ILOM managed devices should be properly configured with a valid Web Server SSL Certificate (default provided self-signed or user installed CA signed). For further information about configuring and validating an SSL Certificate, see *Improve Security by Using a Trusted SSL Certificate and Private Key* in *Oracle ILOM Security Guide For Firmware Releases 3.x and 4.x*

- **Use of REST API Development Tools With Client Examples**

  To access the Oracle ILOM REST API, you can use any tool or programming library that can generate HTTP requests. For instance, the following tools can access the REST API in various capacities:

  – cURL – cURL is a command-line tool and library for transferring data using URL syntax. cURL is an open source software product that is available for download fromCurl Command Line Tool and Library

Note:

This guide includes a mix of HTTP request examples and cURL based examples. The purpose of these examples is to demonstate the syntax for creating client requests and then showing the applicable server responses.

– Web Browser With the Postman REST Client Plug-in Extension – A web browser with REST client plug-in extension enables you to retrieve the REST API data. However, it does not support modifying the REST API data.

– Java and Python HTTP Libraries – Languages such as Java and Python include HTTP libraries that you can use to interact with the REST application. To help users get started with using the Oracle ILOM REST API, client code samples for Java and Python are provided. See Appendix: Java Code Usage Sample for Oracle ILOM REST API Client and Appendix: Python Code Authentication Samples for Oracle ILOM REST API Client .

– Swagger UI tool – The Swagger UI is a user interface tool that enables you to view the REST API descriptions, as well as to modify the REST API data. You can download the Swagger UI tool from the swagger home page (Swagger API Development). For further details, see Appendix: Using Swagger UI to Access Oracle ILOM Web Service REST API.

Access and Authentication

The following topics in this section provide information about how to securely access resources within the Oracle ILOM REST API.

- Discovering Management Resources
- Client Access to REST API Management Resources
- Specifying Required Authentication Credentials in HTTP Requests

Discovering Management Resources

The Oracle ILOM REST API tree is a dynamic data structure of resources. Starting at the root resource (`/rest/v1`) the entire REST API is discoverable through uniform resource identifier (URI) links leading to other resources. As you can see in the following `GET` request example, the root resource does not represent a specific management resource. It does, however, represent a collection of URI links to the top-level management resources in Oracle ILOM.

```
GET /rest/v1
{
      "Target": "/rest/v1",
      "Targets": [
        {"name": "about", "uri": "/rest/v1/about"},
        {"name": "faults", "uri": "/rest/v1/faults"},
        {"name": "SYS", "uri": "/rest/v1/SYS"},
        {"name": "HOST", "uri": "/rest/v1/HOST"},
        {"name": "System", "uri": "/rest/v1/System"},
        {"name": "SP", "uri": "/rest/v1/SP"}
      ]
}
```

A machine-readable description of all the Oracle ILOM REST API resources, in a Swagger data model 2.0 format, is downloadable from the Oracle ILOM SP using the URL: `https://<ILOM SP IP address>:443/swagger.json`.

> **Note:**
>
> The Swagger model describes every available resource in the Oracle ILOM REST API, including the applicable HTTP methods, the media type, the request and response body formats, as well as the response status values. While this document (*Oracle ILOM Web Service REST API User's Guide*) provides some examples of requests and responses, the Swagger model is the definitive resource to obtain all details of requests and responses for all Oracle ILOM REST API resources. Tools such as the Swagger UI, provide a user-friendly browser-based presentation of the `swagger.json` file (also known as swagger model), which enables developers to execute and monitor API requests sent, as well as the results received. For more details about using the Swagger UI tool, see Appendix: Using Swagger UI to Access Oracle ILOM Web Service REST API.

Client Access to REST API Management Resources

Client access to management REST API resources are made through an HTTP request. Each client request sent to the server must include the following: HTTP verb, resource path, HTTP 1.1 version, and all required headers including the Host header. For example:

```
<HTTP verb> <resource_path> HTTP/1.1
<Header Name>: <Header Value>
```

Where:

- The *HTTP verb* indicates the type of operation to perform, for instance, `GET`, `POST`, `PATCH`, and `DELETE`.
- The *resource_path* portion of the URL always begins with `/rest/v<version>` followed by the target resource to be acted upon. For example: `/rest/v1/System`

  When constructing the *resource path* portion of the URL, follow these guidelines:

  – All resource path names are case-insensitive, with the exception of the entities under the `/SYS` resource. In this case, the `/SYS` resource and all its entities are case-sensitive.

  > **Note:**
  >
  > For a descriptive list of /SYS NAC names, refer to the Oracle Service Manual provided for your server.

  – Forward slash (/) characters can be included in the resource path portion of the URL to indicate a hierarchical relationship between resources (for example: */top-level_resource_name/sub-level_resource_name/property_name*).

  – Resource paths are not considered valid if they end with a trailing forward slash '/' character. For instance, the following resource path would be considered invalid: `/rest/v1/System/`

- The *Header Name* and *Header Value* passes additional information about the request to the server. For further details, see Common Request Header Fields.

Specifying Required Authentication Credentials in HTTP Requests

The Oracle ILOM REST API requires you to specify authentication credentials in each client HTTP request. These credentials enable the server to validate the identity of the user logging in or accessing resources. In cases where the server is unable to validate the identity of the user, the server will deny the user access and respond with an HTTP 401 Unauthorized error message.

To validate the identity of an Oracle ILOM user, the REST API supports the following methods:

- Basic Access Authentication
- Token-Based Authentication

Basic Access Authentication

Basic Access Authentication is the primary Oracle ILOM REST API client authentication method. When using this method, the HTTP request includes an Authorization header field to validate the user's identity. For example:

```
Authorization: BASIC encoded_credentials
```

Where *encoded_credentials* is the Base64 encoding of the user name and password joined by a colon.

Basic Access Authentication Example:

```
GET /System HTTP/1.1
Authorization: BASIC cm9vdDpjaGFuX2VtZQ==
Accept: application/json
```

Token-Based Authentication

Oracle ILOM supports Token-Based Authentication as an alternative REST API client authentication method. This method might offer some performance improvement when a rapid succession of multiple requests are targeting a single Oracle ILOM instance. When using this method, the HTTP request includes a unique token in the X-REST-Token header to validate the user's identity for each request. For example:

```
X-Rest-Token:<unique_token>
```

Where *"unique_token"* represents a signed token generated by the API, which is then used by the server to validate the user's identity. After a user's identity is successfully validated, the user remains logged in until the token is no longer needed or it expires.

Note:

Tokens are signed with a secret algorithm to protect against manipulation. To further protect the token from manipulation, an SSL certificate must be properly configured in Oracle ILOM. For configuration details, see *Improve Security by Using a Trusted SSL Certificate and Private Key* in *Oracle ILOM Security Guide For Firmware Releases 3.x and 4.x*

To better understand the process for generating a token and including the token in subsequent requests, see the following:

1. A user requests a token from the REST API by issuing a POST request to the `/rest/v<version>/login` resource

   Syntax:

   ```
   POST /rest/v1/login HTTP/1.1
   Authorization: BASIC <Base64 encoding of username:password>
   Accept: application/json
   ```

2. The server validates the user's credentials and returns the X-Rest-Token in the response to the POST request.

   Example response:

   ```
   {
    X-Rest-Token: LBiDHbTnrGeDJrLSDGaisLXIQMfVjo
   }
   ```

3. The user passes the token in all subsequent requests within the Oracle ILOM REST API specific "X-Rest-Token" header.

   Syntax:

   ```
                                 <GET|POST|PATCH|DELETE> <Resource_Path> HTTP/1.1
   X-Rest-Token: <token string>
   ```

   Subsequent request example:

   ```
   GET /System HTTP/1.1
   X-Rest-Token: LBiDHbTnrGeDJrLSDGaisLXIQMfVjo
   Accept: application/json
   ```

4. The server validates and matches the token in each subsequent request, authenticating the user for the given request.

   Note:

   The requested operation only proceeds if the server is able to validate and match the algorithm of the token. If the server is unable to verify the token, the server would respond with a 401 unauthorized message indicating that the request could not be processed as authorization could not be verified.

5. The token is destroyed when one of the following occurs:

- Token expires after 15 minutes of inactivity.

  -or-

- The user logs out.

  Syntax:

  ```
  POST /rest/v1/logout HTTP/1.1
  X-Rest-Token: <token string>
  ```

Supported Operations

When working with the Oracle ILOM REST API, you specify HTTP verbs to perform operations on resources. The following table describes the type of operations supported by the Oracle ILOM REST API.

