P20 POLIZEI 20/20

Infrastructure Administration

Identity Governance Provisioning

Release 1.0.0

P20 GEMEINSAM. DIGITAL. VERNETZT.

Infrastructure Administration

Publication date 022-06-08

by Sophie Strecke, Dieter Steding, Sylvert Bernet, and Adrien Farkas

Program	Polizei 20/20
Program Director	Holger Gadorosi
Project Manager	Norbert Linde
Document Titel	Identity Governance Provisioning
Version	1.2
Creation Date **Created By**	2022-06-08 Dieter Steding
Latest Update **Latest Update By**	2022-06-08 Dieter Steding

Revision History

Revision	Date	Author	Reference
1.0	022-06-08	Dieter Steding	No previous document

Table of Contents

Preface

Audience

This guide is intended for resource administrators and target system integration teams.

Reference Documents

For information about installing and using Oracle Identity and Access Management, visit the following Oracle Help Center page:

- *https://docs.oracle.com/en/middleware/idm/suite/12.2.1.3/index.html*

Confidentiality

The material contained in this documentation represents proprietary, confidential information pertaining to Oracle products and methods.

The audience agrees that the information in this documentation shall not be disclosed outside of Oracle, and shall not be duplicated, used, or disclosed for any purpose other than to evaluate this procedure.

Notational Conventions

The key words **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL** in this document are to be interpreted as described in *[RFC2119]* These key words are capitalized when used to unambiguously specify requirements of the protocol or application features and behavior that affect the interoperability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

Typographical Conventions

The following table describes the typographic changes that are used in this document.

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

Symbol Conventions

The following table explains symbols that might be used in this document.

Convention	Meaning
[ ]	Contains optional arguments and command options.
{ | }	Contains a set of choices for a required command option.
${ }	Indicates a variable reference.
-	Joins simultaneous multiple keystrokes.
+	Joins consecutive multiple keystrokes.
>	Indicates menu item selection in a graphical user interface.

About the Service

The Identity Governanace Provisioning implementation is based on System for Cross-domain Identity Management (SCIM).

> **Note**
>
> According to the *specification* SCIM means
>
> *The System for Cross-domain Identity Management (SCIM) specification is designed to **manage user identity** in cloud-based applications and services in a standardized way to enable interoperability, security, and scalability.*

Even though SCIM means about a standard way of representing user data and managing them, it is not meant to replace the existing system for user management. SCIM acts as a standard interface on top of the existing systems. The underlying userstore can be either LDAP, SQL Database, NoSQL database, or whatever.

SCIM 2.0 comprises of the following three main specifications and some extensions:

- *RFC 7642 - Definitions, Overview, Concepts, and Requirements*
- *RFC 7643 - Core Schema*
- *RFC 7643 - SCIM Protocol*

Any differences between these documents and the current implementation in Identity Governanace Provisioning are described in the Supported API Operations section of this guide.

The following sections contain examples of API requests and responses currently supported in the Identity Governanace Provisioning implementation, along with important notes and constraints to consider in your design.

Before you begin, we recommend that you first review https://docs.aws.amazon.com/singlesignon/latest/userguide/provision-automatically.html#auto-provisioning-considerations

Troubleshooting

These features are covered in the following sub-topics:

- HTTP Codes
- Error Format
- Data Format
- Sorting
- Pagination
- Filtering
- Path Parameter

HTTP Codes

The REST standards use HTTP response codes to indicate success or failure of the operations. Appropriate HTTP status codes describing the error and body message are returned for all requests in a JSON format.

The table below contains possible common HTTP status codes.

HTTP Status Code	Description
200 Ok	The request was successful. Depending on the request method, the response contains the requested data.
201 Created	The request resulted in a new resource being successfully created. Their URI can be communicated in the Location response header.
204 No Content	The request has been accepted and processed, but there is no content to send back to send back to the client. In this case, clients should continue to display the old content. For example, a DELETE request.
307 Temporary Redirect	When the client is requested to repeat the same http request at the `location` identified. The client should not use the `location` provided in the response as the permanent reference to the resource and should continue to use the original request URI.
308 Permanent Redirect	The requested resource is now available at the address specified in the `location` header field, the old address is no longer valid. The client should follow using the same method as the original request (i.e. a POST is followed by a POST). The client should use the `location` provided in the response as the permanent reference to the resource.
400 Bad Request	The client sent an incorrect request that the server cannot understand due to incorrect syntax. For example, validation error,

