opentext™

Directory and Resource Administrator

REST Services Technical Reference

July 2023

Legal Notice

Contents

About this Reference

The Net IQ fDRA REST Services Technical Reference provides information about the REST endpoint URIs, the payload contents for each URI, and the response returned to the client.

Intended Audience

This guide provides information for application developers to integrate their applications with DRA.

Additional Documentation

This guide is part of the Directory and Resource Administrator documentation set. For the most recent version of this guide and other DRA documentation resources, visit the DRA Documentation website (https://www.netiq.com/documentation/directory-and-resource-administrator/).

Contact Information

We want to hear your comments and suggestions about this book and the other documentation included with this product. You can use the **comment on this topic** link at the bottom of each page of the online documentation, or send an email to Documentation-Feedback@microfocus.com.

For specific product issues, contact Micro Focus Customer Care at https://www.microfocus.com/ support-and-services/.

Understanding the DRA REST Endpoints and APIs

You can develop a client for making DRA requests using .NET managed code or using PowerShell. Either approach provides an easy way to integrate your existing application with DRA.

The DRA REST endpoints allow any REST-enabled client to automate Active Directory administration tasks using DRA. Using the credentials of a DRA administrator, the REST client can perform administration tasks across multiple domains and forests from a single client. The DRA server enforces the DRA delegation of powers to ensure that the client can only perform tasks for which the user has privileges defined.

All requests and responses are formatted as JSON documents and sent to the DRA REST Service via HTTP over SSL. Any client that can handle JSON-formatted data and make HTTP requests can use the DRA REST endpoints.

Authentication

The following users can be used for any REST request:

- **Client user**: The user making the request. The user can authenticate to IIS through basic authentication, Windows integration, or by providing a signed certificate.
    - Basic Authentication: The request header must contain the client user's credentials. The base endpoint address must be `https://localhost:8755/`.
    - Windows Integration: The base endpoint address must be `https://localhost:8755/console`.
    - Certificate-based Authentication: The request must include a client certificate that is installed in the local store on the client. The client certificate must be signed by an authority that is trusted by the DRA REST Server. The base endpoint address must be `https://localhost:8755/certificate`.
- **DRA Administrator**: The DRA Administrator is delegated DRA powers to perform the requested operation. If the DRA Administrator is not the same as the client user, the REST request must include the DRA Administrator's credentials.

Working with the .NET Interfaces

You can connect to the DRA REST Service by sending an HTTPS request to the server where the service is running. The endpoint is selected based on the URI. The JSON-formatted payload provides additional information for the request such as the DRA administrator credentials, the DRA server to use (default is the local server), and additional details about the request.

Most requests will require a payload that describes the object you are working with. See Appendix A for some sample payloads. You can get a list of all implemented payloads in your browser by using the following URL on a server with DRA REST Extensions installed: `https://localhost:8755/Console/help`.

There are several tools you can use for exploring REST interfaces. Any of the tools will work for connecting with the DRA REST endpoints, provided you configure the request with the correct HTTP verb, URI, headers, and payload.

Configuring the REST Client

Use the information in this section to configure the REST Client.

HTTP Verb

All requests to perform DRA operations will be sent using **HTTP POST**. The request will usually include a payload that specifies additional information such as the DRA Admin credentials, a specific DRA server to where the operation should be performed, and properties of the target of the request.

There is one request that uses HTTP GET. This request only verifies the state of the REST Service, it does not perform a DRA operation.

Headers

Each request requires the following two headers:

- Authorization: Header containing the credentials of the user making the request. (Note: The DRA assistant administration user can be a different user. See Connection Parameters.)
- Content-Type: application/json

URI

The client connects to the DRA REST Service by naming the computer and port in the URI. For example:

https://MyServer:port

Where	Specifies
MyServer	The name of the computer running the DRA REST Service.
port	The REST Service Port entered during installation. The default is 8755. To verify the port number, open the file NetIQ.DRA.RestService.exe.config in the DRA installation folder. In <system.serviceModel><services><service><host><baseAddresses> you can see the port number in the <add baseAddresses> element for *localhost*.

After the port, the URI names “dra” as the target. There are different types of requests:

Enumeration	dra/domains/*{domain.name}/{objectType}s/get* to list objects in a domain dra/*domains/get* to list domains
Create	dra/domains/*{domain.name}/{objectType}s/*post
Other Actions	dra/domains/*{domain.name}/{objectType}s/{verb}*

Where	**Specifies**
domain.name	The full domain name using dot notation. For example, mydomain.corp
objectType	The class of the Active Directory object, such as, “computer”. The object type is always plural. For example, *computers*.
verb	The HTTP verb. Options are: **Get**: retrieves information about existing objects **Post**: creates a new object **Put**: updates an existing object **Delete**: removes an existing object

Payload

For most requests, you will include a payload in the form of a JSON document with additional information to complete the request. Payload contents may contain one or more of the following:

- Class objects that supply properties related to a create or update operation.
- Object identifier strings that name an existing object for get, update or delete operations.
- Connection parameters that specify a specific DRA server and/or the credentials for a DRA admin who should perform the request.
- Attributes object that lists the desired properties to return for an enumeration operation.
- EnumerationOptions that control the amount of data returned to the client.

The text in the JSON document is parsed by the REST endpoint according to known names for objects and properties. When writing the JSON document, take care to match the object and property names found in the `DRARestConfiguration.json` file. If the name is misspelt or the case is incorrect, the item may be ignored by the endpoint or it may cause the DRA Server to return an error.