Table 2-1 Supported REST API HTTP Verb Operations

Operation	Description
DELETE Resource `DELETE <resource>`	Removes the resource specified in the DELETE request. For further details, see Remove Resources Using DELETE Requests.
GET Resource `GET <resource>`	Returns resource data corresponding to the requested resource. For further details, see these topics: • Retrieve Resources Using GET Requests • Retrieve Log-Style Resources Using GET Requests
PATCH Resource `PATCH <resource>`	Modifies the resource specified in the PATCH request. For further details, see Modify Resources Using PATCH Requests,
POST Resource `POST` *<resource>*	Creates a resource that is added to the collection represented by the resource specified in the POST request. For further details, see Create Resources Using POST Requests.
POST Action `POST <resource>`	Performs actions on the resource that are outside the scope of the CRUD (Create/Read/Update/Delete) actions supported by HTTP verbs. For further details, see Perform Actions on Resources Using POST Requests.

Common Request Header Fields

The following table describes the common Oracle ILOM REST API request headers.

Table 2-2   Common Request Header Field Names

Request Header Field Names	Description	Required
Authorization *- or -* X-Rest-Token	Authentication header containing authentication data. Examples: `Authorization: BASIC` *`<credentials>`* `X-Rest-Token": ` *`<value>`* For further details, see Specifying Required Authentication Credentials in HTTP Requests	Yes (All requests)
Accept	Use the Accept header to specify the acceptable response media type. **Example**: If the acceptable media type is JSON, the Accept header would look like this: `Accept: application/json`	Yes (GET requests)
Content-Type	Use the Content-Type header to specify the request body media type. **Example**: If the request body media type is JSON, the Content-Type header would look like this: `Content-Type: application/json`	Yes (POST and PATCH request body)
Host	Use the Host header to specify the IP address or hostname of the SP server. Example: The host header would look like this: `Host: <host name/address>[:<port number>]` If a port number is not specified, the default port (443) number is used.	Yes (All requests)

Server Responses to REST API Requests

The Oracle ILOM server sends a response for every REST API request. Each response, at a minimum, includes an HTTP status code to indicate the success or failure of the requested operation. For most operations, the server returns a response body in JSON format, with the exception of the DELETE operation. In this case, a response to a DELETE operation is not returned because there is nothing to return.

For an example of a server response sent in JSON format, see the response body section of Retrieve Resources Using GET Requests.

REST API Error Response

When a request results in an error, the server returns an HTTP status code (3xx, 4xx; 5xx) that indicates the error type. For instance, if an incorrect verb for an operation is used, the server will respond with a 405 (Method Not Allowed) status code. The details describing the error are returned in the response body. These details can include the failure reason, the operation failure code, and a message describing the failure, for example:

```
{ "error": {
          "reason":"<reason for error>",
          "code":<ILOM failure code (integer)>,
          "message":"<message describing the failure code>"
          }
       }
```

Unsupported Oracle ILOM Server Capabilities

The following table identifies Oracle ILOM server capabilities that are not supported by the Oracle ILOM REST API.

Table 2-3 Unsupported Server Capabilities

Unsupported Server Capability	Notes
Oracle ILOM Restricted Shell Hardware Diagnostic Commands	Oracle ILOM restricted shell for hardware diagnostics is not supported by the Oracle ILOM REST API. Information on how to run hardware diagnostics from the Oracle ILOM CLI is provided in the *Oracle x86 Server's Diagnostic Guide*.
Oracle ILOM Fault Management Shell (fmdump, fmstat, fmoverride commands)	Oracle ILOM Fault Management Shell is not supported by the Oracle ILOM REST API. However, the Oracle ILOM Fault Management Shell features are configurable from the Oracle ILOM CLI For details, see *Oracle ILOM User's Guide for System Monitoring and Diagnostics Firmware Release 5.0.x*.
Oracle ILOM Service Shell	The Oracle ILOM Service Shell is not supported by the the Oracle ILOM REST API. This shell is restricted to Oracle service personnel use.

3 Using HTTP Methods to Perform Operations on Oracle ILOM Resources

The topics in this section provide examples of RESTful HTTP requests and responses for the following operations:

Retrieve Resources Using GET Requests

Use a GET request to retrieve information about an Oracle ILOM resource.

Note:

HTTP GET requests will only retrieve data and not have any other effect on the data.

Request Format

```
GET <Resource_Path> HTTP/1.1
<Header Name> : <Header Value>
```

Request Header Fields Required

The required request header fields are as follows: `Authorization`, `Accept`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Response Status Codes

- Success: HTTP Status = 200 OK
- Failure: HTTP Status = 4xx, 5xx

Response Body (Success Status)

When an entity corresponding to the requested resource is found, Oracle ILOM returns the details associated with the requested resource in the response body. In which case, the details in the response body can include: name|value pair properties, a collection of target resources, and a collection of non-CRUD actions supported for the target resource.

Note:

To identify the exact response body media type, refer to the Oracle ILOM Swagger Model (`swagger.json`) description. For more details, see Discovering Management Resources.

```
{
    "Target": "<target URI path>",
    "property1 name": "property1 value",
    ...
    "propertyn name": "propertyn value",
    "Targets": [
        {
            "name": "<name of target1>",
            "uri": "<URI path of target1>"
        },
        ...
        {
            "name": "<name of targetx>",
            "uri": "<URI path of targetx>"
        }
    ],
    "Actions": [
        {
            "name": "action1"
        },
        ...
        {
            "name": "actionx"
        }
    ]
}
```

Where:

- The `"Target"` property would define the resource path.
- The `"Property name:value"` pairs would define the property name and value.
- The `"Targets"`, if any, would define a collection of other accessible resources.
- The "`Actions`", if any, would define non-CRUD actions that are applicable to the resource.

Response Body (Failed Status)

When an entity corresponding to the requested resource is not found, the response body returned includes an HTTP failure status code, as well as details describing the failure. For more details, see REST API Error Response.

Example: HTTP Request

In the following HTTP example, a request is sent to the server SP to retrieve information about the `/System` resource.

```
GET /rest/v1/System HTTP/1.1
Accept: application/json
```

Upon successfully locating the `/System` resource, the Oracle ILOM REST API returns the `/System` resource details in a JSON response body format, for example:

```
{
    "Actions": [
        {
            "name": "power-on"
        },
        {
            "name": "power-off"
        },
        {
            "name": "soft-power-off"
        },
        {
            "name": "reset"
        }
    ],
    "Target": "/rest/v1/System",
    "Targets": [
        {
            "name": "Open_Problems",
            "uri": "/rest/v1/System/Open_Problems"
        },
        {
            "name": "Processors",
            "uri": "/rest/v1/System/Processors"
        },
        {
            "name": "Memory",
            "uri": "/rest/v1/System/Memory"
        },
        {
            "name": "Power",
            "uri": "/rest/v1/System/Power"
        },
        {
            "name": "Cooling",
            "uri": "/rest/v1/System/Cooling"
        },
        {
            "name": "Storage",
            "uri": "/rest/v1/System/Storage"
        },
        {
            "name": "Networking",
            "uri": "/rest/v1/System/Networking"
        },
        {
            "name": "PCI_Devices",
            "uri": "/rest/v1/System/PCI_Devices"
        },
        {
            "name": "Firmware",
            "uri": "/rest/v1/System/Firmware"
        },
        {
            "name": "BIOS",
            "uri": "/rest/v1/System/BIOS"
        },
        {
            "name": "Log",
            "uri": "/rest/v1/System/Log"
        }
    ],
    "actual_power_consumption": 20,
    "health": "Service Required",
    "health_details": "PS1 (Power Supply 1) is faulty.",
    "host_primary_mac_address": "<MAC address>",
    "ilom_address": "<IP address>",
    "ilom_mac_address": "<MAC address>",
    "locator_indicator": false,
    "model": "ORACLE SERVER X5-2L",
    "open_problems_count": 1,
    "part_number": "X5-2L-P1.LAST-20",
    "power_state": "Off",
    "primary_operating_system": "Not Available",
    "primary_operating_system_detail": "<details>",
    "qpart_id": "<QPART id>",
    "serial_number": "<serial number>",
    "system_fw_version": "<ILOM fw version>",
    "system_identifier": "",
    "type": "Rack Mount"
}
```

Retrieve Log-Style Resources Using GET Requests

Use a GET request to retrieve log style resources. Log style resource can include a collection of entries relating to open problems or events recorded in an Oracle ILOM log file.

Note:

Oracle ILOM logs include: Audit Log, Event Log, Session Log or System Log. Open problem entries relate to problems currently open on the system. The collection and properties available for each log style resource are described in the Swagger model. For more details about the Swagger model, see Discovering Management Resources.

Request Format

```
GET <Resource_Path>[?start=x&count=y] HTTP/1.1
<Header Name> : <Header Value>
```

- *Where*:
  - options for <*Resource_Path*> would be:
    - /SP/logs/event/list
    - /SP/logs/audit/list

> **Note:**
>
> As of Oracle ILOM firmware version 5.1.0, user session login and logout events for all Oracle ILOM user interfaces (web, CLI, and API) have been moved from the audit log to the session log.

> **Note:**
>
> User login and logout events captured in the audit log prior to Oracle ILOM 5.1.0 will continue to remain in the audit log.

