DRA Rest Extensions Technical Reference

Getting Started

The DRA Rest Extensions package has 3 features:

- **NetIQ DRA Host Service**: hosts the WCF Rest Services used to communicate with the DRA Administration Service. This service must run on a computer with the DRA Administration Service installed.
- **DRA REST Service and Endpoints**: provides the RESTful interfaces that allow non-DRA clients to request DRA operations. This service must run on a computer with either a DRA console or the DRA Administration Service installed.
- **PowerShell Extensions**: provides a PowerShell module that allows non-DRA clients to request DRA operations using PowerShell cmdlets.

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 Extensions will use these user accounts:

- **DRA Host Service account**: Any account with the Run As Service property enabled. The installer will configure the service to run as LocalSystem. To specify a different account, modify the service configuration after the installation completes. This account is not used to make any requests to the DRA server.
- **PowerShell client**: The user running DRA REST PowerShell cmdlets must be a local administrator. This is required to perform tasks such as importing the DRA REST modules. The PowerShell user must be in a domain that is trusted by the DRA Server domain. The PowerShell client user must be delegated assistant admin powers in DRA.
- **DRA Admin user**: This is the account that has been delegated the roles and powers required to perform the requested operation within DRA. If this user is different than the client user, the client must send Connection Parameters with each request that specify the credentials for the admin user.

Available in This Release

This release provides:

- Interfaces for performing create, update, delete and read operations on computers, contacts, groups, organizational units and users.
- Interfaces for getting a list of objects. The search can specify a specific domain and container or it can search all managed domains.
- Interfaces for requesting DRA operations using the existing COM interface. These interfaces allow the non-DRA clients to access all available DRA operations, even if the operation does not have a RESTful endpoint implementation.
- Ability to customize the data returned in read and enumeration requests.

Understanding Requirements

<u>For this managed release, all extension features should be installed on the same DRA Server.</u> The production release will allow you to install specific features on multiple machines.

In the production release, the features will have these software requirements:

DRA Host Service	• DRA Administration Server, 8.6 SP2 or later • IIS 7.0, 7.5 or 8.0
DRA REST Service and Endpoints	• A DRA console or the DRA Administration service, 8.6 SP2 or later • IIS 7.0, 7.5 or 8.0 • Valid authentication certificate
PowerShell Extensions	• PowerShell 2.0 or later

All of the features have these software requirements:

Operating System	• Microsoft Windows Server 2008 • Microsoft Windows Server 2008 R2 • Microsoft Windows Server 2012
Enabled Software and Support	• Microsoft .Net Framework 4.0.3 • Microsoft Exchange 2007 or later (optional)

Installation

In a production environment, you may want to distribute the REST Service and endpoints across multiple web servers to load balance the activity. For this managed preview, install all features on a DRA server.

The **NetIQ DRA Host Service** and **DRA REST endpoints** can be installed on any DRA server running DRA 8.6 SP2 or later.

If the **REST endpoints** computer is not in a trusted domain, the client will need to provide DRA assistant admin credentials for every request. Also, PowerShell is unable to communicate to a DRA server in an untrusted domain due to PowerShell's lack of support for alternate credentials.

The **PowerShell Extensions** can be installed on a stand-alone computer and communicate with a remote DRA Server. The remote DRA server must have both the Host and DRA Endpoints installed.

Both the Host and DRA Rest Services communicate with the web server using SSL and require a signed certificate. When either of these features is selected, the installer queries the local certificate store and presents the list of valid certificates. You cannot use the certificate labeled "(default)".

The Host Service and the DRA Rest Service each require a communication port. The installer will supply default values for these ports that can be changed, if needed. The DRA Rest Service default port number is 8755. The Host Service default port number is 11192.

The user running the installer must be a local administrator or supply credentials to elevate the installer with administrator credentials.

Log File Locations

The installer will create logs in two places.

- The MSI log is written to %USERPROFILE% of the installing user.
- Logs related to installer custom actions are written to %LOCALAPPDATA%\NetIQ\DRA\Logs\Setup. The files are named MSIActions-[date]-[time].log.

The Host and Rest Services write to %LOCALAPPDATA%\NetIQ\DRA\Logs\[service name]

- By default, the services run using the built-in account LocalSystem.
- For the LocalSystem account, %LOCALAPPDATA% resolves to %Systemroot%\SysWOW64\config\systemprofile\AppData.
- By default, the logs will roll over to a new log when the current log reaches 15MB. The size and number of generations are configured in [InstallLocation]\NetIQ.DRA.DRAHostService.exe.config for the Host service and [InstallLocation]\NetIQ.DRA.DRARestService.exe.config for the Rest Service.