Class Objects

The name of the class object identifies the type of object for the request such as *computer*. ***Object names are case sensitive.*** The object members describe the attributes you want the DRA server to associate with the object you are creating or updating.

When creating new objects, the object class must at least contain a name and the full path to the object. You can provide this information using the following attributes:

Attributes	Sample Payload

name and friendlyParentPath	"user": {
	"description": my user description",
	"name": "user123",
	"friendlyParentPath": "mydomain.corp/ouRoot/oux/ouy"
	}

distinguishedName	"user": {
	"description": my user description",
	"distinguishedName": "cn=user123,ou=ouRoot,ou=oux, ou=ouy,dc=mydomain,dc=corp "
	}

friendlyName or canonicalName	"user": {
	"description": my user description",
	"friendlyName": "mydomain.corp/ouRoot/oux/ouy/user123"
	}

Where possible, the endpoint will supply a default value for an attribute, if DRA requires it. For example, the samAccountName attribute is not required by REST, but it is required in DRA. If the samAccountName attribute is not supplied, the REST endpoint will set the attribute to match the object name (with a trailing $ for computers).

The list of attributes included in the class object will depend on the action and the object type. Typically, the properties have the same name as the AD attribute. **Property names are case-sensitive.** Examples of properties that the client might specify:

- **name**: the internal name of the object
- **samAccountName**: the AD sAMAccountName
- **description**: object description
- **displayName**: the user-friendly name in AD
- **distinguishedName**: the full DN in AD
- **isDisabled**: to set the enabled state of the item

See Understanding The JSON Configuration for the list of properties defined for each object. See Appendix A for object class examples for each operation.

Object Identifier

When performing an operation on an existing object, the payload must contain an object identifier that names the existing object. The object identifier name is always the object class plus “Identifier”. For example, *computerIdentifier*, *userIdentifier*, or *azureGroupIdentifier* (Azure object).

The REST endpoint supports identifying an object using these formats:

Format	Example
Name	"computerIdentifier":"object05"
Distinguished Name	"userIdentifier":"CN=User22,OU=myOU,DC=MYDOM,DC=corp"
Distinguised Name (Azure object)	“azureUserIdentifier”:”CN=ObjectGUID,AZ=TenentGUID”

When using the simple name, the object name must be unique within the domain. The REST endpoint will attempt to resolve the simple name to the full distinguished name by making a call to the DRA server. If more than one object of that class is found in the domain having the same name, the request will fail.

The examples in Appendix A will generally show only one format. However, any format shown above can be used.

Connection Parameters

If the request should go to a particular DRA server, or if the REST client user does not have the DRA admin privileges to perform the request, you will need to populate the connectionParameters object. You can specify either a specific DRA server or credentials for the DRA admin user or both.

The connectionParameters object has the following attributes:

Attribute	Description
username	The full username of the DRA admin.
password	The password for the DRA admin.

Example:

```
{
  "connectionParameters": {
    "draServerNameAndPort": "DRASERVER27:11192",
    "userName": "mydomain\\someUser",
    "password": "$secure1 $"
  }
}
```

Attributes List

Each object type has a default list of properties to return for a Get request. The same default list is applied whether you Get information about a particular object or you request a list of objects. For any Get request, you can override the default list by adding the attributes object to the payload.

The format of the attributes object is:

```
{"attributes":["property1", "property2","property3"]}
```

You can see the default list of properties returned for each object type in the file `DRARestConfiguration.json` in the installation folder. Look for the JSON object having a name like [objectType]GetInfoDefaultProperties.

Enumeration Options

When you search DRA for a list of objects, you can provide information such as where to search and how many objects to return at one time. When the returned list is very large, you can specify options to process the results in sections.

The enumerationOptions object has the following attributes. All attributes are optional.

Attribute	Description
containerDistinguishedName	The distinguished name of the starting container for enumeration. If a container is not specified then the domain in the URI is used as the container and if there is no domain in the URI then the module=accounts special container is searched (basically the container that corresponds to the 'All My Managed Objects' node in the win32 console).
multiMatch	This attribute is similar to “Ambigous Name Resolution” for Active Directory. It is used with a filtered search based on multiple attributes and returns results that match any of those attributes using “begins with” criteria, such as name, display name, first name, and last name attributes.
includeChildContainers	true or false. Default is false.
enforceServerLimit	true or false. Default is true.
objectsPerResponse	The number of objects to return at a time. Default is 250.
resumeString	When the number of objects matching the enumeration filter exceeds the *objectsPerResponse* value, the server returns the *resumeString* attribute in the response. To get the next “page” of objects, specify the *resumeString* provided in the previous response.
isManagedContainerEnum	true or false. Default is false. When true, causes the request to run a Managed Container enumeration. For more information, see Container Enumeration.
ldapEnumerationOptions -- or -- pagedEnumerationOptions	Sets parameters to control the behaviour of an LDAP or paged search request. The options are listed below. Only one of these options should be used in a request. If both options are present, the server will use the containerEnumerationOptions parameter.

The pagedEnumerationsOptions has the following attributes. All attributes are optional. To determine if you want to use a paged or container enumeration request, see Search Operation Types.

Attribute	Description
sortHint	Tells the server which property to use for ordering the results. If this is not specified, the server will choose the order.
startRow	Specify this parameter when the number of objects matching the enumeration filter exceeds the *objectsPerResponse* value and you want the results to contain data from somewhere other than the beginning. Use this parameter with resumeString to request more results from a search. See Paged Enumeration for more information.

