Oracle® Transportation Management

REST API Getting Started Guide

Release 6.4.3

Part No. E92131-01

December 2017

ORACLE

Copyright Notice

Contents

Tables

Send Us Your Comments

Oracle® Transportation Management REST API Getting Started Guide, Release 6.4.3

Part No. E92131-01

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: otm-doc_us@oracle.com

If you would like a reply, please give your name, address, telephone number, and electronic mail address (optional).

If you have problems with the software, contact Support at https://support.oracle.com or find the Support phone number for your region at https://www.oracle.com/support/contact.html.

Preface

Oracle Logistics Cloud provides multiple public REST APIs that can be used to access data stored in Logistics Cloud and construct integrations to other systems.

Change History

Date	Document Revision	Summary of Changes
08/2017	-01	Added “Using the REST API in the Cloud” section to the first chapter.
10/2017	-02	Added new UI information to “URI Format” chapter, and the “Authentication” chapter.

1. Quick Start

Set up your environment and create your first object using the REST API by performing the following tasks.

Prerequisites

Table 1-1: Prerequisites

Prerequisite	More Information
Have an OTM instance, either on-premise or on the Oracle Cloud.	Sign up to use Logistics Cloud.
Configure security endpoints	You must configure security endpoints for any users.

About OTM REST Service

OTM REST service intends to provide lightweight-data communications between clients and OTM.

Different from traditional web-page based data operations, OTM REST service accepts and generates data formatted in JSON. To data operations, OTM REST service enables clients to create, get, edit and delete data at entity level.

Each entity object could be located by a unique URI, which is explained in the URI Format section. For security reason, you should follow the OTM REST Authentication and Authorization instructions to safely access and operate data.

How to Access OTM REST

Here are typical ways to access OTM REST service:

- **Browser Address Bar**: You can send a GET request to the OTM REST server by putting the URI into the browser address bar. This approach may trigger a popup window for authentication information, and will display the response data onto the screen. For the most part, only GET methods will work for this approach.
- **curl**: You can use curl (terminal command tool/library) to access the OTM REST service. Currently, curl supports all PUT (create), GET (get), POST (update), and DELETE (delete) operations. For information about downloading and installing curl, see http://curl.haxx.se/download.html . You must install a version of curl that supports SSL.
- **CORS**: You can use JavaScript code that runs on browsers to access OTM REST service. Currently, CORS supports all PUT (create), GET (get), POST (update), and DELETE (delete) operations with particular CORS settings.
- **Programming Languages**: You can use programming language, such as JAVA, to access OTM REST service. Currently, most popular programming languages supports all PUT (create), GET (get), POST (update), and DELETE (delete) operations.

2. URI Format

URI stands for Uniform Resource Identifier. There are two ways to invoke the API.

When using Logistics Cloud, the first is protected by a SSO server.

```
http://host/GC3/api/sdo/entityName
```

The second is not protected by the SSO server and requires a staged integration user.

```
http://host/GC3/int-api/sdo/entityName
```

Where entityName is a valid OTM entity.

The REST API works at the top level entity level. All interaction with the REST API, even updating child records, will occur at the top level entity. For example, adding a reference number to an order release will still access the OrderRelease entity.

When issuing a GET, POST, or DELETE request you can identify the entity by sending the ID parameter on the query string of the URI. For example to GET information about the PHL location you can issue `http://host/GC3/api/sdo/Location?id=PHL`.

In order to get information about a resource to find out the fields, datatypes, and child records, you can access `http://host/GC3/api/metadata/entityName`. This will give information about the entity in XML format.

Get

Use this method to retrieve data for an entity. The ID parameter must be provided in this request.

The ID parameter is the GID of the related entity.

Put

Use this method to create a new item.

Post

Use this method to update an existing item. The ID parameter must be provided in this request.

The ID parameter is the GID of the related entity.

Delete

Use this method to delete a resource. It does not require a request body. The ID parameter must be provided in this request.

The ID parameter is the GID of the related entity.

3. Entities

The REST API works at the Entity level. Entities represent top level objects such as Shipment, Order Release, Location, and Invoice. To see the full list of supported APIs, append this URL to your server URL: GC3/api/RestApiList. For example, `http://yourserver.com:8080/GC3/api/RestApiList`. Entities encompass the entire hierarchy of the object. This includes the children, grandchildren, etc. of the top level object. For example the Shipment entity contains Reference Numbers, Remarks, and Stops. It also contains child elements of Stops such as Ship Units, and the Ship Unit's Lines. It is not possible to work directly with the children of an entity. All interaction with a child must occur through the parent entity.