* /SP/logs/session/list

> **Note:**
>
> The session log is only available In Oracle ILOM for connections opened on Oracle servers supporting Oracle ILOM 5.1.0 or later.

* /System/log/list
* /System/open_problems

> **Note:**
>
> The resource path for `/System/Open_Problems` does not have a `/list` at the end.

– `start=` *x* `&count=`*y* reflect the *optional* HTTP query parameters to indicate the number of log entries to be returned. For example, if there are 96 entries in the collection, to retrieve the 10 most recent entries (entry 87 to entry 96), the 'start' value would be 97 (even though such an entry does not exist) and the 'count' value would be 10).

> **Note:**
>
> The Open Problems log does not support the use of the start and count (`?start=` *x* `&count=` *y)* query parameters.

Request Header Fields Required

The required request header fields are as follows: `Authorization`, `Accept`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Response Status Codes

- Success: HTTP Status = 200 OK

- Failure: HTTP Status = 4xx, 5xx

Response Body (Success Status)

> **Note:**
>
> To identify the exact response body media type, refer to the Oracle ILOM Swagger Model (`swagger.json`) description. For more details, see Discovering Management Resources.

The general JSON response body for a GET request of log-style resources would look like:

```
{
  "collection name":
    [
      {
        "property1 name": "property1 value",
        ...
        "propertyn name": "propertyn value"
      },
      ...
      {
        "property1 name": "property1 value",
        ...
        "propertyn name": "propertyn value"
      }
    ]
}
```

Example: HTTP Request

In the following HTTP example, a request is sent to the server SP to retrieve a list of audit log entries.

```
GET /rest/v1/SP/logs/audit/list/?start=267&count=3 HTTP/1.1
Accept: application/json
```

The Oracle ILOM REST API returns a list of audit log entries in a JSON response body format, for example:

```
{
  "Target": "/rest/v1/SP/logs/audit/list",
  "Log": [
    {
      "id": "266",
      "datetime": "Wed Jan  9 20:43:47 2019",
      "class": "Audit",
      "type": "UI",
      "severity": "minor",
      "description": "root : Set : object = \"/SP/config/dump_uri\" : value =
\"Browser download\" : success"
    },
    {
      "id": "265",
      "datetime": "Wed Jan  9 20:40:12 2019",
      "class": "Audit",
      "type": "UI",
      "severity": "minor",
      "description": "root : Set : object = \"/System/BIOS/Config/dump_uri\" : value =
\"Browser download\" : success"
    },
    {
      "id": "264",
      "datetime": "Wed Jan  9 20:37:08 2019",
      "class": "Audit",
      "type": "UI",
      "severity": "minor",
      "description": "root : Open Session : object = \"/SP/sessions/27/type\" : value
= \"rest\" : success"
    }
  ]
}
```

Modify Resources Using PATCH Requests

Use a PATCH request to modify an Oracle ILOM resource.

Note:

A list of modifiable properties for each Oracle ILOM resource is available from the Swagger model under the `patch->responses->schema->properties` description section. For more details about the Swagger model, see Discovering Management Resources.

Request Format

```
PATCH <Resource_Path> HTTP/1.1
<Header Name> : <Header Value>

<Request Body>
```

- *Where*:
  - Request Body – Each HTTP PATCH request must specify a <request body> to indicate which property and value needs to be modified under the target resource.

Request Header Fields Required

The required request header fields are as follows: `Authorization`, `Content-Type`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Response Status Codes

- Success: HTTP Status = 200 OK, JSON formatted response body
- Failure: HTTP Status = 4xx, 5xx, JSON formatted error response body.

Response Body

Upon modifying the data sent by the client, the Oracle ILOM REST API returns the following response body for all PATCH requests:

```
{
  "code": <integer code>,
  "description": "description string"
}
```

The value assigned to <*integer code*> can be any of the following:

- 0: PATCH operation completed with no significant consequence
- 1: PATCH operation completed might cause the web server to reset.
- 2: PATCH operation completed will cause the Oracle ILOM network connection to be disabled.

Example: HTTP Request

In the following HTTP example, a PATCH request is sent to the server SP to modify the `system_contact` and `system_location` properties under the `/SP` resource.

```
PATCH /SP HTTP/1.1
Content-Type: application/json

{"system_contact":"Mr. Smith",
 "system_location":"Newtown, MA"}
```

Upon successfully modifying the system properties under the `/SP` resource, the Oracle ILOM REST API returns the following details in a JSON response body format.

```
{
  "code": 0,
    "description": "PATH Action Complete"
}
```

Create Resources Using POST Requests

Use a POST request to create an Oracle ILOM resource.

Note:

Resources supporting Post operations are listed in the Swagger model under the section `post->parameters->name`. Properties that can be specified for a resource during the creation operation are listed in the Swagger model under the section `post->parameters->schema->properties`. Required properties that must be specified for a resource during the creation operation are listed in Swagger model under the section `post->parameters->schema->required`. For more details about the Swagger model, see Discovering Management Resources.

Request Format

```
POST <Resource_Path> HTTP/1.1
<Header Name> : <Header Value>

<Request Body>
```

- *Where*:

  – Request Body – Each HTTP POST request must specify a <*request body*> that contains non-default name-value pairs for the resource properties to be created.

Request Header Fields Required

The required request header fields are as follows: `Authorization`, `Content-Type`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Response Status Codes

- Success: HTTP Status = 201 Created, JSON formatted response body
- Failure: HTTP Status = 4xx, 5xx, JSON formatted error response body.

Example: HTTP Request

In the following HTTP example, a request is sent to the server SP to create a new user resource with the specified name and password.

```
POST /SP/users HTTP/1.1
Content-Type: application/json

{"user_name":"jane",
 "password":"somepassword"}
```

Upon successful creation of the new resource, a JSON representation of the newly created user is returned in the response body.

```
{
  "Target":"/rest/v1/SP/users/jane",
  "user_name":"jane",
  "role":"o",
  "password":"*****",
  "locked":"false",
  "password_expiration":"no expiry",
  "Targets":[{
    "name":"ssh",
    "uri":"/rest/v1/SP/users/jane/ssh"
  }]
}
```

Remove Resources Using DELETE Requests

Use a DELETE request to remove an Oracle ILOM resource.

Request Format

```
DELETE <Resource_Path> HTTP/1.1
<Header Name> : <Header Value>
```

Request Header Fields Required

The required request header fields are as follows: `Authorization` and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Response Status Codes

- Success: HTTP Status = 204 No Content, no response body.
- Failure: HTTP Status = 4xx, 5xx, JSON formatted error response body.

Response Body (Failed Status)

When an entity corresponding to the requested resource is not found, the following error information is returned in a JSON response body format.

```
{
  "error": {
    "reason":"<failure reason>",
    "code":<ILOM failure code (integer)>,
    "message":"< message describing the failure code>"
  }
}
```

Example: HTTP Request

In the following HTTP example, a request is sent to the server SP to delete the resource 'jane'.

```
DELETE /SP/users/jane HTTP/1.1
```

Perform Actions on Resources Using POST Requests

Note:

Some operations on Oracle ILOM management entities cannot be matched to methods in an HTTP request as they do not fall into the CRUD (Create/Read/Update/Delete) scope. These operations are implemented by 'actions' in the Oracle ILOM REST API.

Use a POST request to perform non-CRUD actions on supported Oracle ILOM resources. Resources supporting non-CRUD actions will include an `Action []` array in the response body of a given GET request. The following types of non-CRUD actions are supported by the Oracle ILOM REST API:

- **System reset actions** – "reset" SP, "reset" host, "reset" certificates, "reset" keys, "reset" /HOST/diag/data
- **System power actions** – "soft-reset" (gracefully reset host), "power-on" (power-on the host), "power-off" (power-off the host), "soft-power-off" (soft power-off the host)
- **System service actions** – "prepare-to-remove" (prepare system for removal), "return-to-service" (return system to service), "start" or "stop" /HOST/diag tests, "version" (get firmware version information).
- **SYS fault management actions** – "acquit" (manually acquit problems on a FRU), "replace" (manually mark a fru fixed as a result of replacement) "repair" (mark a fru fixed), "clear" (manually clear fault associated with FRU or UUID)

Request Format

```
POST <Resource_Path> HTTP/1.1
<Header Name> : <Header Value>

{
  "op": "<action>"
}
```

The request body identifies the supported non-CRUD action of the resource.

Where:

- <*action*> identifies the supported resource *action* (such as `"reset"`, `"repair-to-remove"`, `"acquit"`, and so on).

> **Note:**
>
> The set of actions supported by any given resource is available in the 'Actions[]' array in the response to the GET request for that resource.

Request Header Fields Required

The required request header fields are as follows: `Authorization`, `Content-Type`, and `Host`.

> **Note:**
>
> For system service actions such as `op=version`, an `Accept` header must be specified in the request with a value of '`application/json`'.

For a description of these required header fields, see Common Request Header Fields.