Understanding the DRA REST Endpoints and APIs

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.

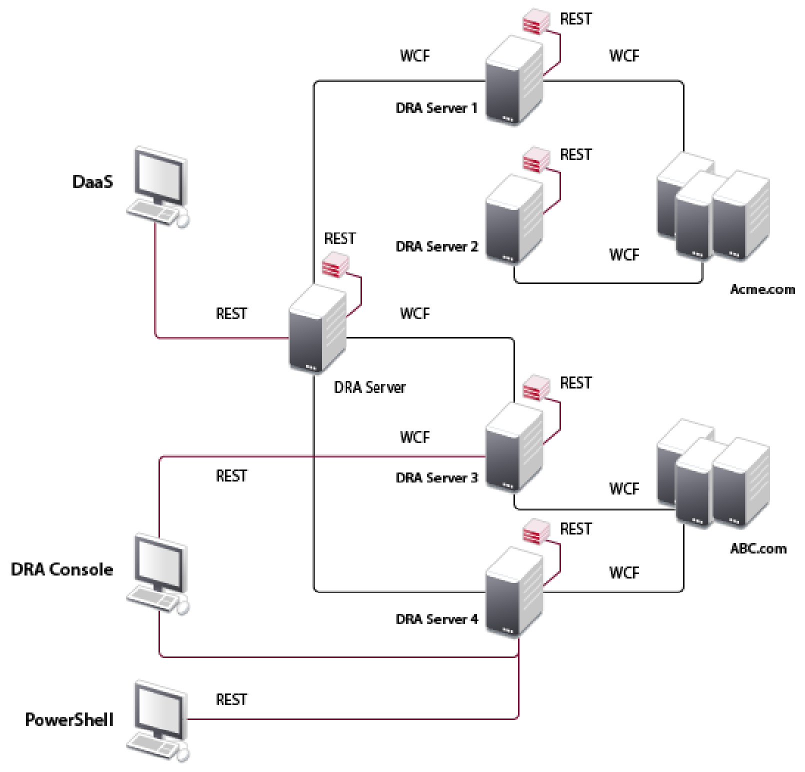
Architecture

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.

The client sends the HTTP over SSL request to the DRA Rest Service, with or without specifying the DRA server. The DRA Rest Service sends one or more DRA Server operation requests. If the draServerNameAndPort parameter is supplied, the requests are sent to the named machine:port. Otherwise, the requests are sent to the Host Service running on the local machine.

The DRA Host Service communicates with the DRA Administration Service using WCF over SSL and COM. The COM objects generated by the DRA Administration Service are translated back to JSON format and the response is sent back to the REST client.

Below is a picture of that workflow:



Authentication

There can be two identities used for any REST request:

- **Client User**: This is the user making the request. This user is authenticated to IIS via Basic Authentication, Windows Integration, or by supplying a signed certificate. When using Basic Authentication, the request header must contain the client user's credentials. When using certificates, the request must include a client certificate that is loaded in the local store on the client. The client certificate must be signed by an authority that is trusted by the DRA Host server.
- **DRA Admin User**: This is the user who has been delegated DRA powers to perform the requested operation. If the DRA Admin is not the same user as the client user, the REST request must include the DRA Admin user'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/or 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:

https://localhost:8755/Console/help

There are several tools you can use for exploring REST interfaces, such as SmartBear's SoapUI or Chrome's Dev HTTP Client. 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

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/or 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 admin 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	Is the name of the computer running the DRA Rest Service
port	is 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 Extensions* 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 several 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 AD 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 misspelled 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	"user": {

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

Where possible, the endpoint will supply a default value for an attribute, if it is required by DRA. 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*.

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"

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 members:

draServerNameAndPort	Specifies a particular DRA server and port whose Host service will perform the DRA operation.
username	The full user name 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. See the section Customizing the Default Query Lists for information on how to configure a different default list of properties.

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 url is used as the container and if there is no domain in the url then the module=accounts special container is searched (basically the container that corresponds to the 'All My Managed Objects' node in the win32 console). For example:

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 behavior 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 requiest, see Search Operation Types.

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.

ldapQueryString	Required for ldap searches. Specifies a complete LDAP query string.
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 the ldap-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
	IS-NOT
	STARTS-WITH
	PRESENT
	
	The virtual attribute query string is a continuation of the LDAP query string. So, the OR 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 LDAP Query Search for more information.

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

Endpoint Request Examples:

Verify that the Rest Service is up and functioning:

This request will return static HTML. Note: This is the only interface that uses the Http GET verb.