4. Authentication

Authentication is run through HTTP authentication or through an SSO server.

HTTP Basic Authentication

A special authorization header is sent along with the request. User ID and Password are sent in a Base64 encoded format. Our REST APIs are protected by HTTP Basic authentication. We highly recommend that all communication is protected by HTTPS.

SSO Environment

The /GC3/api URI is protected by the SSO server. In this case access to the REST API should originate from the browser. This can be accomplished by creating a rich JavaScript application. Calls to the rest API can be made in the user's context when protected by an SSO server. This requires Cross Origin Resource Sharing (CORS) to be set up on the OTM server.

The /GC3/int-api URI is not protected by the SSO server. This URI requires the HTTP basic authentication authorization header. It also requires a staged integration user in OTM. The header you create is the Base64 encoded user name and password of the staged integration user.

5. Authorization

Individual REST APIs are protected by access control lists (ACL). Each API is associated with two ACLs: View and Edit. Users must be granted access to the ACLs before the APIs can be used. For more information on configuring ACLs, see the Security Guide in My Oracle Support Document 796594.1: Version 6.x Documentation and Training Resources.

For additional details such as specific roles to access a REST resource, refer to the following guides:

- Oracle Fusion Applications Understanding Security - http://docs.oracle.com/cd/E51367_01/common_op/OASCP/toc.htm
- Oracle Applications Cloud Security Reference for Common Features - http://docs.oracle.com/cd/E51367_01/common_op/OACSM/toc.htm

Enabling CORS Service

Due to the Same-Origin Policy (https://en.wikipedia.org/wiki/Same-origin_policy), web browsers prevent JavaScript code from making requests against a different origin. For example, JavaScript code in origin http://foo.A.com/ could not ask service from http://foo.B.com/, or even https://foo.A.com/ with normal request settings. Cross-Origin Resource Sharing (CORS) (https://www.w3.org/TR/cors/) is a technique that allows this type of communication. To enable CORS service, new settings are required on both server side and client side.

Server Setting

In order to enable CORS, server requires the following property to be set to the domain that will accept these requests.

- **glog.webserver.cors.origins**: This property defines the valid origin domains of clients that are accepted by the server. If there are multiple valid origin domains, please use "," to separate them. For example, "glog.webserver.cors.origins = http://localhost:port,https://localhost:port"

Client Setting

In order to distinguish CORS requests from normal REST requests, the client should contain the following headers.

- **CORS**: Server executes CORS actions only when it detects the header "CORS" in the request and the value of this header is true.
- **withCredentials**: Server requires and accepts cookie, so "withCredentials" should be set to true in the CORS request.
- **Authorization (For non-SSO server)**: Non-SSO server requires basic http authentication. Username and password should be encoded as the value of "Authorization" as one of the request headers.

JS Example at client side

```
var url = "http://host:port/GC3/api/sdo/Location?id=GUEST.4444";
var method = "GET";
var xhr = createCORSRequest(method, url);
xhr.send();

function createCORSRequest(method, url){
   var xhr = new XMLHttpRequest();

   if("withCredentials" in xhr){
          xhr.open(method, url, true);
          xhr.withCredentials = true;
          xhr.setRequestHeader("Authorization", "Basic **********");
          xhr.setRequestHeader("cors","true");
          xhr.onreadystatechange = function(){
                if(xhr.responseText !==null && xhr.responseText!== ""){
                      document.getElementById("d1").innerHTML =
xhr.responseText;
                }
          };

   // for IE 9 and older, which does not support "withCredentials", and thus
could not send request with cookie
   }else if(typeof XDomainRequest != "undefined"){
```



```
            xhr = new XDomainRequest();
            xhr.open(method, url, true);
            xhr.setRequestHeader("Authorization", "Basic **********");
            xhr.setRequestHeader("cors","true");
            xhr.onreadystatechange = function(){
                  if(xhr.responseText !==null && xhr.responseText!== ""){
                        document.getElementById("d1").innerHTML =
xhr.responseText;
                  }
            };
      }else{
            xhr = null;
      }

      return xhr;
}
```

6. Example - Get a Location

The following example shows how to use a REST API to get a location.