Response Status Codes

- Success: HTTP Status = 200-OK *or* 204-No content

  > **Note:**
  >
  > A 200 status code is shown when a response body is included for the "action" specified.

  > **Note:**
  >
  > A 204 status code is shown when a response body is not included for the "action" specified.

- Failure: HTTP Status = 4xx, 5xx, JSON formatted error response body.

Response Body

A response body is not provided for non-CRUD actions, unless the specified action is for `"op"="version`." In this case, a response body similar to the following is returned from the server.

```
{
    "Firmware Version":"5.0.0.0",
    "Firmware Build Number":"127654",
    "Firmware Build Date":"Mon Oct 15 11:07:17 PDT 2018",
    "Firmware File System Version":"0.3.180406"
}
```

Example: HTTP Request

In the following HTTP example, a request is made to the server to reset the Oracle ILOM SP.

```
POST /SP HTTP/1.1
Content-Type: application/json

{"op":"reset"}
```

4

Sending REST API Requests to Perform Oracle ILOM Server Management Tasks

The following topics in this section describe the server management tasks supported by the Oracle ILOM REST API.

- Managing Oracle ILOM Firmware Updates
- Retrieving System FRU Information
- Managing System Hardware Faults
- Uploading and Downloading REST API Data
- Downloading Host Console History
- Downloading Snapshot Data

> **Note:**
>
> In some instances cURL examples, in addition to the HTTP request and response examples, are included in this chapter to demonstrate the use of the REST API management task.

Managing Oracle ILOM Firmware Updates

The Oracle ILOM REST API supports the following Oracle ILOM Firmware management tasks:

- Updating Oracle ILOM Firmware
- Storing and Managing Firmware Images
- Removing a Local Firmware Package Instance

Updating Oracle ILOM Firmware

To ensure that users have access to the latest Oracle ILOM features and product enhancements,all upgradable system devices should be updated with the latest Oracle ILOM firmware release.

> **Note:**
>
> In addition to using the Oracle ILOM REST API to update firmware on system devices, you can use the Oracle ILOM web interface or CLI. For further details, see *Updating Oracle ILOM Firmware* in *Oracle ILOM 5.1 Administrator’s Guide*

The process to update Oracle ILOM firmware image on a system device involves these three steps:

- Step 1: Upload the Local Firmware Package File

  > **Note:**
  >
  > For firmware download instructions, refer to *Oracle ILOM Firmware Versions and Download Methods* in *Oracle ILOM Feature Updates and Release Notes Firmware Release 5.1.x*

- Step 2: View and Answer the Firmware Configuration Questions.
- Step 3: Start the Firmware Installation and Poll the Status

Step 1: Upload the Local Firmware Package File

Use a POST request to upload a local firmware package file.

> **Note:**
>
> Firmware packages can take several minutes to upload depending on network speed.

HTTP Request Format:

```
POST /rest/v<version>/SP/firmware/update HTTP/1.1
<Header Name>: <value>

<Request body>
```

Where:

- The `<Request body>` contains the package file contents in a multi-part data form format.

Request Headers Required:

The following request headers are required to upload a local firmware package file.

- Accept header – The `Accept:` header must specify the value of `application/json` or a superset of `application/json`.
- Content-Type header – The `Content-Type:` header must specify the value of `multipart/form-data`.

  > **Note:**
  >
  > Some tools such as cURL provide the Content-Type and file contents when the user supplies the file path.

- Other headers– Authenication and Host headers are required. For a description of these headers, see Common Request Header Fields.

Response: Status Codes

- Success: HTTP Status = 200 OK
- Failure: HTTP Status = 4xx, 5xx

Response Body:

Note:

To identify the exact response body media type, refer to the Oracle ILOM Swagger Model (`swagger.json`) description. For more details, see Discovering Management Resources.

```
{
  "Target":"/rest/v<version>/SP/firmware/update/1",
  "Targets":[
    {
      "name":"questions",
      "uri":"/rest/v1/SP/firmware/update/1/questions"
    },
    {
      "name":"status",
      "uri":"/rest/v1/SP/firmware/update/1/status"
    },
    {
      "name":"versions",
      "uri":"/rest/v1/SP/firmware/update/1/versions"
    }
  ]
}
```

Where:

- The context number `/1` represents the first local firmware package uploaded to the server SP. For every new firmware package created on the server SP, the content number that the REST API assigns is incremented by one. For example, 1 is assigned to the first package created. If the first package is later removed and another package is created, the second package instance is then assigned 2.

  Note:

  As of Oracle ILOM firmware version 5.0, only one firmware update image can be up uploaded to the server SP at a given time. Earlier versions of Oracle ILOM firmware versions (4.0.1 and later 4.x.releases), supported up to three active firmware update instances at a given time.

- The `"questions"` are the same firmware update questions that are asked when using the Oracle ILOM CLI to perform the update. All answers to these questions must be answered either true or false. For information on how to view and provide answers to these questions, see Step 2: View and Answer the Firmware Configuration Questions.

Step 2: View and Answer the Firmware Configuration Questions

Use a GET and PATCH request to retrieve and modify the firmware configuration questions that were returned by the POST operation in Step 1: Upload the Local Firmware Package File .

HTTP Request Format: View Questions

The HTTP request to view the firmware configuration questions, would look like this:

```
GET /rest/v<version>/SP/firmware/update/1/questions HTTP/1.1
<Header Name>: <Value>
```

Request Headers Required

The request header fields required to retrieve resources are as follows: `Accept`, `Authorization`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

cURL Request Example: View Questions

Using cURL, the request to view the firmware configuration questions would look like this:

```
curl -k -u "root:changeme" -H "Accept:application/json" https://<IP addr>:443/
rest/v1/SP/firmware/update/1/questions
```

Response: Status Codes

- Success: HTTP Status = 200 OK
- Failure: HTTP Status = 4xx, 5xx

Response Body

Note:

To identify the exact response body media type, refer to the Oracle ILOM Swagger Model (`swagger.json`) description. For more details, see Discovering Management Resources.

```
{
  "Target": "/rest/v<version>/SP/firmware/update1/questions",
  "questions":
  [
    {
      "text": "Preserve existing SP configuration",
      "value": true
    },
    {
      "text": "Preserve existing BIOS configuration",
      "value": true
    },
    {
      "text": "Delay BIOS upgrade until next server poweroff or reset",
      "value": true
    }
  ]
}
```

HTTP Request Format: Answer Questions

The HTTP request to modify a firmware configuration question, would look like this:

```
PATCH /rest/v<version>/SP/firmware/update/1/questions HTTP/1.1
<Header Name>: <Value>

<Request Body>
```

The *<Request Body>* specifies the firmware configuration question and its value in the format described in the Swagger model.

Request Headers Required

The request header fields required to modify resources are as follows: `Content-Type`, `Authorization` , and `Host`.

For a description of these required header fields, see Common Request Header Fields.

cURL Request Example: Answer Questions

Using cURL, the request to modify a firmware question would look like this:

```
curl --request PATCH --data '{"questions":[{"text":"Preserve existing SP
configuration","value":false}]}' -v -k -v -u "root:changeme" -H "Accept:application/
json" https://IP addr:443/rest/v1/SP/firmware/update/1/questions
```

Response: Status Codes

- Success: HTTP Status = 200 OK
- Failure: HTTP Status = 4xx, 5xx

Response Body:

Note:

To identify the exact response body media type, refer to the Oracle ILOM Swagger Model (`swagger.json`) description. For more details, see Discovering Management Resources.

```
{
  "code": 0,
    "description": "PATH Action Complete"
}
```

Step 3: Start the Firmware Installation and Poll the Status

Use a PATCH request to start the firmware update. To poll the status of the firmware update while the installation is in process, use a GET request.

Note:

Upon the completion of the firmware update process, the SP will automatically reset and become temporarily non-responsive.

HTTP Request Format: Start Firmware Installation

```
PATCH /rest/v<version>/SP/firmware/update/1 HTTP/1.1
<Header Name>: <Value>

<Request body>
{
  "start": true
}
```

Where:

- The `<Request body>` specifies the JSON content.
- The `"start":true` initiates the configuration and installation of the firmware update package.

Request Headers Required

The request header fields required to modify resources are as follows: `Content-Type`, `Authorization`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

HTTP Request Format: Poll Installation Status

```
GET /rest/v<version>/SP/firmware/update/1/status HTTP/1.1
<Header Name>: <Value>
```

Request Headers Required

The request header fields required to retrieve resources are as follows: `Accept`, `Authorization`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Response: Status Codes

- Success: HTTP Status = 200 OK
- Failure: HTTP Status = 4xx, 5xx

Response Body:

Note:

To identify the exact response body media type, refer to the Oracle ILOM Swagger Model (`swagger.json`) description. For more details, see Discovering Management Resources.

```
{
  "Target":"/rest/v<version>/SP/firmware/update/1/status",
  "state":"In Progress",
  "result":"",
  "component":"uboot",
  "component_progress":"7 of 8"
}
```

Related Information

- Perform Actions on Resources Using POST Requests

Removing a Local Firmware Package Instance

Use a DELETE request to remove a firmware package instance from the server SP.