For example:

```
{
    "enumerationOptions": {
        "objectsPerResponse": 5,
        "includeChildContainers": true,
        "resumeString": "CN=Accounting-DG, OU=Accounting,OU=My Product
Preview,DC=MYDOMAIN,DC=CORP"
    }
}
```

The ldapEnumerationOptions has the following attributes.

Attribute	Description
vaQueryString	Optional. Specifies search criteria for virtual attributes. The format of this string is the format sent by the console when making a request. It is not LADP-format. For example, this is an OR query that checks the value of VA01 and VA02: <VAQUERY> <OR> <USER> <ATTRIBUTE>VA01</ATTRIBUTE> <CONDITION>STARTS-WITH</CONDITION> <VALUE>Z</VALUE> </USER> <USER> <ATTRIBUTE>VA02</ATTRIBUTE> <CONDITION>STARTS-WITH</CONDITION> <VALUE>Q</VALUE> </USER> </OR> </VAQUERY> Valid conditions are: ENDS-WITH IS

Attribute	Description
	IS-NOT STARTS-WITH PRESENT The virtual attribute query string is a continuation of the LDAP query string. The OR AND statements are added to the LDAP query filter.
startRow	Optional. Default is 0. Specify this parameter when the number of objects matching the enumeration filter exceeds the *objectsPerResponse* value and you want the results to contain data from somewhere other than the beginning. Use this parameter with resumeString to request more results from a search.

See the Enumeration section of Appendix A for more enumeration examples.

List All DRA Servers:

This request will return all available DRA servers for the domain.

URI	`https: //myDRAserver:8755/dra/domains/MyDomain.corp/draservers/get`
Payload	None

Response	{
	"draServers": [{
	"name": "HOUDVDR202.MYDOMAIN.CORP",
	"machine": "HOUDVDR200.MYDOMAIN.CORP",
	"mmsId": "{87023C1A-3066-4954-B104-6F2617AC5E22}",
	"version": "8.70.0",
	"type": "Primary",
	"forest": "MYDOMAIN.CORP",
	"site": "Default-First-Site-Name",
	"domain": "MYDOMAIN.CORP",
	"restServicePort": 8755,
	}]
	}

Other Examples

See Appendix A for a detailed list of examples.

Searching for DRA Objects (Enumeration)

The DRA REST extensions have several endpoints that allow you to query the DRA server for a list of objects matching the filter criteria. You can search for specific object types within a domain, or you can search across all object types in all the domains managed by DRA. In either case, the endpoint request will return the results in an array of objects. If the results include multiple object types, each object type will be in a separate array.

When searching, the DRA server will always honor the delegation rules. The results will only contain the objects over which the requesting user has been granted at least view property powers. For more information about setting the credentials of the requesting user see Authentication.

There are two types of enumeration: container and LDAP. The container enumeration allows you to filter using DRA object properties. The LDAP enumeration allows you to filter by specifying an LDAP query string.

Container Enumeration

The container enumeration endpoints are:

- dra/domains/{domain fqdn}/{object type}/get

  This endpoint allows you to search for a single type of object within a single domain. When using this endpoint, you can filter on the object type named in the URI. The results will only contain objects of the type named in the URI.

- dra/managedObjects/get

  This endpoint allows you to search across all managed domains for any of the supported object types. This endpoint accepts filters for multiple object types. The results will contain several object arrays: one for each type of object managed by DRA.

Container Enumeration Payload

The payload for an enumeration request defines the filters for the search as well as the contents and format of the search results. When results exceed the maximum number of objects you want to process at one time, there are options to request the next set of data. In this way, you can process all the objects found by your search, even if the results contain thousands of objects.

The payload can include any of the following objects:

Enumeration Options	Optional. Specifies parameters such as the path to search. When not specified, the search will be done on the root container. Generally, you will want to specify at least the *containerDistinguishedName* parameter. See Enumeration Options for a list of options you can specify.
Object Filters	Optional. When none specified, all objects are returned. Filters are specified by using the same class structure and properties as you use for creating and updating objects managed by DRA. For example, when you want to search for users, you would specify the filter using the properties of the User object. The JSON object names in the payload follow the pattern of object type plus AndFilter or OrFilter. For example: userAndFilter, computerOrFilter.

	Objects supporting And Or filters are computer, contact, domain, group, and user. Or filters are available for performing a search of EquipmentResourceMailbox and RoomResourceMailbox objects. Technical Note: When the search includes any filter for a Resource Mailbox object, then the results will be limited to Resource Mailboxes. Other filters are ignored by the server. You specify whether the search should return objects that match any of the criteria using an OR filter. You specify that the search should only return objects matching all the criteria using an AndFilter.
Attributes	Optional. Specifies the list of fields you want to be returned in the results. See Attributes List for more information.
Connection Parameters	Optional. Specifies the name of the DRA server that should perform the search and the DRA Admin whose credentials should be used for checking powers over the objects returned. If not specified, the search will be performed on the local machine using the credentials of the user making the request. See Connection Parameters for more information.

Container Enumeration Filters

DRA supports the following filtering values:

* (asterisk)	Matches any number of characters. Can be used anywhere in the property. XXX* equates to 'begins with' *XXX equates to 'ends with' XX*YY equates to 'contains'
# (number sign)	Matches a single numeric digit. Can be used anywhere in the property. You can use multiple number sign characters to mask multiple numeric digits. For example: ### matches any 3-digit number (001, 987, etc.)
? (question mark)	Matches a single non-numeric character, including special characters. You can use multiple question marks to mask multiple characters. For example: ??? matches any 3 non-numeric characters (abc, ZZZ, etc.)