> **Note:** You must be granted the "REST - Location - View" Access Control List (ACL) to use this API.

If you want to retrieve a Location entity, modify the following example URL, which shows the method call to get a location:

```
curl -u <userName>:<password> [-w "%{http_code}\n"]
http://host:port/GC3/api/sdo/Location?id=<LOCATION_GID> [-o
</PATH/FileName.json>]
```

Explanation

Table 6-1

String	Definition
<>	Required field sign. Replace its content with real data, and remove the sign when you execute the command.
[]	Optional field sign. Remove the sign when you execute the command.
-u <userName>:<password>	User Credential
[-w "%{http_code}\n"]	Print status code
http://host:port/GC3/api/sdo/Location?id=<LOCATION_GID>	URI for working with Location entity
[-o </PATH/FileName.json>]	Where to store resulting JSON

Response

The process responds with various status codes, such as 200 OK and 400 Bad Request.

When executing this command the -o will indicate where this file will be stored on your file system. This file is a JSON representation of the Location entity. It includes all of the attributes related to Location as well as the child relationships. These child relations include Location Role Profile, Location Reference Number, and Location Service Provider Preference. Since Location has a 1-to-many relationship with the child relations, the children are represented by arrays in the resulting JSON.

```
{
    "Location":{
           "locationGid":"GUEST.LOCATION.AIR1",
           "locationXid":"AIR1",
           "locationName":"CRIS TEST",
           "city":"Philadelphia",
           "provinceCode":"PA",
           "postalCode":"19046",
           "countryCode3Gid":"USA",
```

```
"timeZoneGid":"EST",
"lat":80.0,
"lon":80.0,
"isTemporary":false,
"isMakeApptBeforePlan":false,
"isShipperKnown":true,
"isAddressValid":"N",
"slotTimeInterval":null,
"isLtlSplitable":true,
"useAppointmentPriority":false,
"scheduleLowPriorityAppoint":false,
"enforceTimeWindowAppoint":true,
"scheduleInfeasibleAppoint":true,
"bbIsNewStore":false,
"excludeFromRouteExecution":false,
"appointDisplay":null,
"appointDisplayStartTime":0,
"isTemplate":false,
"allowDriverRest":false,
"apptObjectType":"S",
"isFixedAddress":false,
"primaryAddressLineSeq":1,
"domainName":"GUEST",
"attributeDate1":null,
"attributeDate2":null,
"attributeDate3":null,
"attributeDate4":null,
"attributeDate5":null,
"attributeDate6":null,
"attributeDate7":null,
"attributeDate8":null,
"attributeDate9":null,
"attributeDate10":null,
"isActive":true,
"addressUpdateDate":{
       "date":"2014-08-12T15:36:25.000Z"
},
"updateDate":{
       "date":"2016-09-22T12:11:53.000Z"
},
"locationStatus":[
       {
              "transactionCode":"NP",
              "locationStatusBeanData":{
                     "locationGid":"GUEST.LOCATION.AIR1",
                     "statusTypeGid":"GUEST.CREDIT LEVEL",
                     "statusValueGid":"GUEST.CREDIT LEVEL_UNKNOWN",
                     "domainName":"GUEST"
              }
       }
       ],
       "locationRoleProfile":[
              {
                     "transactionCode":"NP",
                     "locationRoleProfileBeanData":{
                            "locationGid":"GUEST.LOCATION.AIR1",
                            "locationRoleGid":"XDOCK",
                            "xDockIsInboundBias":false,
```



```
                                    "fixedHandlingTime":null,
                                    "varHandlingTime":null,
                                    "createXdockHandlingShipment":true,
                                    "createPoolHandlingShipment":false,
                                    "varHandlingWeight":null,
                                    "maxFreightIdleTime":null,
                                    "domainName":"GUEST",
                                    "isAllowMixedFreight":false,
                                    "varHndcstWgt":null,
                                    "varHndcstWgtCost":null,
                                    "varHndcstVol":null,
                                    "varHndcstVolCost":null
                              }
                        }
                        ],
                        "locationRefnum":[
                              {
                                    "transactionCode":"NP",
                                    "locationRefnumBeanData":{
                                          "locationGid":"GUEST.LOCATION.AIR1",

"locationRefnumQualGid":"GUEST.ACCOUNT_NUMBER",
                                          "locationRefnumValue":"123456",
                                          "domainName":"GUEST"
                                    }
                              }
                              ],
                              "locationServprovPref":[
                                    {
                                          "transactionCode":"NP",
                                          "locationServprovPrefBeanData":{

"locationServprovPrefGid":"GUEST.PRE ME TOO",

"locationServprovPrefXid":"PRE ME TOO",

"locationGid":"GUEST.LOCATION.AIR1",
                                                      "servprovGid":"ATLC",
                                                      "domainName":"GUEST",
                                                      "locationServprovPrefD":[
                                                            {

"transactionCode":"NP",

"locationServprovPrefDBeanData":{

"locationServprovPrefGid":"GUEST.PRE ME TOO",

"locationServprovPrefDSeq":1486,

"dayOfWeek":1,

"begin":0,

"preferenceLevel":2,

"duration":86340,
```



```
    "isStanding":false,

    "domainName":"GUEST"
                                                                }
                                                        }
                                                        ]
                                                }
                                        },
                                        {
                                                "transactionCode":"NP",
                                                "locationServprovPrefBeanData":{

    "locationServprovPrefGid":"GUEST.6.3 TEST",

    "locationServprovPrefXid":"6.3 TEST",

    "locationGid":"GUEST.LOCATION.AIR1",
                                                        "serviceLocationGid":"NGLOS",
                                                        "domainName":"GUEST"
                                                }
                                        }
                                        ]
    }
}
```