URI	https : // myDRAserver:8755/rest
Payload	None. (No payload is allowed with a GET request.)
Respons e	The response is static HTML: `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"> <html> <head>`

	`<title>`
	`NetIQ Directory and Resource Administrator</title></head> <body><h1>NetIQ`
	`Directory`
	`and Resource Administrator</h1><h3>Implementation Version:`
	`8.60.2.360</h3><h3>Product`
	`Documentation: <a`
	`href="https://www.netiq.com/support/dra/extended/documentation.asp">Documenta`
	`tion</a>`
	`</h3><h3>Product URI: <a href="https://www.netiq.com/products/directory-`
	`resource-administrator">Product page</a></h3></body></html>`

This request only verifies that the Rest Service is functional on the server named in the URI. It does not make any call to the Host service or request an operation from the DRA Server.

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,`
	`    "hostServicePort": 11192`
	`  }]`
	`}`

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}s/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 of the objects found by your search, even if the results contain several thousand 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 and Or filters are: computer, contact, donmain, 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 of the criteria using an AndFilter.
Attributes	Optional. Specifies the list of fields you want 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, etc. 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 of 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 of the filter criteria. For example:

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

will return all user objects in the specified path whose location matches the AndFilter AND whose managedBy field contains George OR whose department contains 'sales'.

<u>**Important Note**</u>: 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.

Initial Request	"enumerationOptions": {
	  "containerDistinguishedName": "OU=Accounting,DC=MYDOMAIN,DC=CORP",
	  "resumeString": ""
	}

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

2nd Request	"enumerationOptions": {
	  "containerDistinguishedName": "OU=Accounting,DC=MYDOMAIN,DC=CORP",
	  "resumeString": "CN=Accounting-DG,OU=Accounting,DC=MYDOMAIN,DC=CORP"
	}

Server Response	{
	  ...
	  "resumeString": "",
	  "totalNumberOfObjects": 294,
	  "numberOfRowsReturned": 44,
	  "isSearchFinished": true
	}

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.

Initial Request	"enumerationOptions": {
	  "containerDistinguishedName": "OU=Accounting,DC=MYDOMAIN,DC=CORP",

	"objectsPerResponse": 15,
	"resumeString": "",
	"pagedEnumerationOptions ": {
	"startRow": 1,
	"cacheId": ""
	}
	}

Server Response	{
	...
	"resumeString": "{214A15E1-44D4-460D-9CC4-D8CA04EFDE6E}",
	"totalNumberOfObjects": 294,
	"numberOfRowsReturned": 15,
	"isSearchFinished": false
	}

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

LDAP Query Search

The LDAP query endpoint is:

- dra/ldapQuery/get

This endpoint allows you to provide an LDAP query to search for objects. The results will contain several object arrays: one for each type of object managed by DRA.

LDAP Query Payload

The payload for an LDAP query request defines the filter for the search as well as how much data to return in 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 of the objects found by your search, even if the results contain several thousand objects.

The payload can include the following objects:

Enumeration Options	Required. You must specify the parameter ldapEnumerationOptions LDAP query string parameter. Generally, you will also want to specify at least the *containerDistinguishedName* parameter, unless you want the server to search in the root. See Enumeration Options for a list of options you can specify.
Attributes	Optional. Specifies the list of fields you want returned in the results. See Attributes List for more information.
Connection	Optional. Specifies the name of the DRA server that should perform the

Parameters	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.

The results contain the flag *isSearchFinished* that indicates whether the server has more results to return to the client. The search results will contain two attributes that you will use 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.

Handling multiple result pages

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.

Initial Request	`{`
	`  "enumerationOptions": {`
	`    "containerDistinguishedName": "OU=Accounting,DC=MYDOMAIN,DC=CORP",`
	`    "resumeString": "",`
	`    "ldapEnumerationOptions ": {`
	`      "ldapQueryString": "(&(objectClass=User)(!(email=*)))"`
	`    }`
	`  }`
	`}`
Server Response	`{`
	`  ...`
	`  "resumeString": "4f453ef8-679f-11e4-8d77-0050568e0b4a",`
	`  "totalNumberOfObjects": 294,`
	`  "numberOfRowsReturned": 250,`
	`  "isSearchFinished": false`
	`}`
2nd Request	`{`
	`  "enumerationOptions": {`
	`    "containerDistinguishedName": "OU=Accounting,DC=MYDOMAIN,DC=CORP",`
	`    "resumeString": "4f453ef8-679f-11e4-8d77-0050568e0b4a",`
	`    "ldapEnumerationOptions ": {`
	`      "ldapQueryString": "(&(objectClass=User)(!(email=*)))"`
	`    }`
	`  }`
	`}`
Server Response	`{`
	`  ...`
	`  "resumeString": "",`
	`  "totalNumberOfObjects": 294,`
	`  "numberOfRowsReturned": 44,`
	`  "isSearchFinished": true`
	`}`

Performance Considerations

TBD – discuss need to search on properties included in the cache

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, the client property names are more user-friendly. The property name maps cannot be changed or edited.