You can use the filter values together in a single filter. Matches are not case-sensitive. Here are some examples showing how you would specify some filters in the payload. The examples show the JSON-formatted filter for one field.

"location":"HOU-7##*"	Searches for objects whose Location field begin with the literal value HOU-7, followed by two numeric digits, followed by any characters. The character searches are not case-sensitive. So, the above filters would return locations that began with HOU, Hou, hou, and so on. Matches: `HOU-722 xx yy`

	Does not match: `HOU-7A22`
"managedBy":"*george*"	Searches for objects whose managedBy field contain 'george'. Matches: `CN=Hernandez, George,DC=MyDomain,DC=corp`
"employeeId":"??9*"	Searches for objects whose employeeID field start with two characters followed by the number 9 and ending with any number of characters. Matches: `ZZ987345` or `AB9-ZZZ` Does not match: `00987345`

For each object type, you can supply a filter that tells the search to match all the specified filters (AndFilter) or any of the specified filters (OrFilter). When you use the domain/object specific endpoint, the payload can contain filters for the object type named in the URI. If you use the managedObjects endpoint, you can supply filters for each type of object you want returned in the results.

Any object returned from the search will match all the filter criteria. For example:

userAndFilter	"location":"HOU-7##*"
userOrFilter	"managedBy":"*George*","department":"*sales*"
userOrFilter	"multiMatch":"adm*"

Returns all user objects in the specified path whose location matches the AndFilter AND whose managedBy field contains George OR whose department contains 'sales' OR who has one or more multi-match attributes that begin with the first three letters a d m.

NOTE: Each filter includes the class of the object in the filter name. (E.g., userAndFilter will only match objects of type User.) When you supply multiple filters, all filters will be applied to the entire result set. This means that if you are searching across multiple object types in one request, you should use OrFilters. If you specify two AndFilters for different object types, the result set will be empty: any object can only match one class.

Container Enumeration Search Types

The type of search operation you request depends on what the application you are writing will do with the results. The managedObjects/get endpoint supports three types of searches:

- **Container Enumeration**: Searches for all objects in the current container and child containers, if specified. The results are returned in object arrays; there is one array for each object type. You would iterate over the results by using the ResumeString parameter. The ResumeString contains the search path to the last item in the current result set. This search would be used for automation tasks where you would perform further processing on the objects returned from the search.
- **Managed Container Enumeration**: An EnumerationOption flag that returns objects matching the filters and all child containers. *Does not search child containers*, even if includeChildContainers is true. This search could be used to populate a tree structure. This type of search is requested by setting the isManagedContainerEnum flag. Otherwise, it is the same as a ContainerEnumeration.
- **Paged Enumeration**: Performs a Container Enumeration search and returns a list of rows. Using the startRow and cachId parameters, you can request data from a specific place in the results. This type of search would be used in a user interface where you want to present the search results as pages and allow the user to scroll up and down.

Container Enumeration

A Container Enumeration search is the default type of search. You can explicitly request a Container Enumeration search by specifying the containerEnumerationOptions object in the payload.

The search results will contain two attributes that you will need to iterate over multiple pages of results: totalNumberOfObjects and resumeString. If resumeString is empty, then there are no more results to process for this search request. If there is a value in resumeString, resend the same request with the resumeString attribute set. Results are only processed once, going forward in the list each time.

The following table shows the sequence of payload and response JSON that demonstrates how to control the result set and iterate over multiple pages of results. Only the JSON that is needed to demonstrate the flow of data is shown.

Request	Server Response

"enumerationOptions": {
"containerDistinguishedName": "OU=Accounting,DC=MYDOMAIN,DC=CORP",
"resumeString": ""
}
	{
	...
	"resumeString": "CN=Accounting-DG,OU=Accounting,DC=MYDOMAIN,DC=CORP",
	"totalNumberOfObjects": 294,
	"numberOfRowsReturned": 250,
	"isSearchFinished": false
	}

Paged Enumeration

A Paged Enumeration request performs the search the same as the Container Enumeration. However, the result set has a different set of attributes for retrieving multi-paged results. The paged enumeration allows you to retrieve the results multiple times, starting at different places in the result set. This is designed for displaying data in a graphical user interface.

Here is how you could use the Paged Enumeration options to present data to the user:

- In each request, set the objectsPerResponse attribute to the number of rows you want to display at one time.
- Use the totalNumberOfObjects attribute in the response to calculate the number of pages.
- When the user chooses a different page for viewing, calculate the row number that is needed and set the startRow attribute to that number.

The following table shows the sequence of payload and response JSON that demonstrates how to control the result set and iterate over multiple pages of results. Only the JSON that is needed to demonstrate the flow of data is shown.

Request	Server Response
(Initial request) "enumerationOptions": { "containerDistinguishedName": "OU=Accounting,DC=MYDOMAIN,DC=CORP", "objectsPerResponse": 15,	{ ... "resumeString": "{214A15E1-44D4-460D-9CC4-D8CA04EFDE6E}", "totalNumberOfObjects": 294,

"resumeString": "",
"pagedEnumerationOptions ": {
"startRow": 1,
"cacheId": ""
}
}
	"numberOfRowsReturned": 15,
	"isSearchFinished": false
	}

(2nd request)
"enumerationOptions": {
"containerDistinguishedName": "OU=Accounting,DC=MYDOMAIN,DC=CORP",
"objectsPerResponse": 15,
"resumeString": "{214A15E1-44D4-460D-9CC4-D8CA04EFDE6E}",
"pagedEnumerationOptions ": {
"startRow": 31
}
}
	Retrieves page 3

Understanding the JSON Configuration

The REST Extensions installation folder contains a file called `DRARestConfiguration.json`. This file contains several JSON objects that are loaded by the REST Service during startup. The JSON describes all the supported object types and their properties.