> **Note:**
>
> As of Oracle ILOM firmware version 5.0, the Oracle ILOM REST API only allows one firmware package image to be uploaded on the server at a time.

HTTP Request Format:

```
DELETE /rest/v<version>/SP/firmware/update/<instance> HTTP/1.1
<Header Name>: <Value>
```

Where:

- <*instance*> represents the context number that is assigned to the firmware update package file that is currently loaded on the server SP.

  > **Note:**
  >
  > As of Oracle ILOM firmware version 5.0, only one firmware update image can be up uploaded to the server SP at a given time. Earlier versions of Oracle ILOM firmware (such as 4.0.1 and later 4.x.releases) support up to three active update firmware instances at one time.

Request Headers Required

The request header fields required to delete resources are as follows: `Accept`, `Authorization`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Storing and Managing Firmware Images

Note:

The ability to manage multiple SP firmware images in Oracle ILOM is supported on newer Oracle server platforms such as SPARC T7 and later platforms; as well as X86 X7 and later platforms. This functionality is *not* supported on SPARC M-Series platforms.

The Oracle ILOM REST API, as of firmware version 5.0, supports the ability to load a backup image for deferred installation and view the backup image properties. Note that after uploading a backup image to the SP, it can be activated at a later time for installation through the Oracle ILOM web interface or CLI. For more details about performing these management tasks, see the following:

- Upload Backup Image for Deferred Installation - Using REST API
- View Backup Image Properties - Using REST API
- *Activate Firmware Backup Image for Immediate Installation* in *Oracle ILOM 5.1 Administrator's Guide*

Upload Backup Image for Deferred Installation - Using REST API

The process to upload a backup image for deferred installation involves the following three steps:

- Step 1: Upload a Backup Firmware Package to SP
- Step 2: Store Posted Backup Image As a Secondary Image
- Step 3: Verify Backup Operations Succeeded for Steps 1 and 2

Step 1: Upload a Backup Firmware Package to SP

Use a POST request to upload a firmware package to the SP as a backup image.

HTTP Request Format

```
POST /rest/v<version>/SP/firmware/backupimage/update HTTP/1.1
<Header Name>: <Value>

<request body>
```

Where:

- The `<Request body>` contains the package file contents in a multi-part data form format.

Note:

The backup image remains in a pending state until it is activated for installation. To active the backup image for immediate installation using the Oracle ILOM web interface or CLI, see *Activate Firmware Backup Image for Immediate Installation* in *Oracle ILOM 5.1 Administrator's Guide*.

Request Headers Required

The following request headers are required to post the backup firmware package on the SP.

- Accept header – The `Accept:` header must specify the value of `application/json` or a superset of `application/json`.
- Content-Type header – The `Content-Type:` header must specify the value of `multipart/form-data`.

  Note:

  Some tools such as cURL provide the Content-Type and file contents when the user supplies the file path.

- Other headers– Authentication and Host headers are required. For a description of these headers, see Common Request Header Fields.

Step 2: Store Posted Backup Image As a Secondary Image

Use a PATCH request to place the posted backup firmware image on the SP in the secondary storage bank.

HTTP Request Format

```
PATCH /rest/v<version>/SP/firmware/backupimage/update/1 HTTP/1.1
<Header Name>: <Value>

{
"start": true
}
```

Where:

- `"start": true` updates the secondary storage bank on the SP with the posted backup image.

Note:

The backup image remains in a pending state until it is activated for installation. To active the backup image for immediate installation using the Oracle ILOM web interface or CLI, see *Activate Firmware Backup Image for Immediate Installation* in *Oracle ILOM 5.1 Administrator's Guide*.

Request Headers Required

The request header fields required to update a resource are as follows: `Content-Type`, `Accept`, `Authorization`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Step 3: Verify Backup Operations Succeeded for Steps 1 and 2

Use a GET request to verify the backup operations for Steps 1 and 2 succeeded.

HTTP Request Format

```
GET /rest/v<version>/SP/firmware/backupimage/update/1/status HTTP/1.1
<Header Name>: <Value>
```

Note:

The backup image remains in a pending state until it is activated for installation. To active the backup image for immediate installation using the Oracle ILOM web interface or CLI, see *Activate Firmware Backup Image for Immediate Installation* in *Oracle ILOM 5.1 Administrator's Guide*.

Request Headers Required

The request header fields required to retrieve the status of the backup operations performed in Steps 1 and 2 are as follows: `Accept`, `Authorization`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Response: Status Codes

- Success: HTTP Status = 200 OK
- Failure: HTTP Status = 4xx, 5xx

Response Body:

Note:

To identify the exact response body media type, refer to the Oracle ILOM Swagger Model (`swagger.json`) description. For more details, see Discovering Management Resources.

```
{
  "Target":"/rest/v<version>/SP/firmware/backupimage/update/1/status",
  "state":"<value>",
  "result":"<value>",
  "component":"<value>",
  "component_progress":"value>"
}
```

View Backup Image Properties - Using REST API

Use a GET request to view the properties associated with the backup image.

HTTP Request Format

```
GET /rest/v<version>/SP/firmware/backupimage HTTP/1.1
<Header Name>: <Value>
```

Request Headers Required

The request header fields required to retrieve resources are as follows: `Accept`, `Authorization`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Response: Status Codes

- Success: HTTP Status = 200 OK
- Failure: HTTP Status = 4xx, 5xx

Response Body:

Note:

To identify the exact response body media type, refer to the Oracle ILOM Swagger Model (`swagger.json`) description. For more details, see Discovering Management Resources.

```
< HTTP/1.1 200 OK
{
  "Target":"/rest/v<version>/SP/firmware/backupimage",
  "build":"r129156",
  "date":"Tue Feb 19 13:05:36 PST 2019",
  "upload_date":"Wed Feb 20 16:44:21 2019",
  "version":"5.0.0.0"
}
```

Note:

The backup image remains in a pending state until it is activated for installation. To active the backup image for immediate installation using the Oracle ILOM web interface or CLI, see *Activate Firmware Backup Image for Immediate Installation* in *Oracle ILOM 5.1 Administrator's Guide*.

Retrieving System FRU Information

As of firmware release 4.0, the Oracle ILOM REST API supports the ability to retrieve information about Field Replacement Units (FRUs) currently installed on your system. All FRU information is accessible under the `/rest/v<version>/SYS` top level resource.

GET FRU Information

Use a GET request to retrieve FRU information.

Note:

Since FRUs can vary between platforms, the Swagger model does not support the ability to describe every FRU instance. It is assumed that the REST API client is aware of the FRU path names. In cases where the REST API client is not aware of the path name, but it is aware of the FRU name, the path name for that FRU instance is discoverable through the `Target` links appearing in the response.

HTTP Request Format:

```
GET /rest/v<version>/SYS/<FRU name>  HTTP/1.1
<Header Name> : <Header Value>
```

Where:

- SYS, the top level resource target, is always entered as upper-case. All resource path names under /SYS are always case-sensitive.

Request Header Fields Required

The required request header fields are as follows: `Authorization`, `Accept`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Response: FRU Resource Properties

The following list identifies the typical FRU response properties:

- `fru_type` – Identifies the type of field replacement unit.
- `fru_part_number` – Identifies the part_number assigned to the field replacement unit.
- `fru_serial_number` – When available, identifies the serial number assigned to the field replacement unit.
- `fru_rev_level` – When available, identifies the revision level assigned to the field replacement unit.
- `fru_manufacturer` – When available, identifies the manufacturer name for the field replacement unit:
- `fru_description` – When available, provides additional information about the field replacement unit.

Note:

Some FRU resources, including /SYS, are containers of FRU resources or other FRU containers. These type of resources might not include a list of all the FRU properties.

HTTP Example: Retrieve Disk Backplane Information

```
GET /rest/v1/SYS/DBP HTTP /1.1
<Header Name> : <Header Value>
```

Response: Status Codes

- Success: HTTP Status = 200 OK
- Failure: HTTP Status = 4xx, 5xx

HTTP Example: Response Body

Note:

To identify the exact response body media type, refer to the Oracle ILOM Swagger Model (`swagger.json`) description. For more details, see Discovering Management Resources.

```
{
    "fru_type":"Disk Backplane",
    "fru_part_number":"7077819",
    "fru_serial_number":"489089M+14336U0060",
    "fru_rev_level":"01",
    "fru_manufacturer":"MiTAC International Corporation",
    "fru_description":"ASM, BB, DISK CAGE,1",
    "Faults":[],
    "Actions":[
      {"name":"acquit"},
      {"name":"repair"},
      {"name":"replace"}],
    "Targets":[
      {"name":"HDD0", "uri":"/rest/v1/SYS/DBP/HDD0"},
      {"name":"SASEXP", "uri":"/rest/v1/SYS/DBP/SASEXP"}],
    "Target":"/rest/v1/SYS/DBP"
}
```

Managing System Hardware Faults

As of firmware release 5.0, the Oracle ILOM REST API supports the ability to manage system related hardware component faults.