The baseObjectPropertyNameMap describes properties common to all of 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 *NetIQ DRA Rest Service* will stop.

Integration using the DRA Varset and .NET

DRA provides a .NET interface that will allow you to perform any DRA operation, provided the client user has been delegated the necessary powers in DRA. This interface works with the existing DRA Varset, which is a proprietary text format for communicating with the DRA server.

To use approach, you would:

1. Enable maximum logging on a DRA server.
2. Perform the operation you intend to automate using the DRA console.
3. Open the file MCSAdminSvc.nq1 in %LOCALAPPDATA%\NetIQ\Logs\Server using the NetIQ Logviewer application.
4. Search for the operation, for example: *VACreate*.
5. Select all the lines for the incoming operation that start with "In.". Right-click and select the *Show Full Text* option. For VACreate, you would select lines like this:
```
[In]  <VT_EMPTY>
[In.AttributeName]  <VT_BSTR> NVA04
[In.Description]  <VT_BSTR> joel02
[In.LocaleID]  <VT_I4> 1033  (     0X409)
[In.MultiValued]  <VT_BOOL> 0
[In.OperationName]  <VT_BSTR> VACreate
[In.VarType]  <VT_I4> 6  (       0X6)
```

6. Copy all the text in the displayed dialog and save it to a file.
7. Reformat these lines into JSON values represented by key-value pairs. For example, if the log shows:
```
[In.AttributeName]  <VT_BSTR> NVA04
```
the JSON equivalent would be:
```
"AttributeName":"NVA04"
```

8. Edit any values in the file, as needed. For example, change the object distinguished name.
9. Use this interface:

URI	https : //myDRAserver:8755/dra/operations/varset/post

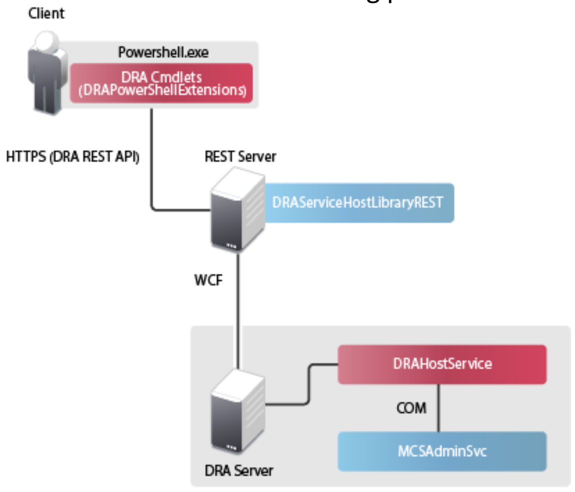
Payload	{
	  "varsetData": {
	    "AttributeName": "MyVirtualAttribute",
	    "Description": "some virtual attribute",
	    "LocaleID": 1033,
	    "MultiValued": false,
	    "OperationName": "VACreate",
	    "VarType": 6
	  }
	}

Response	{
	  "returnedVarset": {
	    "ClientName": "CN=Administrator",
	    "ClientPath": "OnePoint://CN=myUser,CN=Users,DC=MYDOMAIN,DC=LAB",
	    "Errors": null,
	    "Errors.LastError": 0,

```
        "Errors.NumErrors": 0,
        "Result": "All Done",
        "Warnings": null,
        "Warnings.LastWarning": 0,
        "Warnings.NumWarnings": 0
    },
    "errors": []
}
```

Working With the PowerShell Extensions

The PowerShell extensions implement the DRA Rest interfaces, allowing you to integrate with the DRA server using PowerShell cmdlets. The following picture shows the path that a DRA request will take:



Consider the following requirements when deciding where to install the DRA REST PowerShell module:

- The computer running the PowerShell cmdlets must be in a domain that is trusted by the domain of the DRA server.
- The user running the cmdlets must be in the local administrators group on the computer running the PowerShell cmdlets.

After importing the DRA Rest PowerShell module, you can use all the familiar PowerShell utility commands to list the available interfaces and show help for parameters and examples. The DRA Rest PowerShell cmdlets return a PowerShell object that can be piped into a subsequent PowerShell call. In this way, you can string requests together. For example the client could request a list of all DRA servers, pipe the result into a subsequent request that iterates over the list and selects a server to perform a DRA operation.

The DRA REST Services installer will copy the PowerShell cmdlets and binaries to the local computer. To work with the DRA REST cmdlets, you must first import the module into the environment. You can either import the module at the beginning of each session, or configure the PowerShell profile of the client’s user to always import the module.