Object Property Maps

The object property name maps make the association between the property names sent from the DRA REST client and the names expected by the DRA server. In many cases, client property names are more user-friendly. The property name maps cannot be changed or edited.

The baseObjectPropertyNameMap describes properties common to all the supported objects. For each supported object, you will find a corresponding [*objectType*]ObjectPropertyNameMap that describes properties unique to that object. To determine all the properties available for an object, consider both the baseObjectPropertyNameMap and the object-specific Property Name Map. The entries in the object map look like this:

```
"contactPropertyNameMap": [{
        "draPropertyName": "l",
        "clientPropertyName": "city"
    }, {
        "draPropertyName": "company",
        "clientPropertyName": "company"
    }
    ...
```

The REST client should specify the values named on the "clientPropertyName" side of the map.

Default Properties Lists

The DRA REST feature is installed with a default list of attributes to return when performing a query to gather information about one or more objects. You can modify this default list to meet the needs of your environment.

To modify this list, create a backup of the file DRARestConfiguration.json in the installation folder. Then, edit the original file. The default attributes for each object type are listed in the *PropertiesToGet* field. Change the list and save the document. Changes take effect the next time that the REST service is restarted.

If the edits result in invalid JSON, the *DRA REST Service* will stop.

Working with Virtual Attributes

You can set the value for existing virtual attributes from your REST client by specifying the values in the "additionalAttributes" object. Both the PowerShell Extensions and REST Endpoints will pass the value of a supplied virtual attribute to the server.

NOTE: If the virtual attribute is configured to require a value, the create request coming from the REST Extensions will not validate that requirement. If the virtual attribute is not included in a create request, the object will be created with an empty virtual attribute value.

Sending Virtual Attributes in a Payload

After you define a virtual attribute and associate it with a class, you can set the value for the virtual attributes by sending it in a special payload object called "additionalAttributes". This object is a collection of keys and values. The key is the name of the property on the DRA server. For example, if you defined a virtual attribute called "OfficeBuilding" and associated it with computers, you could create the computer and set the value using a payload like this:

```
{
  "computer": {
    "description": "computer for testing",
    "distinguishedName": "CN=TESTCREATE25,OU=someOU,DC=MyDomain,DC=corp",
    "additionalAttributes": {
      "OfficeBuilding": "Downtown",
      "A_Multi_Value": ["Value1", "Value2"],
      "An_Integer_Value": 47,
    }
  }
}
```

Notice that "additionalAttributes" uses curly braces to wrap the values. The example also shows how to format a multi-valued attribute and an integer.

Appendix A – REST Endpoint Examples

This appendix provides just a few examples of how to use DRA REST extensions on endpoints to execute actions in Active Directory, such as creating, updating deleting, and searching for objects. For detailed comprehensive Help of the DRA REST Extensions SDK, see the *DRA REST Services Guide* on the DRA Documentation site.

Create

The examples below demonstrate payload options for creating a computer:

Create Using friendlyName Attribute

URI	/dra/domains/MyDomain.corp/computers/post

Payload	{
	"computer": {
	"description": "my 05 favorite computer",
	"friendlyName": "MyDomain.corp/Sharon/TESTCREATE03"
	}
	}

Response	{"errors":[]}

Create Using Name + friendlyParentPath Attribute

URI	/dra/domains/MyDomain.corp/computers/post

Payload	{
	"computer": {
	"name": "TESTCREATE09",
	"description": "my 09 favorite computer",
	"friendlyParentPath": "MyDomain.corp/Sharon"
	}
	}

Response	{"errors":[]}

Create Using distinguishedName Attribute

URI	dra/domains/MyDomain.corp/computers/post
Payload	{

	"computer": {
	"description": "my distinguished 07 computer",
	"distinguishedName": "CN=TESTCREATE07,OU=Sharon,DC=MyDomain,DC=corp"
	}
	}

Response	{"errors":[]}

Create Using canonicalName Attribute

URI	/dra/domains/MyDomain.corp/computers/post

Payload	{
	"computer": {
	"description": "my 05 favorite computer",
	"canonicalName": "MyDomain.corp/Sharon/TESTCREATE03"
	}
	}

Response	{"errors":[]}

Create Missing Required Input

URI	/dra/domains/MyDomain.corp/computers/post

Payload	{
	"computer": {
	"name": "TESTCREATE03",
	"description": "my favorite computer"
	}
	}

Response	{
	"errors": [{
	"message": " The request to create a computer is not valid. You must provide a name and a path to the container where it should be created.",
	"suggestion": " Provide one of: distinguishedName, friendlyName, or both name and friendlyParentPath.",
	"stacktrace": ""
	}]

	}

Delete

The example below demonstrates deleting a contact:

URI	dra/domains/MyDomain.corp/contacts/delete

Payload	{
	"contactIdentifier": "Contact001"
	}

Response	{"errors":[]}

Get

The examples below demonstratde payload options for “group” get operations:

Basic

URI	dra/domains/MyDomain.corp/groups/get

Payload	{
	"groupIdentifier": "DRA Admins"
	}

Response	{
	"group": {
	"additionalAttributes": {},
	"class": "Group",
	"distinguishedName": "CN=DRA Admins,CN=Users,DC=MYDOMAIN,DC=CORP",
	"sAMAccountName": "DRA Admins"
	},
	"errors": []
	}