HTTP Status Code	Description
	missing data or wrong filter syntax. The client should not retry the request without correcting the erroneous data.
401 Unauthorized	The request cannot be processed without authorization.
403 Forbidden	Access is permanently prohibited. Authorization is not recognized and the client should not make the request again. For example, Delete access without appropriate permissions.
404 Not Found	The server could not find the resource requested by the client. The status code 404 can also be returned if no other status code is applicable or if the client is deliberately to be rejected without giving a more detailed reason.
405 Method Not Allowed	The request method (such as GET or POST) is not allowed. This response can returned, for example, if a method is used without the server being able to use them. The server lists all allowed methods in an `Allow` header.
406 Not Acceptable	The resource cannot be supplied in a form requested by the client. For example, data with the media type `application/xml` can be requested, but the server only finds one hit for `application/json`.
408 Request Time-out	The request was not completed by the client in the time specified by the server. The error can occur, for example, when processing large amounts of data.
409 Conflict	The request cannot be processed because this would result in a conflict with the requested resource. If, for example, a resource is to be changed using the PUT method, but which already has a newer version on the server, the editing would create a conflict.
411 Length Required	Processing the request without a `Content-Length` header will be rejected by the server.
500 Internal Server Error	The server detects an internal error and therefore cannot process the request. Errors in programs on the server, e.g. in scripts, can provoke this response.
501 Not Implemented	The server does not have the necessary functionality to process the request. This response is appropriate when the HTTP method used in the request is not recognized or supported.
503 Service Unavailable	The service is temporarily unable to service a request. This can happen due to capacity problems or maintenance downtime.

Date Format

Times and dates are expected and returned in *[ISO 8601]* date format:

```
YYYY-MM-DDTHH:MM:SSZ
```

Instead of 'Z' for UTC time zone you can specify your time zone's locale offset using the following notation:

```
YYYY-MM-DDTHH:MM:SS+hh:mm
```

Example for CET (1 hour behind UTC):

```
2020-03-31T13:00+01:00
```

> **Note**
>
> In HTTP headers, the appropriate recommended date formats are used instead of ISO 8601.

Error Format

The error response data for any API call can be viewed in JSON format. JSON (JavaScript Object Notation) is a lightweight data-interchange format.

Example of the error response to a GET request for a resource that does not exist:

```
{ "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:Error"
  ]
, "status": "404"
, "scimType": "notFound"
, "detail": "Resource of type [Group] with id [uid.generated] does not exists."
}
```

Segment	Description
schemas	
status	The HTTP status code expressed as a JSON string.
scimType	A detail error keyword.
detail	A detailed human-readable message.

Authentication

Clients authenticate to the SCIM APIs by using bearer token authentication, as defined by *RFC 6750*. Every request must include an Authorization request header, where the header value uses the form `Bearer {access token}`

A bearer token must be obtained from an external authorization server, such as Oracle Access Manager.

Authorization

By default, the access token provided by the client in the request (see Authentication) is used to control access to requested resources. The Authorization Server?s access control policies are customizable, but in general, the scopes granted by the access token determine which resources and attributes are returned.

If access controls determine that the client cannot access the requested resource, then a response with a `403` status code is returned.

```
{ "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:Error"
  ]
, "scimType" : "insufficient_scope"
, "status"   : 403
, "detail"   : "Requested operation not allowed by the granted OAuth2 scopes."
}
```

A client may be allowed to access a resource but not all of its attributes. Clients should be prepared to receive incomplete resources, including resources stripped of attributes that are required by the schema.

Sorting

Sorting allows you to specify the order in which the resources are returned, by specifying a combination of `sortBy` and `sortOrder` URL parameters.

Sorting is applied in two conditions:

- When querying a resource using a HTTP GET method.
- When querying a resource using a HTTP POST method to a resource or search.

The value of the sort query parameter is a comma-separated list of sort keys.

Parameter	Description
sortBy	Specifies the attribute whose value is used to order the returned responses. If `sortBy` attribute corresponds to a singular attribute, resources are sorted according to that attributes value. If it is a multi valued attribute, resources are sorted by the value of the primary value, or else, first value in the list. If the attribute is complex, the attribute name **MUST** be a path to a sub attribute in standard attribute notation. For example,; `sortBy=id`
sortOrder	Specifies the order in which the sorting is performed, like in ascending or descending order. For example,; the GET request- *GET /igs/scim/v2/Users?sortOrder=descending*, sorts the list in descending order of participants.