7. Example - Create a Location

In this example you have a JSON file that contains new Location data. You want to send this JSON file to server with curl command to create a new record.

> **Note**: You must be granted the "REST - Location - Update" Access Control List (ACL) to use this API.

In order to create a location using the REST API an HTTP PUT should be executed on the Location URI. Below is an example of the JSON playload that should be sent to the server. The following curl command can be modified and used to create new Locations.

```
curl [-d @</PATH/FileName.json>] -H 'Content-Type:application/json' -u
<userName>:<password> [-w "%{http_code}\n"] -X PUT
http://<host>:<port>/GC3/api/sdo/Location
```

Explanation

Table 7-1

String	Definition
<>	Required field sign. Replace its content with real data, and remove the sign when you execute the command.
[]	Optional field sign. Remove the sign when you execute the command.
[-d @</PATH/FileName.json>]	Where to get the JSON file
-u <userName>:<password>	User credential
[-w "%{http_code}\n"]	Print status code
http://<host>:<port>/GC3/api/sdo/Location	URI for working with Location entity

Request

The flowing is an example of the JSON file. This file is a JSON representation of the Location entity. It includes all of the attributes related to Location as well as the child relationships. These child relations include Location Role Profile, Location Reference Number, and Location Service Provider Preference. Since Location has a 1-to-many relationship with the child relations, the children are represented by arrays in the resulting JSON.

```
{
   "Location":{
          "locationGid":"GUEST.LOCATION.AIR1",
          "locationXid":"AIR1",
          "locationName":"CRIS TEST",
          "city":"Philadelphia",
          "provinceCode":"PA",
          "postalCode":"19046",
          "countryCode3Gid":"USA",
          "timeZoneGid":"EST",
          "lat":80.0,
```

```
"lon":80.0,
"isTemporary":false,
"isMakeApptBeforePlan":false,
"isShipperKnown":true,
"isAddressValid":"N",
"slotTimeInterval":null,
"isLtlSplitable":true,
"useAppointmentPriority":false,
"scheduleLowPriorityAppoint":false,
"enforceTimeWindowAppoint":true,
"scheduleInfeasibleAppoint":true,
"bbIsNewStore":false,
"excludeFromRouteExecution":false,
"appointDisplay":null,
"appointDisplayStartTime":0,
"isTemplate":false,
"allowDriverRest":false,
"apptObjectType":"S",
"isFixedAddress":false,
"primaryAddressLineSeq":1,
"domainName":"GUEST",
"attributeDate1":null,
"attributeDate2":null,
"attributeDate3":null,
"attributeDate4":null,
"attributeDate5":null,
"attributeDate6":null,
"attributeDate7":null,
"attributeDate8":null,
"attributeDate9":null,
"attributeDate10":null,
"isActive":true,
"locationRoleProfile":[
      {
            "transactionCode":"I",
            "locationRoleProfileBeanData":{
                  "locationGid":"GUEST.LOCATION.AIR1",
                  "locationRoleGid":"XDOCK",
                  "xDockIsInboundBias":false,
                  "fixedHandlingTime":null,
                  "varHandlingTime":null,
                  "createXdockHandlingShipment":true,
                  "createPoolHandlingShipment":false,
                  "varHandlingWeight":null,
                  "maxFreightIdleTime":null,
                  "domainName":"GUEST",
                  "isAllowMixedFreight":false,
                  "varHndcstWgt":null,
                  "varHndcstWgtCost":null,
                  "varHndcstVol":null,
                  "varHndcstVolCost":null
            }
      }
      ],
      "locationRefnum":[
            {
                  "transactionCode":"I",
                  "locationRefnumBeanData":{
```



```
                                        "locationGid":"GUEST.LOCATION.AIR1",

    "locationRefnumQualGid":"GUEST.ACCOUNT_NUMBER",
                                        "locationRefnumValue":"123456",
                                        "domainName":"GUEST"
                                  }