Requesting Specific Attributes

URI	dra/domains/MyDomain.corp/groups/get
Payload	{

	"groupIdentifier": "CN=Sales-SU,OU=Sales,DC=MYDOMAIN,DC=CORP", "attributes": ["samAccountName", "distinguishedName", "description", "mail", "hideFromAddressLists"] }

Response	{
	"group": {
	"mail": "Sales-SU_0009@MYDOMAIN.CORP",
	"hideFromAddressLists": true,
	"additionalAttributes": {},
	"description": "Description of: Sales-SU_0009",
	"class": "Group",
	"distinguishedName": " CN=Sales-SU,OU=Sales,DC=MYDOMAIN,DC=CORP ",
	"sAMAccountName": "Sales-SU_0009"
	},
	"errors": []
	}

NOTE:

When listing specific attributes, you can specify “members” to get a list of the members of the group. You can specify “memberOf” to get a list of the groups to which this group belongs. The request returns the results from a single query, without recursion.

Get a List of Groups in a Specific Domain

URI	dra/domains/MyDomain.corp/groups/get

Payload	{
	"enumerationOptions": {
	"containerDistinguishedName": "OU=SomeDept,DC=MYDOMAIN,DC=CORP"
	}
	}

Response	{
	"groups": [{
	"additionalAttributes": {},
	"description": "some description",
	"class": "Group",
	"friendlyName": " GROUP05",
	"friendlyPath": "MYDOMAIN.CORP/Sharon/GROUP05",
	"isManaged": true,
	"distinguishedName": "CN=GROUP05,OU=SomeDept,
	DC = MYDOMAIN,
	DC = CORP ",
	},
	Etc.
	],
	"resumeString": "",
	"totalNumberOfObjects": 12,
	"errors": []
	}

This example gets all groups in the path Domain\SomeDept. The default container for the domain/groups/get request is the domain root.

Update

The examples below demonstrate payload options for updating a user:

Basic

URI	dra/domains/MyDomain.corp/users/put

Payload	{
	"userIdentifier": "user05",
	"user": {
	"description": "modified description QQQ",
	"isDisabled": "true"
	}
	}

Response	{"errors":[]}

Update Office 365 Properties

URI	dra/domains/MyDomain.corp/users/put
Payload	{ "userIdentifier": "CN=Alfred O365-2,OU=Tax,DC=MyDomain,DC=CORP", "user": { "office365FullAccessAdd": ["CN=A AAATest,OU=Tax,DC=MyDomain,DC=CORP", "CN=Jim Bob-1,OU=Tax,DC=MyDomain,DC=CORP"], "office365FullAccessRemove": ["CN=AAA-Alfred Don1,OU=Tax,DC=MyDomain,DC=CORP", "CN=AATestCase6,OU=Tax,DC=MyDomain,DC=CORP"], "office365SendAsAdd": ["CN=Ron Jackson,OU=Tax,DC=MyDomain,DC=CORP", "CN=Eric Jones,OU=Tax,DC=MyDomain,DC=CORP"], "office365SendAsRemove": ["CN=A AAATest,OU=Tax,DC=MyDomain,DC=CORP", "CN=Jim Bob-1,OU=Tax,DC=MyDomain,DC=CORP"], "office365SendOnBehalfAdd": ["CN=AAA-Alfred Don1,OU=Tax,DC=MyDomain,DC=CORP", "CN=AATestCase6,OU=Tax,DC=MyDomain,DC=CORP"], "office365SendOnBehalfRemove": ["CN=Ron Jackson,OU=Tax,DC=MyDomain,DC=CORP", "CN=Eric Jones,OU=Tax,DC=MyDomain,DC=CORP"] } }
Response	{ "user": {

	"class": "User", "distinguishedName": "CN=AlfredO365-2,OU=Tax,DC=MyDomain,DC=CORP", "additionalAttributes": {}, "draServerAndPort": "MYDRASERVER:11192", "errors": [] } }

Enable (enable and disable)

URI	dra/domains/MyDomain.corp/users/disable/put dra/domains/MyDomain.corp/users/enable/put

Payload	{
	"userIdentifier": "user12"
	}

Response	{"errors":[]}

Unlock and Reset Password

These two options use the same DRA operation. In the REST code, we translate the endpoint location to the correct DRA parameters.

URI	dra/domains/MyDomain.corp/users/unlock/put

Payload	{
	"userIdentifier": "user12"
	}

Response	{
	"user": {
	"additionalAttributes": {},
	"class": "User"
	},
	"errors": []
	}

URI	dra/domains/MyDomain.corp/users/resetpassword/put

Payload	{
	"userIdentifier": "user12"
	}

Response	{
	"user": {
	"userPassword": "T9KBGhq~z",
	"additionalAttributes": {},
	"class": "User"
	},
	"errors": []
	}

The password is generated by the DRA server and returned to the client. If userPassword is passed in the payload, then that value is used, and no password is returned to the client.

Enable Email

URI	dra/domains/MyDomain.corp/users/enableemail/put

Payload	{
	"userIdentifier": "user05",
	"user": {
	"legacyExchangeDn": "/o=First/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Recipients/cn=User05",
	"mailNickname": "User05",
	"emailAddress": "user05email@mydomain.corp"
	}
	}

Response	{"errors":[]}

NOTE: When the mailNickname is not provided, the alias will be generated by the DRA server to be compliant with the configured alias naming policy. If the client supplies a mailNickname attribute that is out of compliance, the request will fail.