Caution

The `sortOrder` parameter works only when `sortBy` parameter is defined.

Pagination

Pagination parameters can be used together to *"page through"* large numbers of resources so as not to overwhelm the client or service provider.

> **Caution**
> Because pagination is not stateful, clients **MUST** be prepared to handle inconsistent results.

For example, a request for a list of 10 resources beginning with a startIndex of 1 **MAY** return different results when repeated, since resources on the service provider may have changed between requests. Pagination parameters and general behavior are derived from the OpenSearch Protocol.

The following table describes the pagination request parameter, its description and the default value:

Parameter	Default	Description
startIndex	1	Non-negative integer. The 1-based index of the first query result. A value less than 1 is interpreted as 1.
count	None	Non-negative integer. Specifies the desired number of query results per page, example: 10. A value less than 0 is interpreted as 0. A value of 0 indicates that no resource results are to be returned except for total results. When specified, the service provider does not return more results than specified, although it may return fewer results. If not specified, the maximum of results is set by the service provider.

The following table displays the pagination response elements:

Element	Description
totalResults	Non-negative integer. Specifies the total number of results matching the query. For example, 1000.
startIndex	Non-negative integer. The 1-based index of the first result in the current set of query results. For example: 1
itemsPerPage	Non-negative integer. The number of query results that are returned in a query response page. For example, 10.

Example

```
curl --request GET \
     --location 'https://<service-host>:<service-port>/igs/scim/v2/Users?startIndex=1&count=10' \
     --header 'Accept: pplication/scim+json' \
     --header 'Content Type: pplication/scim+json' \
     --header 'Authorization: Bearer eyJraWQiOiJTZWN1cmVEb21haW4yIiwieDV0IjoiSjFs...'
```

The response to the query above returns metadata regarding paging similar to the following example.

```
{ "schemas": [
    "urn:ietf:params:scim:api:messages:2.0:ListResponse"
  ]
, "totalResults": 21
, "startIndex": 1
, "itemsPerPage": 10
  "Resources": [
    ...
  ]
}
```

Note

In the previous example, to continue paging, set the startIndex to 11 and re-fetch, that is:

`/igs/scim/v2/Users?startIndex=11&count=10`

Filtering

Filters provide an additional flexibility to users interacting with large data by narrowing down the result sets.

Attribute names and attribute operators used in filters are case insensitive. For example, the following two expressions will evaluate to the same logical value:

The filter parameter **MUST** contain at least one valid expression. Each expression **MUST** contain an attribute name followed by an attribute operator and optional value. Multiple expressions **MAY** be combined using logical operators. Expressions **MAY** be grouped together using round brackets (and).

The following table shows the filter operators that are supported:

Attribute Operators

Filter	Description
eq	Equal
ne	Not Equal
co	Contains
sw	Starts With
ew	Ends With

Filter	Description
gt	Greater Than
ge	Greater Than or Equal To
lt	Less Than
le	Less Than or Equal To

Logical Operators

Filter	Description
and	And
or	Or
not	Not

Grouping Operators

Filter	Description
()	Precedence Grouping

Filters are evaluated in the following order of operations:

1. Grouping Operators
2. Logical Operators
3. Attribute Operators

Path parameters

If an endpoint has path parameters, the documentation displays them with enclosing curly brackets.

Example

```
DELETE /igs/scim/v2/Users/{id}
```

The {id} path parameter needs to be replaced with the identifier of the account type in this example. The curly bracket must not be included.

Path parameters are required to be URL-encoded. If not, it does not match an API endpoint and responds with a HTTP status code `404`.

Supported API Endpoints

The following API enpoints are supported by the IGS SCIM Provisioning implementation:

- Service Provider Config
- Resource Types
- Schema

Service Provider Config

You can use the `/ServiceProviderConfig` endpoint for `GET` requests to view additional information about the IGS SCIM Provisioning implementation. The `/ServiceProviderConfig` endpoint is read only.