Note:

A *fault* indicates that a hardware component is present but is unusable or degraded due one or more problems diagnosed by the Oracle ILOM. A faulted component is automatically disabled to prevent further damage to the system.

Note:

Alternatively, you can manage hardware component faults using the Oracle ILOM Fault Management Shell. For more information about the Fault Management Shell, see *Managing Oracle Hardware Faults Through the Oracle ILOM Fault Management Shell* in *Oracle ILOM Users Guide for System Monitoring and Diagnostics Firmware Release 5.1.x*.

Retrieve a List of System Fault Records

Use a GET request to retrieve a list of records under the `/rest/v<version>/faults` resource target. Note that each fault record is assigned a universally unique identifier (UUID), which you can choose to query for additional fault information.

HTTP Request Format

```
GET /rest/v<version>/faults  HTTP/1.1
<Header Name> : <Header Value>
```

Request Header Fields Required

The required request header fields are as follows: `Authorization`, `Accept`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Response: Status Codes

- Success: HTTP Status = 200 OK
- Failure: HTTP Status = 4xx, 5xx

Response Body

Note:

To identify the exact response body media type, refer to the Oracle ILOM Swagger Model (`swagger.json`) description. For more details, see Discovering Management Resources.

```
{
  "Target": "/rest/v<version>/faults",
  "Targets": [
    {
      "name": "<UUID>",
      "uri": "/rest/v1/faults/<UUID>"
    }
  ]
}
```

Where:

- `Targets[… ]`' contains references to the individual fault events.
- *UUID* identifies the universally unique identifier that is assigned to a fault event.

Retrieve Additional Information About a Specific Fault Record

Use a GET request to retrieve additional information about a fault record.

HTTP Request Format:

```
GET /rest/v<version>/faults/<UUID>  HTTP/1.1
<Header Name> : <Header Value>
```

Request Header Fields Required

The required request header fields are as follows: `Authorization`, `Accept`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Response: Status Codes

- Success: HTTP Status = 200 OK
- Failure: HTTP Status = 4xx, 5xx

Response Body

Note:

To identify the exact response body media type, refer to the Oracle ILOM Swagger Model (`swagger.json`) description. For more details, see Discovering Management Resources.

```
{
  "Target": "/rest/v<version>/faults/<UUID>",
  "id": "<UUID>",
  "time": "<timestamp>",
  "msgid": "<id>",
  "severity": "<level assigned>",
  "status": "<level assigned>",
  "diag_engine": "fdd",
  "system_manufacturer": "Oracle Corporation",
  "system_name": "ORACLE SERVER <model number>",
  "system_part_number": "<p/n>",
  "system_serial_number": "<serial number>",
  "system_firmware_manufacturer": "Oracle Corporation",
  "system_firmware_version": "(ILOM)5.0.0.0",
  "system_firmware_release": "(ILOM)2018.08.14",
  "Actions": [
    {
      "name": "acquit"
    },
    {
      "name": "clear"
    }
  ],
  "Targets": [
    {
      "name": "<suspects>",
      "uri": "/rest/v1/faults/<UUID>/suspects"
    }
  ]
}
```

Where:

- *UUID* identifies a numeric string (universally unique identifier (UUID)) that is assigned to fault record.
- `"time": "`*<timestamp>*" identifies the date and time when the problem was detected.
- `Actions[… ]` contains references to names of supported operations.
- `"severity": "`*<level assigned>* identifies the severity of the fault record. Examples include: Debug, Down, Critical, Major, and Minor.
- `"status": "`*<level assigned>*" identifies the status of the fault record, for example, Open.
- `"name": "`*<suspects>*" identifies the suspect faulty hardware component(s) causing the problem.

Retrieve a List of Fault Record Suspects

Use a GET request to retrieve a list of fault suspects associated with a specific fault record.

HTTP Request Format:

```
GET /rest/v<version>/faults/<fault_record_UUID>/suspects HTTP/1.1
<Header Name> : <Header Value>
```

Request Header Fields Required

The required request header fields are as follows: `Authorization`, `Accept`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Response: Status Codes

- Success: HTTP Status = 200 OK
- Failure: HTTP Status = 4xx, 5xx

Response Body:

```
{
  "Target": "/rest/v<version>/faults/<fault_record_UUID>/suspects",
  "Targets": [
    {
      "name": "<suspect_UUID>",
      "uri": "/rest/v<version>/faults/<fault_record_UUID>/suspects/<suspect_UUID>"
    }
  ]
}
```

Where:

- *fault_record_UUID* identifies a numeric string (universally unique identifier (UUID)) that is assigned to a specific fault record.
- *suspect_UUID* identifies a numeric string (universally unique identifier (UUID)) that is assigned to a suspect faulty hardware component.

Retrieve Additional Information About a Fault Record Suspect

Use a GET request to retrieve additional information about a fault record suspect.

HTTP Request Format:

```
GET /rest/v<version>/faults/<fault_record_UUID>/suspects/<suspect_UUID>  HTTP/1.1
<Header Name> : <Header Value>
```

Request Header Fields Required

The required request header fields are as follows: `Authorization`, `Accept`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Response: Status Codes

- Success: HTTP Status = 200 OK
- Failure: HTTP Status = 4xx, 5xx

Response Body:

```
{
  "Target": "/rest/v<version>/faults/<fault_record_UUID>/suspects/<suspect_UUID>",
  "id": "<fault_record_UUID>",
  "class": "<name_of_suspected component failure>",
  "certainty_percentage": <percentage_value>,
  "affects": "/SYS/<resource_path>",
  "status": "<level_assigned>",
  "description": "<Message_describing_the_problem>.",
  "response": "<Message-describing _future operation of system>.",
  "impact": "<Message describing impact to system component>.",
  "action": "<Message_describing_action_to_resolve _the_problem.>",
  "Actions": [
    {
      "name":"clear"
    }
  ]
}
```

Retrieve Fault Records Associated With a Specific FRU

Use a GET request to retrieve a list of fault suspects associated with a specific fault record.

HTTP Request Format:

```
GET  /rest/v<version>/SYS/<FRU_resource_name> HTTP/1.1
<Header Name> : <Header Value>
```

Where:

- *FRU_resource_name* identifies the name of the resource target. For example: `/rest/v1/SYS/PS1`

Request Header Fields Required

The required request header fields are as follows: `Authorization`, `Accept`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Response: Status Codes

- Success: HTTP Status = 200 OK
- Failure: HTTP Status = 4xx, 5xx

Response Body

```
{
  "fru_type": "FRU_name",
  "fru_part_number": "<value>",
  "fru_serial_number": "<value>",
  "fru_rev_level": "<value>",
  "fru_manufacturer": "<value>",
  "fru_description": "<value>",
  "Faults": [
    {
      "name": "<Fault_record_UUID>",
      "uri": "/rest/v<version>/faults/<Fault_record_UUID>"
    }
  ],
  "Actions": [
    {
      "name": "acquit"
    },
    {
      "name": "repair"
    },
    {
      "name": "replace"
    }
  ],
  "Targets":[],
  "Target":"/rest/v<version>/SYS/PS1"
}
```

Where:

- *fault_record_UUID* identifies a numeric string (universally unique identifier (UUID)) that is assigned to a specific fault record.
- *suspect_UUID* identifies a numeric string (universally unique identifier (UUID)) that is assigned to a suspect faulty hardware component.

Manually Clear Fault for Undetected Replaced or Repaired Components

Use a POST request to clear a fault record associated with an undetected hardware repair or replacement. For more details, see:

- Clear or Acquit an Active Fault Record
- Clear an Active Fault Associated With a FRU
- Clear an Active Fault Suspect Record

Clear or Acquit an Active Fault Record

Use a POST request to clear or acquit an active fault record appearing under the `/rest/v<version>/faults` resource target.

HTTP Request Format:

```
POST /rest/v<version>/faults/<fault_record_UUID> HTTP/1.1
<Header Name> : <Header Value>

Request Data:
{"op": "clear"}
or
{"op": "acquit"}
```

Where:

- *{"op": "clear "}* is the action used when a fault event or uuid should no longer exist.

  Note:

  In Oracle ILOM, "clear" is equivalent to "repaired."

- *{"op"acquit"}* is the action used when a suspect component is not the cause of the problem.

Request Header Fields Required

The required request header fields are as follows: `Authorization`, `Content-Type`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Related Information

- Perform Actions on Resources Using POST Requests

Clear an Active Fault Associated With a FRU

Use a POST request to clear an active fault associated with FRU.