                            }
                            ],
                            "locationServprovPref":[
                                  {
                                        "transactionCode":"I",
                                        "locationServprovPrefBeanData":{
                                              "locationServprovPrefGid":"GUEST.PRE
ME TOO",
                                              "locationServprovPrefXid":"PRE ME
TOO",
                                              "locationGid":"GUEST.LOCATION.AIR1",
                                              "servprovGid":"ATLC",
                                              "domainName":"GUEST",
                                              "locationServprovPrefD":[
                                                    {
                                                          "transactionCode":"I",

    "locationServprovPrefDBeanData":{

    "locationServprovPrefGid":"GUEST.PRE ME TOO",

    "locationServprovPrefDSeq":1486,
                                                                              "dayOfWeek":1,
                                                                              "begin":0,

    "preferenceLevel":2,

    "duration":86340,

    "isStanding":false,

    "domainName":"GUEST"
                                                          }
                                                    }
                                                    ]
                                        }
                                  }
                                  ]
    }
}
```

Response

The process responds with various status codes, such as 200 OK and 400 Bad Request.

8. Example - Update a Location

> **Note**: You must be granted the "REST - Location - Update" Access Control List (ACL) to use this API.

In order to update a location using the REST API an HTTP POST should be executed on the Location URI. Note that the location ID should be passed on the query string as the **id** parameter. Below is an example of the JSON playload that should be sent to the server. The following curl command can be modified and used to update locations.

```
curl [-d @</PATH/FileName.json>] -H 'Content-Type:application/json' -u
<userName>:<password> [-w "%{http_code}\n"] -X POST
http://<host>:<port>/GC3/api/sdo/Location?id=<LOCATION_GID>
```

Explanation

Table 8-1

String	Definition
<>	Required field sign. Replace its content with real data, and remove the sign when you execute the command.
[]	Optional field sign. Remove the sign when you execute the command.
[-d @</PATH/FileName.json>]	Where to get the JSON file
-u <userName>:<password>	User credential
[-w "%{http_code}\n"]	Print status code
http://<host>:<port>/GC3/api/sdo/Location?id=<LOCATION_GID>	URI for working with Location entity

Request

Here, we have a JSON file that contains the updated Location data. And we want to send this json file to server with CURL command to update a record.

The following is an example of the JSON file. This file is a JSON representation of the Location entity. It includes all of the attributes related to Location as well as the child relationships. These child relations include Location Role Profile, Location Reference Number, and Location Service Provider Preference. Since Location has a 1-to-many relationship with the child relations, the children are represented by arrays in the resulting JSON.

Before executing a POST to update the record it is best to execute a GET in order to obtain the latest copy of the record from the server. Once the latest JSON is obtained for the record it can be modified as required. While modifying the JSON it is possible to change values, add new child records, and delete child records, etc. When working with child records it is important to understand the transactionCode attribute in the JSON. The transactionCode defines how we want to operate on the child record. "NP" means no-change, "I" means insert, "CD" means delete, "IU" means update.