Using the Cmdlets

To start using the DRA Extension cmdlets, you must first import the module in your session:

```
      import-module NetIQ.DRA.DRAPowerShellExtensions
```

To list the available interfaces and to see the parameters required for each cmdlets, you can use the following PowerShell utility command:

```
      get-command -module NetIQ.DRA.DRAPowerShellExtensions
```

To list the parameters required for any cmdlets, use the get-help cmdlets, such as:

```
      get-help Get-DRAUser
```

The get-help options –Full, -Detailed, and -Examples are supported. (-Online is not implemented.)

If you are using self-signed certificates for the REST service, you will need to override the check that the certificate is valid. On each command, you will need to include the `–IgnoreCertificateErrors` parameter. To also suppress the confirmation message, add the `–Force` parameter.

Properties Parameter Details

Formatting the Parameter

For any DRA request that allows you to set properties on an object (e.g., create or update requests), you will pass the property values as a single string on the Properties parameter. The parameter is formatted as a collection of name=value pairs, separated by semi-colons. For example:

```
      -Properties "Description=some 
description;EmployeeID=z4563;Department=Accounting"
```

When the property value contains a semi-colon, escape the semi-colon with a second semi-colon. For example:

```
      -Properties "Description=some description;; more 
      description...end;EmployeeID=z4563;Department=Accounting"
```

When the property has multiple values, separate each value with a comma. For example:

```
      -Properties "ConditionalStateOrProvince=TX,IA,CA"
```

When the values of a multi-valued property value contain a comma, enclose each value in quotes. This is very common for object properties that expect a list of distinguished names. The example shows 2 properties: Description and AcceptMessagesFrom. AcceptMessagesFrom expects a list of distinguished name values. The property value has each distinguished name surrounded by quotes and delimited by commas. Note that when a command-line parameter has embedded quotes, the quotes need to be escaped using the back tick (`).

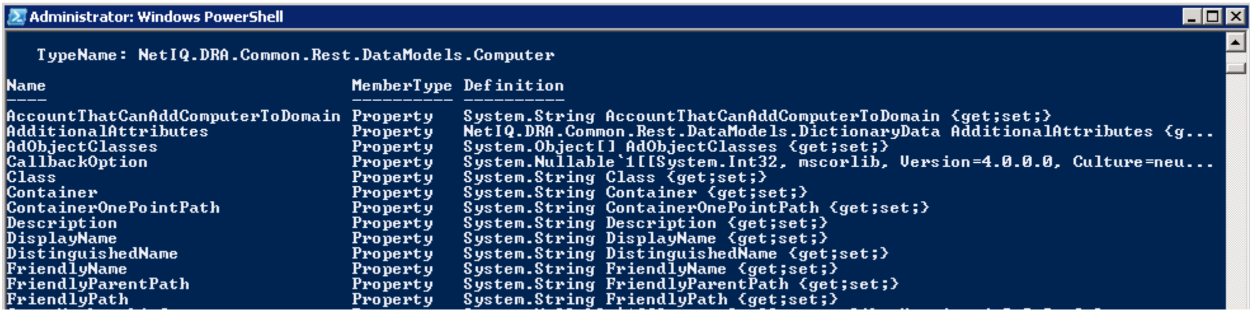
```
      -Properties "Description=some 
      text;AcceptMessagesFrom=`"CN=George,OU=Accounting,DC=MyDomain,DC=
      corp`", 
      `"CN=Contact022,OU=HR,DC=MyDomain,DC=corp`",`"CN=PSGroup01,OU=Acc
      ounting,DC=MyDomain,DC=corp`""
```

Getting the list of available properties

Each DRA object has many properties. To see the property names available for any object, you can perform a Get-DRA cmdlet and query the results using PowerShell's Get-Member cmdlet. Here are the steps to get the list of defined properties for a computer.

1. Run a Get-DRAComputer cmdlet and set the returned object to a local variable. For example:
   ```
   $draObj = Get-DRAComputer –Domain mydomain.corp -Identifier
   "myComputer"
   ```
2. Using the local variable created in step 1, call the Get-Member cmdlet. For example:
   ```
   Get-Member -InputObject $draobj -MemberType Property
   ```