DisableEmail

URI	dra/domains/MyDomain.corp/users/disableemail/put
Payload	{

	"userIdentifier": "user05" }
Response	{"errors":[]}

Search (Enumeration)

The examples below demonstrate options payload and non-payload options when searching for objects:

Searching for a Specific Object Type in a Specific Domain

Default Search – No Payload

URI	dra/domains/MyDomain.corp/computers/get
Payload	{}

Response	{
	"computers": [{
	"trustedForDelegation": false,
	"groupMembershipCount": 1,
	"additionalAttributes": {},
	"isDisabled": false,
	"class": "Computer",
	"friendlyPath": "MYDOMAIN.CORP/Computers/HOUDVDR152",
	"isManaged": true,
	"distinguishedName": "CN=HOUDVDR152,CN=Computers,DC=MYDOMAIN,DC=CORP",
	"sAMAccountName": "HOUDVDR152$",
	"displayName": "HOUDVDR152$"
	},
	Etc.
	],
	"errors": []
	}

** *computers* can be replaced with *contacts, domains, draservers, dynamicdistributiongroups, equipmentresourcemailboxes, groups, ous, resourcemailboxes, roomresourcemailboxes, or users*. The returned attributes in the response depend on the specified object type. This default search looks in the root of the specified domain and does not search for any child containers. The search returns the first 250 objects found. The properties in the response are the default properties for that object type.

List all Objects in a Category in a Specific Container

URI	dra/domains/MyDomain.corp/computers/get

Payload	{
	"enumerationOptions": {
	"includeChildContainers": true,
	"containerDistinguishedName": "OU=My ou,DC=mydom,DC=corp"
	}
	}

Response	{
	"computers": [{
	"trustedForDelegation": false,
	"groupMembershipCount": 1,
	"additionalAttributes": {},
	"isDisabled": false,
	"class": "Computer",
	"friendlyPath": "MYDOMAIN.CORP/Computers/HOUDVDR152",
	"isManaged": true,
	"distinguishedName": "CN=HOUDVDR152,CN=Computers,DC=MYDOMAIN,DC=CORP",
	"sAMAccountName": "HOUDVDR152$",
	"displayName": "HOUDVDR152$"
	},
	Etc.
	],
	"resumeString": "xxxxxxxx",
	"totalNumberOfObjects": 322,
	"errors": []
	}

The *includeChildContainers* attribute is optional and defaults to false. To specify a particular container, provide the containerDistinguishedName attribute. This example returns the first 250 computers found in the "My OU" container. To request the remaining objects, send the query again with a *'containerEnumerationOptions'* object that specifies the resume string "xxxxxxxx". See Paging Examples an example.

List all Objects and Request Non-default Attributes

URI	dra/domains/MyDomain.corp/computers/get

Payload	{
	"attributes": [
	"accountThatCanAddComputerToDomain",
	"description",
	"displayName",
	"distinguishedName"
	]
	}

Response	{
	"computers": [{
	"trustedForDelegation": false,
	"additionalAttributes": {},
	"class": "Computer",
	"distinguishedName": "CN=HOUDVDR152,CN=Computers,DC=MYDOMAIN,DC=CORP",
	"displayName": "HOUDVDR152$"
	},
	Etc.
	],
	"resumeString": "xxxxxxxx",
	"totalNumberOfObjects": 322,
	"errors": []
	}

Search Using Filter, Enumeration Options and Non-Default Attributes

URI	dra/domains/MyDomain.corp/users/get

Payload	{
	"userAndFilter": {
	"distinguishedName": "*Wilson*"
	},
	"enumerationOptions": {
	"includeChildContainers": true,
	"containerDistinguishedName": "OU=My ou,DC=mydom,DC=corp"
	},

	"attributes": ["description", "displayName", "distinguishedName"] }

Response	{
	"users": [{
	"additionalAttributes": {},
	"class": "User",
	"distinguishedName": "CN=George Wilson,OU=Sales,DC=MYDOMAIN,DC=CORP",
	"displayName": "George Wilson",
	"description": "Sales Associate"
	},
	Etc.
	],
	"resumeString": "",
	"totalNumberOfObjects": 18,
	"errors": []
	}

This payload searches for users whose distinguished name contains 'Wilson'. The search will look in the OU 'My OU' and its children. The results will have 3 attributes: description, displayName and distinguishedName. This combination of JSON objects is what you will want to submit for many requests.

Search Across All Domains and All Object Types

Default Search – No Payload

URI	dra/managedObjects/get
Payload	None

Response	{
	"computers": [],
	"contacts": [],
	"domains": [{
	"additionalAttributes": {},
	"class": "Domain"
	}],
	"groups": [],
	"ous": [],

	"users": [], "resumeString": "", "totalNumberOfObjects": 1, "numberOfRowsReturned": 1, "isSearchFinished": true, "errors": [] }

As you can see, the default search across all domains and object types does not return any useful data. This search looks in the DRA root container and it will not search into any of the domains. Therefore, the results will probably be a set of empty object arrays.