Note: When sending the JSON to the server it is required to send back all of the top level attributes for the object and only the child elements that are being operated upon.


```
{
    "Location":{
           "locationGid":"GUEST.LOCATION.AIR1",
           "locationXid":"AIR1",
           "locationName":"CRIS TEST",
           "city":"Philadelphia",
           "provinceCode":"PA",
           "postalCode":"19046",
           "countryCode3Gid":"USA",
           "timeZoneGid":"EST",
           "lat":40.0,
           "lon":80.0,
           "isTemporary":false,
           "isMakeApptBeforePlan":false,
           "isShipperKnown":true,
           "isAddressValid":"N",
           "slotTimeInterval":null,
           "isLtlSplitable":true,
           "useAppointmentPriority":false,
           "scheduleLowPriorityAppoint":false,
           "enforceTimeWindowAppoint":true,
           "scheduleInfeasibleAppoint":true,
           "bbIsNewStore":false,
           "excludeFromRouteExecution":false,
           "appointDisplay":null,
           "appointDisplayStartTime":0,
           "isTemplate":false,
           "allowDriverRest":false,
           "apptObjectType":"S",
           "isFixedAddress":false,
           "primaryAddressLineSeq":1,
           "domainName":"GUEST",
           "attributeDate1":null,
           "attributeDate2":null,
           "attributeDate3":null,
           "attributeDate4":null,
           "attributeDate5":null,
           "attributeDate6":null,
           "attributeDate7":null,
           "attributeDate8":null,
           "attributeDate9":null,
           "attributeDate10":null,
           "isActive":true,
           "locationRoleProfile":[
                  {
                         "transactionCode":"NP",
                         "locationRoleProfileBeanData":{
                                "locationGid":"GUEST.LOCATION.AIR1",
                                "locationRoleGid":"XDOCK",
                                "xDockIsInboundBias":false,
                                "fixedHandlingTime":null,
                                "varHandlingTime":null,
                                "createXdockHandlingShipment":true,
                                "createPoolHandlingShipment":false,
                                "varHandlingWeight":null,
```



```
                              "maxFreightIdleTime":null,
                              "domainName":"GUEST",
                              "isAllowMixedFreight":false,
                              "varHndcstWgt":null,
                              "varHndcstWgtCost":null,
                              "varHndcstVol":null,
                              "varHndcstVolCost":null
                        }
                  }
                  ],
                  "locationRefnum":[
                        {
                              "transactionCode":"IU",
                              "locationRefnumBeanData":{
                                    "locationGid":"GUEST.LOCATION.AIR1",

   "locationRefnumQualGid":"GUEST.ACCOUNT_NUMBER",
                                    "locationRefnumValue":"123456",
                                    "domainName":"GUEST"
                              }
                        }
                        ],
                        "locationServprovPref":[
                              {
                                    "transactionCode":"NP",
                                    "locationServprovPrefBeanData":{
                                          "locationServprovPrefGid":"GUEST.PRE
ME TOO",
                                          "locationServprovPrefXid":"PRE ME
TOO",
                                          "locationGid":"GUEST.LOCATION.AIR1",
                                          "servprovGid":"ATLC",
                                          "domainName":"GUEST",
                                          "locationServprovPrefD":[
                                                {
                                                      "transactionCode":"NP",

   "locationServprovPrefDBeanData":{

   "locationServprovPrefGid":"GUEST.PRE ME TOO",

   "locationServprovPrefDSeq":1486,
                                                                  "dayOfWeek":1,
                                                                  "begin":0,

   "preferenceLevel":2,

   "duration":86340,

   "isStanding":false,

   "domainName":"GUEST"
                                                      }
                                                }
                                                ]
                                    }
                              }
                              ]
```



```
        }
    }
```

Response

The process responds with various status codes, such as 200 OK and 400 Bad Request.

9. Example - Delete a Location

The following example shows how to use a REST API to delete a location.

> **Note**: You must be granted the "REST - Location - Update" Access Control List (ACL) to use this API.

If you want to delete a Location entity, modify the following example URL, which shows the method call to delete a location:

```
curl -u <userName>:<password>[-w "%{http_code}\n"] -X DELETE
http://host:port/GC3/api/sdo/Location?id=<LOCATION_GID>
```

Explanation

Table 9-1

String	Definition
<>	Required field sign. Replace its content with real data, and remove the sign when you execute the command.
[]	Optional field sign. Remove the sign when you execute the command.
-u <userName>:<password>	User Credential
[-w "%{http_code}\n"]	Write out HTTP Response code when done
http://host:port/GC3/api/sdo/Location?id=<LOCATION_GID>	URI for working with Location entity

Response

The process responds with various status codes, such as 200 OK and 500 Internal Server Error.