In this endpoint the service provider defines specifications compliance and the types of authentication they support to the API (most commonly used is basic authentication, Bearer token authentication and Oauth 2.0authentication). Authentication for this configuration endpoint is not required, unlike the other endpoints.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- None

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- None

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

Example Response

Resource Types

You can use the `/ResourceTypes` endpoint for `GET` to view the supported resources types, like Users, Groups, and all Schema extensions. The `/ResourceTypes` endpoint is read only.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- None

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- None

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

Example Response

Schema

You can use the `/Schemas` endpoint for GET TBD. The `/Schemas` endpoint is read only.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- None

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- None

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

Example Response

Supported API Operations

The following API operations are supported by the IGS SCIM Provisioning implementation:

User Operations

The following API operations are supported by the IGS SCIM Provisioning `/Users` endpoint implementation:

- Create User
- Lookup User
- Search User
- Delete User
- Modify User

Create User

You can create new users from a POST request using the IGS SCIM Provisioning implementation **/Users** endpoint. See the examples below.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- `ims`, `photos`, `x509Certificates` and `entitlements` are ignored.
- `display` subattribute for `emails` and `phoneNumbers` are ignored.

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- The `userName` field can contain letters, accented characters, symbols, numbers, punctuation.
- The `userName` field is required.
- Multiple values in multi-value attributes (such as `emails`, `addresses` `phoneNumbers`) are not supported. Only single values are permitted.
- The `emails` attribute value must be marked as primary.
- The `phoneNumbers` attribute value must be marked as primary.

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
ResourceConflictException	User already exists.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

```
curl --requesr POST
     --location 'https://<service-host>:<service-port>/igs/scim/v2/Users' \
     --header 'Accept: application/scim+json' \
     --header 'Content-Type: application/scim+json' \
     --header 'Authorization: Bearer <your-token>' \
     --data-raw \
     '{ "schemas"  : [
         "urn:ietf:params:scim:schemas:core:2.0:User"
       ]
     , "userName" : "bk4711127"
     , "password" : "changeit"
     , "name": {
         "formatted"  : "Dieter Steding"
       , "familyName" : "Steding"
       , "givenName"  : "Dieter"
       }
     , "preferredLanguage" : "de"
     , "emails":[
         { "primary" : true
         , "type"    : "work"
         , "value"   : "dieter.steding@oracle.com"
         }
       ]
     , "phoneNumbers":[
         { "primary" : true
         , "type"    : "work"
         , "value"   : "+49 177 5948 437"
         }
       ]
     }'
```

Request Header

Invoking this endpoint requires following request header:

Name	Type	Value
accept	string	application/scim+json
content-type	string	application/scim+json
authorization	string	*Bearer* appended with the token obtained by Obtaining a Token

Query Parameter

The supported query parameter are the following:

Name	Type	Required	Description
startIndex	integer	no	The 1-based index of the first query result to return. A value less than 1 is interpreted as 1.

Name	Type	Required	Description
count	integer	no	Specifies the desired maximum number of query results per batch. A negative value is interpreted as 0. A value of 0 indicates that no resource results are to be returned.
filter	string	no	The filter criteria to apply on the search. See Filter Criteria.
sortBy	string	no	The sort criteria to apply on the search.
sortOrder	string	no	The order in which the sorting is performed, like in ascending or descending order.
attributes	string list	no	The value of this parameter is a comma-separated list of resource attribute names, expressed in standard attribute notation (e.g., id, name).
excludedAttributes	string list	no	The value of this parameter is a comma-separated list of resource attribute names, expressed in standard attribute notation (e.g., id, name).

Example Response

Lookup User

Existing users can be retrieved by making a GET request using the IGS SCIM Provisioning implementation **/Users** endpoint with an existing user ID.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- None

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- None

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
NotFoundException	Specified user does not exist.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

Example Response

Search Role

This endpoint provides the ability to perform filter queries on an existing list of users through aGET request to **/Roles** by inserting additional filters. Only a maximum of 1000 results can be returned at a time. See the Constraints section for more information.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- None

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- At this time, the SearchRoles API is only capable of returning up to 1000 results.
- Filter must be specified as follows: <filterAttribute> eq "<filterValue>"

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
ResourceConflictException	Role already exists.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

Example Response

Filter Examples

The following different filter combinations are supported.

- id
- userName
- firstName
- lastName

The filters can be applied in the formats as shown.