HTTP Request Format:

```
 POST /rest/v<version>/SYS/<FRU_resource_name> HTTP/1.1
<Header Name> : <Header Value>

Request data:
{"op": "repair"}
                        {"op": "replace"}
                        {"op": "acquit"}
```

Where:

- *{"op": "replace"}* is the action used when a suspect component has been replaced or removed.
- *{"op":"repair"}* is the action used when a suspect component has been physically repaired to resolve the reported problem. For example, a component has been reseated or a bent pin has been fixed.
- *{"op"acquit"}* is the action used when a suspect component is not the cause of the problem.

Request Header Fields Required

The required request header fields are as follows: `Authorization`, `Content-Type`, and `Host`.

For a description of these required header fields, see Common Request Header Fields.

Related Information

- Perform Actions on Resources Using POST Requests

Clear an Active Fault Suspect Record

Use a POST request to clear an active suspect faulty component.

HTTP Request Format:

```
POST /rest/v<version>/faults/<fault_record_UUID>/<suspects/suspect_UUID> HTTP/1.1
<Header Name> : <Header Value>

Request Data:
{"op": "clear"}
```

Where:

- *{"op"clear"}* is the action used when a fault event or uuid should no longer exist.

Note:

In Oracle ILOM, "`clear`" is equivalent to "`repaired`."

Related Information

- Perform Actions on Resources Using POST Requests

Uploading and Downloading REST API Data

As of Oracle ILOM firmware version 5.0, the Oracle ILOM REST API supports the ability to upload data to Oracle ILOM from a client-local location, as well as download data from Oracle ILOM to a client-local location. For more details, see:

- Upload REST API Data to Oracle ILOM From Local Client
- Download REST API Data From Oracle ILOM to Local Client

Upload REST API Data to Oracle ILOM From Local Client

Use a POST request to upload REST API data to Oracle ILOM from a local client.

HTTP Request Format

```
POST /rest/v<version>/<content_resource_path> HTTP/1.1
Content-Type: multipart/form-data
```

Where:

- *<content_resource_path>* specifies the resource path of the content to be uploaded that ends in `/content`. For example, to upload a banner message that appears when connecting to Oracle ILOM, the content resource path would look like this: `/rest/v1/SP/preferences/banner/connect/content`

Request Header Fields Required

The required request header fields are as follows: `Authorization`, `Content-Type`, and `Host`.

Note:

The Content-Type request header must specify a `multipart/form-data` for the target media type.

For a description of these required header fields, see Common Request Header Fields.

cURL Request Example: Upload Content for Connect Banner

Using cURL, a POST request to upload content to the connect banner resource would look like this:

```
curl -v -k -u "root:changeme" --request POST -F
"datafile=@/home/xyz/banner.txt" https://<IP
address>:443/rest/v1/SP/preferences/banner/connect/content
```

Note:

cURL adds the Content-Type header value of multipart/form-data when a `datafile` value is specified.

Download REST API Data From Oracle ILOM to Local Client

Use a GET request to download REST API data from Oracle ILOM to a local client.

HTTP Request Format

```
GET /rest/v<version>/SP/<content_resource_path> HTTP/1.1
Accept: <media type>
```

Where:

- <*content_resource_path*> specifies the resource path of the content to be downloaded. For example, to download the banner message that appears when connecting to Oracle ILOM, the content resource path would look like this: `/rest/v1/SP/preferences/banner/connect/content`
- <`media type`> represents the media type that is associated with the resource content to be downloaded. For a list of download 'content' resources and their associated media types, see the following table.

Download Resource	Media Type
/SP/preferences/banner/connect/content	text/plain
/SP/preferences/banner/login/content	text/plain
/SP/config/content	text/xml
/SP/services/snmp/mibs/content	application/zip
/System/bios/config/content	text/xml
/HOST/console/history/content	application/octet-stream

Request Header Fields Required:

The required request header fields are as follows: `Authorization`, `Accept`, and `Host`.

Note:

The request `Accept` header must specify the media type expected from the target resource.

For a description of these required header fields, see Common Request Header Fields.

Response: Status Codes

- Success: HTTP Status = 200 OK, <media type> response body
- Failure: HTTP Status = 4xx, 5xx, JSON formatted error response body

- **Note:**

  The response media types are specified in the Swagger model.

cURL Request Example

Using cURL, a request to download the connect banner to a binary file (`sp_config.xml`) would look like this:

```
curl -k -v -u "root:changeme" -H
"Accept:text/xml"
https://<IPaddress>:443/rest/v1/SP/config/content -o sp_config.xml
```

Downloading Host Console History

To extract the entire console history to a binary (non-text) file using the Oracle ILOM REST API, see one of the following topics

- Using a PATCH Request to Download Host Console History (Prior to firmware version 5.0)

  - or -
- Using a GET Request to Download Host Console History (As of firmware version 5.0 and later)

> **Note:**
>
> The ability to download the console history is not supported on multi-domain platforms such as SPARC M-Series servers.

Using a PATCH Request to Download Host Console History (Prior to firmware version 5.0)

The Oracle ILOM REST API, prior to firmware version 5.0, support the ability to use a PATCH request to extract the entire console history to a binary (non-text) file.

HTTP Request Format: PATCH

```
PATCH /rest/v<version>/HOST/console/history
<Header Name>: <value>

{
"dump": true
}
```

Where:

- `"dump":true` initiates the extraction process of the host console history data.

Request Headers Required

The request header fields required to modify resources are as follows: `Content-Type`, `Accept`, `Authorization`, and `Host`.

Note:

The request `Accept` header must specify `application/octet-steam` as the target media type. The `Content-Type` request header must specify `application/json` as the target media type.

For a description of these required header fields, see Common Request Header Fields.

cURL Example

Using cURL, a request to extract the host console history to a binary (non-text) file (`console_History.log`) would look like this:

```
curl -k --request PATCH -u "root:changeme" -H "Content-Type:application/json"
--data '{"dump":true}' https://<IPaddress>:443/rest/v1/HOST/console/history >
console_history.log
```

Using a GET Request to Download Host Console History (As of firmware version 5.0 and later)

The Oracle ILOM REST API, as of firmware version 5.0, supports the ability to use a GET request to extract the entire console history to a binary (non-text) file.

HTTP Request Format:

```
GET /rest/v<version>/HOST/console/history HTTP/1.1
<Header Name>: <value>
```

Request Headers Required

The request header fields required to retrieve resources are as follows: `Accept`, `Authorization`, and `Host`.

Note:

The request `Accept` header must specify `application/octet-steam` as the target media type.

For a description of these required header fields, see Common Request Header Fields.

cURL Request Example

Using cURL, a request to extract the host console history to a binary (non-text) file (`console_history.log`) would look like this:

```
curl -k --request GET -u "root:changeme" https://<IPaddress>:443/rest/v1/HOST/
console/history/content >
console_history.log
```

Downloading Snapshot Data

The Oracle REST API, as of firmware version 5.0, supports the ability to use a POST request to download snapshot data from the `/SP/diag/snapshot/content` resource.

> **Note:**
>
> A POST request is necessary rather than a GET request as certain request data is required to complete the download operation of the snapshot data.

HTTP Request Format:

```
POST /rest/v<version>/SP/diag/snapshot/content HTTP/1.1
<Header Name>: <value>

Request body:
{
"dataset": "<dataset_value>",
"encryption_passphrase": "<passphrase>"
}
```

Where:

- `"dataset"`: is mandatory and "<*dataset_value*>" can be one of "normal","normal-logonly", "fruid", "fruid-logonly", "full", "full-logonly".
- `"encryption_passphrase"`:"*passphrase*" is optional and is only required when the snapshot contents must be encrypted.

Request Headers Required:

The request header fields required to modify resources are as follows: `Content-Type`, `Accept`, `Authorization`, and `Host`.

> **Note:**
>
> The request `Accept` header must specify `application/octet-stream` as the target media type. The `Content-Type` request header must specify `application/json` as the target media type.

For a description of these required header fields, see Common Request Header Fields.

Response: Status Codes

- Success: HTTP Status = 200 OK, `application/` POST
- Failure: HTTP Status = 4xx, 5xx, JSON formatted error response body
- > **Note:**
  >
  > The response media types are specified in the Swagger model.

cURL Example

Using cURL, a request to extract the snapshot data to a data (non-text) file (`snapshot.dat`) would look like this:

```
curl --request POST -H "Content-Type:application/json" --data
'{"encryption_passphrase":"foobar","dataset":"full"}' -u
"root:changeme" -k -v https://<IP
address>:443/rest/v1/SP/diag/snapshot/content -o snapshot.dat
```

Related Information

- Perform Actions on Resources Using POST Requests

5

Appendix: Using Swagger UI to Access Oracle ILOM Web Service REST API

How To Access Oracle ILOM Web Service API Using Swagger

A machine readable description of the Oracle ILOM REST API is available in Swagger format. This description of the Web Service REST API includes information on every resource available under the five top-level resources: `/About`, `/System`, `/SP`, `/SYS`, and `/Host`. The applicable HTTP verbs, resource property types, configurable properties, and a description string for each available property are available in the Swagger description

You can obtain a Swagger description of the Oracle ILOM Web Service REST API by issuing a URL to the `swagger.json` file on the managed SP device, for instance:

```
https://<target_SP_IP_address>/swagger.json.
```

The following instructions explain how to install the Swagger UI tool and use it to access the `swagger.json` file on the SP:

1. Extract the Swagger UI zip file (downloaded from http://swagger.io) to a directory on the local file system. For example:

   ```
   /C:/Users/your_username/Downloads/
   ```

2. Open a web browser and enter the file path to the Swagger UI tool. For example:

   ```
   file:///C:/Users/your_username/Downloads/swagger-
   ui-master/dist/index.html
   ```

   The Swagger UI appears in the web browser.