Here is a picture of the results:



```
   TypeName: NetIQ.DRA.Common.Rest.DataModels.Computer

Name                                MemberType Definition
----                                ---------- ----------
AccountThatCanAddComputerToDomain Property   System.String AccountThatCanAddComputerToDomain {get;set;}
AdditionalAttributes              Property   NetIQ.DRA.Common.Rest.DataModels.DictionaryData AdditionalAttributes {g...
AdObjectClasses                   Property   System.Object[] AdObjectClasses {get;set;}
CallbackOption                    Property   System.Nullable`1[[System.Int32, mscorlib, Version=4.0.0.0, Culture=neu...
Class                             Property   System.String Class {get;set;}
Container                         Property   System.String Container {get;set;}
ContainerOnePointPath             Property   System.String ContainerOnePointPath {get;set;}
Description                       Property   System.String Description {get;set;}
DisplayName                       Property   System.String DisplayName {get;set;}
DistinguishedName                 Property   System.String DistinguishedName {get;set;}
FriendlyName                      Property   System.String FriendlyName {get;set;}
FriendlyParentPath                Property   System.String FriendlyParentPath {get;set;}
FriendlyPath                      Property   System.String FriendlyPath {get;set;}
```

When the Definition shows System.String[], then the property accepts multiple values.

Examples:

Get information about a specific user:

```
Get-DRAUser -Identifier "CN=Alexander
Pompeani,OU=Accounting,OU=Houston,DC=keystone,DC=local"
-Domain "MyDomain.corp" -Attributes
"DistinguishedName,EmployeeID,Department"
-DRAHostServer "myDRAserver"
-DRARestPort 9999
-DRAHostPort 9999
-AssistantAdmin "myDomain\\someAA"
-AssistantAdminPassword "xxxyyy"
```

Where:	**Specifies**:
Get-DRAUser	The name of the cmdlet
Domain	The parameter to specify the domain of the user in the fully qualified domain name format.
Identifier	The parameter to specify the user. You can use the distinguished name, friendly name or display name. Note that if you use the display name, the REST endpoint will make an additional call to the DRA server to resolve the display name to the distinguished name.
Attributes	Lists the names of properties to return, separated by commas. This request will return the distinguished name, employeeID and department. If this attribute is not specified, a default set of

	properties will be returned. See here for more information about the default list of properties.
DRAHostServer	Optional parameter to specify a remote computer running the DRA REST services. The default value is '*localhost*'.
DRARestPort	Optional parameter that specifies the port where the DRA REST service listens for requests. If the port is not specified, the request will go to port 8755.
DRAHostPort	Optional parameter that specifies the port where the DRA Host service listens for requests. If the port is not specified, the request will go to port 11192.
AssistantAdmin	Optional parameter that specifies the DRA assistant administrator user whose credentials should be used for the DRA Server operation. The default is to use the credentials of the PowerShell client user. If this is specified, the AssistantAdminPassword must also be specified.
AssistantAdminPassword	Optional parameter that specifies the password for the DRA assistant administrator. Required if AssistantAdmin is specified.

Add a new computer:

```
Add-DRAComputer -Domain MyDomain.corp -Properties
"DistinguishedName=cn=COMPUTER123,OU=Accounting,DC=MyDomain,DC=co
rp;Description=Computer for powershell;TrustedForDelegation=true"
```

where:	**specifies**:
Add-DRAComputer	The name of the cmdlet
Domain	The parameter to specify the domain of the user in the fully qualified domain name format.
Properties	The list of properties to set for the new computer. You can specify the target container and name for the new object using the same list of properties described in various create examples for computer objects in the Appendix.

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. (ENG331778)

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.

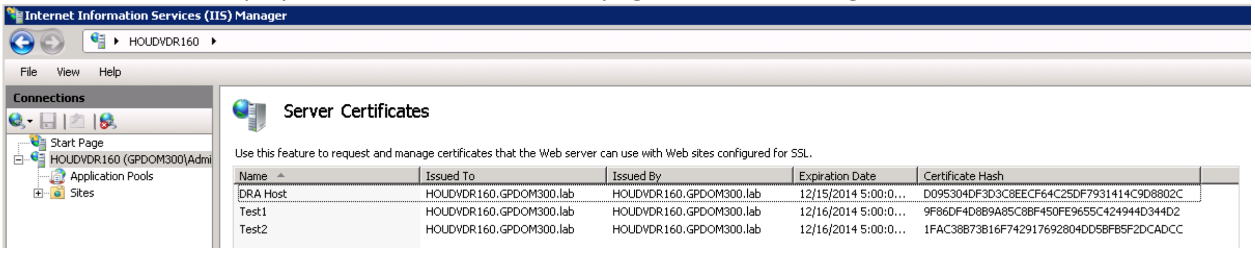
Troubleshooting

Handling Certificates for the DRA REST Extensions

The DRA endpoint service requires a certificate binding on the communication port. During installation, the installer will perform the commands for binding the port to the certificate. The purpose of this section is to describe how to validate the binding and how to add or remove a binding, if needed.