Single filter

Multiple filter

Delete User

A user can be deleted by making a DELETE request using the IGS SCIM Provisioning implementation **/Users** endpoint with an existing user ID.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- None

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- None

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
NotFoundException	Specified user does not exist.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

Example Response

Modify User

Role Operations

The following API operations are supported by the IGS SCIM Provisioning `/Roles` endpoint implementation:

- Create Role
- Lookup Role
- Search Role
- Delete Role
- Modify Role

Create Role

You can create new roles from a POST request using the IGS SCIM Provisioning implementation **/Roles** endpoint. See the examples below.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- `ims`, `photos`, `x509Certificates` and `entitlements` are ignored.
- `display` subattribute for `emails` and `phoneNumbers` are ignored.

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- The `roleName` field can contain letters, accented characters, symbols, numbers, punctuation.
- The `roleName` field is required.
- Multiple values in multi-value attributes (such as `emails`, `addresses` `phoneNumbers`) are not supported. Only single values are permitted.
- The `emails` attribute value must be marked as primary.
- The `phoneNumbers` attribute value must be marked as primary.

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
ResourceConflictException	Role already exists.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

Example Response

Lookup Role

Existing roles can be retrieved by making a GET request using the IGS SCIM Provisioning implementation **/Roles** endpoint with an existing role ID.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- None

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- None

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
NotFoundException	Specified role does not exist.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

Example Response

Search Role

This endpoint provides the ability to perform filter queries on an existing list of roles through a GET request to **/Roles** by inserting additional filters. Only a maximum of 1000 results can be returned at a time. See the Constraints section for more information.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- None

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- At this time, the SearchRoles API is only capable of returning up to 1000 results.
- Filter must be specified as follows: <filterAttribute> eq "<filterValue>"

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
ResourceConflictException	Role already exists.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

Example Response

Filter Examples

The following different filter combinations are supported.

- id
- roleName
- firstName
- lastName

The filters can be applied in the formats as shown.

Single filter

Multiple filter

Delete Role

A role can be deleted by making a DELETE request using the IGS SCIM Provisioning implementation **/Roles** endpoint with an existing role ID.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- None

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- None

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
NotFoundException	Specified role does not exist.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

Example Response

Modify Role

Tenant Operations

The following API operations are supported by the IGS SCIM Provisioning `/Tenants` endpoint implementation:

- Create Tenant
- Lookup Tenant
- Search Tenant
- Delete Tenant
- Modify Tenant

Create Tenant

You can create new tenants from a POST request using the IGS SCIM Provisioning implementation **/Tenants** endpoint. See the examples below.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- None

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- None

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
ResourceConflictException	Role already exists.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

Example Response

Lookup Tenant

Existing tenants can be retrieved by making a GET request using the IGS SCIM Provisioning implementation **/Tenants** endpoint with an existing tenant ID.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- None

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- None

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
NotFoundException	Specified role does not exist.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

Example Response

Search Tenant

This endpoint provides the ability to perform filter queries on an existing list of tenants through a GET request to **/Tenants** by inserting additional filters. Only a maximum of 1000 results can be returned at a time. See the Constraints section for more information.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- None

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- At this time, the SearchTenants API is only capable of returning up to 1000 results.
- Filter must be specified as follows: <filterAttribute> eq "<filterValue>"

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
ResourceConflictException	Tenant already exists.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

Example Response

Filter Examples

The following different filter combinations are supported.

- id

The filters can be applied in the formats as shown.

Single filter

Multiple filter

Delete Tenant

A tenant can be deleted by making a DELETE request using the IGS SCIM Provisioning implementation **/Tenants** endpoint with an existing tenant ID.

Not supported

The IGS SCIM Provisioning implementation does not support the following aspects of this API operation.

- None

Constraints

The IGS SCIM Provisioning implementation has the following constraints for this API operation.

- None

Errors

The following IGS SCIM Provisioning implementation errors are common for this API operation.

Error	Condition
NotAllowedException	Operation is not permitted based on the supplied authorization.
NotFoundException	Specified role does not exist.
ServerErrorException	Service failed to process the request.

Examples

Following are example requests and responses for this API operation.

Example Request

Example Response

Modify Tenant

Troubleshooting

403 Forbidden

The server is responding with HTTP 403 Forbidden

The request was valid, but the server is refusing action. The user might not have the necessary permissions for a resource, or may need an account of some sort.

A potential issue can be a wrong *Accept* and/or *Content-Type*

> **Important**
>
> Both request haeder have to be *application/scim+json*.
>
> Neither *application/json* or *application/json+scim* are valid values for the request headers.