Search for One Type of Object

URI	dra/managedObjects/get

Payload	{
	"userAndFilter": {
	"distinguishedName": "*Wilson*"
	},
	"enumerationOptions": {
	"includeChildContainers": true,
	"containerDistinguishedName": "OU=Sales,DC=mydom,DC=corp"
	},
	"attributes": ["description", "displayName", "distinguishedName"]
	}

Response	{
	"builtinContainers": [],
	"computers": [],
	"contacts": [],
	"containers": [],
	"domains": [],
	"groups": [],
	"ous": [],
	"users": [{
	"additionalAttributes": {},
	"class": "User",

```
      "distinguishedName": "CN=George Wilson,OU=Sales,DC=MYDOMAIN,DC=CORP",
      "displayName": "George Wilson",
      "description": "Sales Associate"
    }, {
      "additionalAttributes": {},
      "class": "User",
      "distinguishedName": "CN=Peter Wilson,OU=Sales,OU=USA,DC=MYDOMAIN,DC=CORP",
      "displayName": "George Wilson",
      "description": "Sales Associate"
    }, {
      "additionalAttributes": {},
      "class": "User",
      "distinguishedName": "CN= Wilson Jones,OU=Sales,OU=EMEA,DC=MYDOMAIN,DC=CORP",
      "displayName": "George Wilson",
      "description": "Sales Associate"
    }],
    "resumeString": "",
    "totalNumberOfObjects": 3,
    "numberOfRowsReturned": 3,
    "isSearchFinished": true,
    "errors": []
}
```


This search looks for users having ‘Wilson’ in the distinguished name. The search includes the container ‘my ou’ and all its children. Each object matching the result will contain 3 properties listed in the payload’s attributes object.

Search for Multiple Object Types

URI	dra/managedObjects/get

Payload	{
	"computerOrFilter": {
	"managedBy": "CN=Wilson Jones,OU=Sales,OU=EMEA,DC=MYDOMAIN,DC=CORP"
	},
	"groupOrFilter": {
	"managedBy": "CN=Wilson Jones,OU=Sales,OU=EMEA,DC=MYDOMAIN,DC=CORP"
	},

	"ouOrFilter": {
	"managedBy": "CN=Wilson Jones,OU=Sales,OU=EMEA,DC=MYDOMAIN,DC=CORP"
	},
	"enumerationOptions": {
	"includeChildContainers": true
	},
	"attributes": ["distinguishedName"]
	}

Response	{
	"builtinContainers": [],
	"computers": [{
	"additionalAttributes": {},
	"class": "Computer",
	"distinguishedName": "CN=COMP123,OU=Computers,DC=MYDOMAIN,DC=CORP",
	},
	Etc.
	],
	"contacts": [],
	"containers": [],
	"domains": [],
	"groups": [{
	"additionalAttributes": {},
	"class": "Group",
	"distinguishedName": "CN=SomeGroup,OU=Sales,OU=Marketing,DC=MYDOMAIN,DC=CORP",
	},
	Etc.
	],
	"ous": [{
	"additionalAttributes": {},
	"class": "OrganizationalUnit",
	"distinguishedName": "OU=Sales,DC=MYDOMAIN,DC=CORP",
	},
	Etc.

	], "users": [], "resumeString": "", "totalNumberOfObjects": 3, "numberOfRowsReturned": 3, "isSearchFinished": true, "errors": [] }

This search looks for computers, groups, and OUs having the managedBy field set to a specific distinguished name. The enumeration options specify no specific container and that the search should search all children. Take care when performing searches across all containers for multiple object types. See Performance Considerations for more information. xxOrfilter can be prefixed by any of these object types: *builtinContainer, computer, contact, container, ddg, domain, equipmentMailbox, group, ou, roomMailbox, or user.*

Search Using an LDAP Query

URI	dra/ldapObjects/get

Payload	{
	"enumerationOptions": {
	"includeChildContainers": true,
	"containerDistinguishedName": "cn=Users,DC=DRDOM610,DC=lab",
	"objectsPerResponse": "5",
	"ldapEnumerationOptions": {
	"ldapQueryString": "(&(objectClass=User)(!(email=*)))"
	}
	},
	"attributes": ["friendlyName",
	"distinguishedName",
	"mail",
	"title",
	"physicalDeliveryOfficeName",
	"userPrincipalName",
	"hasMailbox"
	]
	}

Response	{

```
  "builtinContainers": [],
  "computers": [],
  "contacts": [],
  "containers": [],
  "domains": [],
  "dynamicDistributionGroups": [],
  "groups": [],
  "ous": [],
  "users": [{
      "additionalAttributes": {},
      "class": "User",
      "distinguishedName": "OU=Sales,DC=MYDOMAIN,DC=CORP",
    },
    Etc.
  ],
  "resumeString": "",
  "totalNumberOfObjects": 3,
  "numberOfRowsReturned": 3,
  "isSearchFinished": true,
  "errors": []
}
```


Search with a Filter for Virtual Attributes

URI	dra/ldapObjects/get

Payload	{
	"enumerationOptions": {
	"includeChildContainers": true,
	"containerDistinguishedName": "cn=Users,DC=DRDOM610,DC=lab",
	"objectsPerResponse": "5",
	"ldapEnumerationOptions": {
	"ldapQueryString": "(&(objectClass=User)(!(email=*)))",
	"vaQueryString": "<VAQUERY><OR><USER><ATTRIBUTE>VA01</ATTRIBUTE><CONDITION>IS</CONDITION><VALUE>Z</VALUE></OR></VAQUERY>",

	} }, "attributes": ["friendlyName", "distinguishedName", "mail", "title", "physicalDeliveryOfficeName", "userPrincipalName", "hasMailbox"] }

Response	{
	"builtinContainers": [],
	"computers": [],
	"contacts": [],
	"containers": [],
	"domains": [],
	"dynamicDistributionGroups": [],
	"groups": [],
	"ous": [],
	"users": [{
	"additionalAttributes": {},
	"class": "User",
	"distinguishedName": "OU=Sales,DC=MYDOMAIN,DC=CORP",
	},
	Etc.
	],
	"resumeString": "",
	"totalNumberOfObjects": 3,
	"numberOfRowsReturned": 3,
	"isSearchFinished": true,
	"errors": []
	}