Basic Information

Default Endpoint Service Port: 8755
App ID for DRA REST Extensions: 8031ba52-3c9d-4193-800a-d620b3e98508
Certificate Hash: Displayed on the SSL Certificates page of the IIS Manager



Use this feature to request and manage certificates that the Web server can use with Web sites configured for SSL.

Name	Issued To	Issued By	Expiration Date	Certificate Hash
DRA Host	HOUDVDR160.GPDOM300.lab	HOUDVDR160.GPDOM300.lab	12/15/2014 5:00:0...	D095304DF3D3C8EECF64C25DF7931414C9D8802C
Test1	HOUDVDR160.GPDOM300.lab	HOUDVDR160.GPDOM300.lab	12/16/2014 5:00:0...	9F86DF4D8B9A85C8BF450FE9655C424944D344D2
Test2	HOUDVDR160.GPDOM300.lab	HOUDVDR160.GPDOM300.lab	12/16/2014 5:00:0...	1FAC38B73B16F742917692804DD5BFB5F2DCADCC

Checking For Existing Bindings

In a CMD window, run this command:

```
      netsh http show sslcert
```

This will display a list of certificate bindings for this computer. Look through the list for the App ID of the DRA REST Extensions. The port number should match the config port. The certificate hash should match the certificate hash displayed in IIS Manager.

```
      IP:port                     : 0.0.0.0:8755
      Certificate Hash            :
      d095304df3d3c8eecf64c25df7931414c9d8802c
      Application ID              : {8031ba52-3c9d-4193-800a-d620b3e98508}
      Certificate Store Name      : (null)
      Verify Client Certificate Revocation    : Enabled
      Verify Revocation Using Cached Client Certificate Only    :
      Disabled
      Usage Check    : Enabled
      Revocation Freshness Time : 0
      URL Retrieval Timeout    : 0
      Ctl Identifier              : (null)
      Ctl Store Name              : (null)
      DS Mapper Usage    : Disabled
      Negotiate Client Certificate    : Disabled
```

Removing a binding

To remove an existing binding, enter this command in a CMD window:

```
      netsh http delete sslcert ipport=0.0.0.0:9999
```

Where: "9999" is the port number to remove.

The netsh command will display a message indicating that the SSL Certificate was successfully removed.

Adding a binding

To add a new binding, enter the following command in a CMD window:

```
      netsh http add sslcert ipport=0.0.0.0:9999 certhash=[HashValue]
      appid={8031ba52-3c9d-4193-800a-d620b3e98508}
```

Where: 9999 = the port number of the endpoint service
[HashValue] = the Certificate Hash value displayed in IIS Manager

Handling Errors from the DRA server

EnableEmail returns operation failed

When creating a mail-enabled object or calling one of the EnableEmail endpoints, you might get an error back from the DRA server such as "*Server failed to complete the requested operation workflow successfully. Operation UserEnableEmail failed*". This can be caused by including a **mailNickname** property in the payload that does not conform to the policy defined on the server.

Remove the **mailNickname** property from the payload and let the DRA server generate the email alias value according to the defined policy.

Every PowerShell Command Results In PSInvalidOperation Error

When you are the DRA REST service is bound to a self-signed certificate, the PowerShell cmdlets will return the following error:

```
Get-DRAServerInfo : One or more errors occurred.
An error occurred while sending the request.
The underlying connection was closed: Could not establish trust
relationship for the SSL/TLS secure channel.
The remote certificate is invalid according to the validation
procedure.
```

On each command, you will need to include the `-IgnoreCertificateErrors` parameter. To also suppress the confirmation message, add the `-Force` parameter.

WCF Trace Logging

If your REST requests are resulting in errors that cannot be resolved by reading the REST or HOST service logs, you might need to raise the level of WCF's trace logging to see details about how the request is travelling through the WCF layer. The volume of data generated by this level of trace can be significant. So, the shipped logging level is set to "Critical, Error".

An example of when this might be useful is if the requests are resulting in null value exceptions though you are sending the objects in the payload. Another case would be if the REST or Host service is becoming unresponsive.

To increase the WCF trace logging, you need to edit the configuration file for the service that is under scrutiny. Payload exceptions are likely to be evident from reviewing the WCF trace log for the REST service.

Steps to enable logging

1. In Windows File Explorer, navigate to the DRA Extensions installation folder. Typically, this will be `C:\Program Files (x86)\NetIQ\DRA Extensions`.
2. Open the appropriate configuration file with Notepad. The configuration file would be either *NetIQ.DRA.RestService.exe.config* or *NetIQ.DRA.HostService.exe.config*.
3. Locate the <*source*> element in the following xml path: <system.diagnostics><sources>.
4. In the *source* element change the *switchValue* attribute value to "Verbose, ActivityTracing".
5. Save the file.
6. Restart the service.