3. In the Swagger UI search box, enter the URI to the swagger.json file. For example:

   ```
   https://SP _IP_address:443/swagger.json
   ```

   Where: *SP_IP_address* represents the IP address of the target SP device.

   If the `swagger.json` appears in the web browser, proceed to **Step 5**. Otherwise, if the `swagger.json` file fails to load and the following message appears, proceed to **Step 4** for troubleshooting instructions.

   ```
   Can't read from Service. It may not have the appropriate access-control-origin
   settings.
   ```

4. (Optional Troubleshooting Step) If the swagger.json file failed to display in Step 3, follow these steps:

   > **Note:**
   >
   > The web browser might have not accepted the web server certificate on the SP.

   a. Open another web browser window and enter the following URL:

      ```
      https://SP_IP_address/swagger.json
      ```

   b. Follow the instruction on the browser dialogs to add an exception for the SP.

      The `swagger.json` file appears in the browser window.

   c. Return to Swagger UI browser window (described in Step 3) and click the Explore button.

      The `swagger.json` file appears.

5. To view the full swagger description of the Oracle ILOM Web Service REST API, click the Expand Operations link.

6. To use the Swagger UI tool to access and modify Oracle ILOM resources, do the following:

   a. Click the Authorize button.

      A prompt appears prompting you to enter a generated X-REST-Token data value.

   b. Enter the X-Rest-Token data value that was generated from your user credentials, as described in the Token-Based Authentication.

      At the time of login, the web service uses the token data value to authenticate your Oracle ILOM user credentials on the target SP device.

6

Appendix: Python Code Authentication Samples for Oracle ILOM REST API Client

To help users get started with using the Oracle ILOM Web Service REST API, refer to the following Python client code samples for token-based authentication and HTTP basic authentication:

- ilom-rest-token.py
- ilom-rest-basic-auth.py

ilom-rest-token.py

```
# Copyright (c) 2018, Oracle and/or its affiliates. All rights reserved.
# example ILOM REST Client using authentication token over several requests
# tested with Python 2.7, 3.6

import requests
from requests.packages import urllib3

urllib3.disable_warnings(urllib3.exceptions.SubjectAltNameWarning)

AUTH_TOKEN_NAME = 'X-Rest-Token'
REST_URI = 'https://<SP host or IP>/rest/v1'

session = requests.Session()
session.verify = '<path to ILOM Certificate'  # set to True if you have a CA signed
cert
session.hooks = {
   'response': lambda r, *args, **kwargs: r.raise_for_status()
}

# login, retrieve auth token, store in session so it's included in all requests
response = session.post(REST_URI + '/login', auth=('root', 'changeme'))
login = response.json()
session.headers.update({AUTH_TOKEN_NAME: login.get(AUTH_TOKEN_NAME)})

try:
    # ilom command to retrieve version
    response = session.post(REST_URI + '/SP',json={'op': 'version'})
    print ('Version {}'.format(response.json().get('Firmware Version')))

    # add a new user Elwood
    session.post(REST_URI + '/SP/users', json={'user_name': 'Elwood', 'password':
'changeme'})

    # delete user elwood
    session.delete(REST_URI + '/SP/users/Elwood')

    # set the /SP system_location property to Santa Clara
    session.patch(REST_URI + '/SP', json={ 'system_location': 'Santa Clara'})
```

```
finally:
    session.post(REST_URI + '/logout')
```

ilom-rest-basic-auth.py

```
# Copyright (c) 2018, Oracle and/or its affiliates. All rights reserved.
# simple ILOM rest Client basic auth example for a single request to print the
system model / serial number
# tested with Python 2.7, 3.6

import requests
from requests.packages import urllib3

urllib3.disable_warnings(urllib3.exceptions.SubjectAltNameWarning)

response = requests.get('https://<ILOM host or IP>/rest/v1/System',
                        verify='<path to ILOM Certificate',  # set to True if
you have a CA signed cert
                        auth=('root', 'changeme'))
response.raise_for_status()
system = response.json()
print (system.get('model'), system.get('serial_number'))
```

7

Appendix: Java Code Usage Sample for Oracle ILOM REST API Client

The following sample Java code is primarily intended to illustrate the use of the Oracle ILOM REST API. Therefore, the server IP address and resources are defined as constants and exception handling is omitted. The sample code requires the use of the Apache HTTP Client v4.5.6 library and the GSON v2.8.2 library.

Java Code Sample

Prerequisites:

- Install Apache HTTP Client (v4.5.6) and GSON (v2.8.2) libraries and their dependencies in your class path before attempting to build and run the provided sample code .
- Set the `SERVER_IP_ADDRESS` constant to the value of your Oracle ILOM SP IP address before compiling the sample code.

Note:

This Java sample code was tested on Java v1.8.

```
Copyright (c) 2018, Oracle and/or its affiliates. All rights reserved.

/
package com.oracle.ssm.ilomrestapi.client;

import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;

import java.util.ArrayList;
import java.util.List;
import javax.net.ssl.HostnameVerifier;
import javax.net.ssl.SSLContext;
import javax.net.ssl.SSLSession;

import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.entity.ContentType;
import org.apache.http.entity.StringEntity;
import org.apache.http.HeaderIterator;
import org.apache.http.HttpEntity;
import org.apache.http.NameValuePair;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPatch;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpDelete;
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.conn.ssl.TrustSelfSignedStrategy;
import org.apache.http.impl.client.BasicCredentialsProvider;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.ssl.SSLContexts;
import org.apache.http.util.EntityUtils;


public class IlomRestApiClient {

    public static final String SERVER_IP_ADDRESS = <IP_address>;
    public static final String SERVER_PORT = "443";
    public static final String ILOM_ROOT_PATH = "/rest/v1";
    public static final String ILOM_ROOT_URI = "https://" + SERVER_IP_ADDRESS +
":" + SERVER_PORT + ILOM_ROOT_PATH;
    public static final String URI_PATH_LOGIN = "/login";
    public static final String ILOM_URI_LOGIN = ILOM_ROOT_URI + URI_PATH_LOGIN;
    public static final String URI_PATH_LOGOUT = "/logout";
    public static final String ILOM_URI_LOGOUT = ILOM_ROOT_URI + URI_PATH_LOGOUT;
    public static final String URI_PATH_SYSTEM = "/System";
    public static final String ILOM_URI_SYSTEM = ILOM_ROOT_URI + URI_PATH_SYSTEM;
    public static final String URI_PATH_SP_USERS = "/SP/users";
    public static final String ILOM_URI_SP_USERS = ILOM_ROOT_URI +
URI_PATH_SP_USERS;
    public static final String URI_PATH_ELWOOD_USER = "/Elwood";
    public static final String ILOM_URI_ELWOOD_USER = ILOM_URI_SP_USERS +
URI_PATH_ELWOOD_USER;
    public static final String ACCEPT_HEADER = "Accept";
    public static final String CONTENT_TYPE_HEADER = "Content-Type";
    public static final String JSON_MEDIA_TYPE = "application/json";
    public static final String TOKEN_HEADER = "X-Rest-Token";

    private String token;

    private CloseableHttpClient getHttpClient(boolean useCreds) throws Exception
{
        SSLContext sslContext = SSLContexts.custom()
            .loadTrustMaterial(new TrustSelfSignedStrategy())
            .build();
        HostnameVerifier hostNameVerifier = new HostnameVerifier() {
            public boolean verify(String hostname, SSLSession session) {
                return true;
            }
        };
        SSLConnectionSocketFactory socketFactory =
            new SSLConnectionSocketFactory(sslContext,
               new String[] { "TLSv1.2" }, null,
               hostNameVerifier);
        if (useCreds) {
            CredentialsProvider credsProvider = new BasicCredentialsProvider();
            credsProvider.setCredentials(
                   new AuthScope(SERVER_IP_ADDRESS,
Integer.parseInt(SERVER_PORT)),
                   new UsernamePasswordCredentials("root", "changeme"));
            return HttpClients.custom()
                .setDefaultCredentialsProvider(credsProvider)
                .setSSLSocketFactory(socketFactory)
                .build();
        } else {
            return HttpClients.custom()
                .setSSLSocketFactory(socketFactory)
                .build();
        }
    }
```