Reviewing the logs

The WCF trace data is written in a proprietary format. You can read the traces.svslog using the SvcTraceViewer.exe utility. You can find more information about this utility here: https://msdn.microsoft.com/en-us/library/ms732023(v=vs.110).aspx .

Appendix A – Rest Endpoint Examples

Computer

Create

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

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": "" }] }

Get

Get a list of computers in a specific domain

URI	dra/domains/MyDomain.corp/computers/get
Payload	{ "enumerationOptions": { "containerDistinguishedName": "OU=SomeDept,DC=MYDOMAIN,DC=CORP" } }
Response	{ "computers": [{ "additionalAttributes": {}, "isDisabled": true, "class": "Computer", "friendlyPath": "MYDOMAIN.CORP/Sharon/COMPUTER05", "isManaged": true, "distinguishedName": "CN=COMPUTER05,OU=SomeDept, DC = MYDOMAIN, DC = CORP ", "sAMAccountName": "COMPUTER05$", "winNTPath": "MYDOMAIN\\COMPUTER05$" }, Etc.], "resumeString": "", "totalNumberOfObjects": 12,

	"errors": [] }

This example gets all computers in the path Domain\Computers. The default container for the domain/computers/get request is the domain root. To search for computers across multiple domains, see Enumeration examples.

Get information about a specific computer

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

Payload	{
	  "computerIdentifier": "COMPUTER05"
	}
	        Or
	{
	  "computerIdentifier": "CN=COMPUTER05,OU=SomeDept,DC=MYDOMAIN,DC=CORP"
	}

Response	{
	  "computer": {
	    "trustedForDelegation": false,
	    "groupMembershipCount": 1,
	    "additionalAttributes": {},
	    "description": "modified description 33",
	    "isDisabled": true,
	    "class": "Computer",
	    "friendlyPath": "MYDOMAIN.CORP/Sharon/COMPUTER05",
	    "isManaged": true,
	    "distinguishedName": "CN=COMPUTER05,OU=SomeDept,
	    DC = MYDOMAIN,
	    DC = CORP ",
	    "sAMAccountName": "COMPUTER05$"
	  },
	  "errors": []
	}

Get specific computer specifying attributes to retrieve

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

Payload	{
	  "computerIdentifier": "COMPUTER05",
	  "attributes": [
	    "accountThatCanAddComputerToDomain",
	    "description",
	    "displayName",
	    "distinguishedName",
	    "friendlyPath",
	    "friendlyName",
	    "friendlyParentPath",
	    "sAMAccountName",
	    "trustedForDelegation",

	"objectCategory"] }

Response	{
	"computer": {
	"trustedForDelegation": false,
	"accountThatCanAddComputerToDomain": "Unimplemented",
	"additionalAttributes": {},
	"description": "modified description 33",
	"class": "Computer",
	"friendlyName": "COMPUTER05",
	"friendlyParentPath": "MYDOMAIN.CORP/Sharon",
	"friendlyPath": "MYDOMAIN.CORP/Sharon/COMPUTER05",
	"distinguishedName": "CN=COMPUTER05,OU=SomeDept,DC=MYDOMAIN,DC=CORP",
	"sAMAccountName": "COMPUTER05$"
	},
	"errors": []
	}

Delete

Delete

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

Payload	{
	"computerIdentifier": "COMPUTER05"
	}

Response	{"errors":[]}

Delete from recycle bin

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

Payload	{
	"computerIdentifier ": "cn=COMPUTER04,OU=NetIQRecycleBin,DC=MyDomain,DC=corp"
	}

Response	{"errors":[]}

Move

URI	dra/domains/MYDOMAIN.CORP/computers/move

Payload	{
	"computerIdentifier": "CN=Computer02,OU=Tax,DC=MYDOMAIN,DC=CORP",
	"targetContainer": "OU=FINANCE,DC=MYDOMAIN,DC=CORP"
	}

Response	{"errors":[]}

Restore

To original container

URI	dra/domains/MyDomain.corp/computers/restore
Payload	{ "computerIdentifier": "CN=Computer02,OU=NetIQRecycleBin,DC=MYDOMAIN,DC=CORP" }
Response	{"errors":[]}

To other container

URI	dra/domains/MyDomain.corp/computers/restore
Payload	{ "computerIdentifier": "CN=Computer02,OU=NetIQRecycleBin,DC=MYDOMAIN,DC=CORP", "restoreToContainer": "OU=TempCon,DC=MyDomain,DC=CORP" }
Response	{"errors":[]}

Update

Update using name

URI	dra/domains/MyDomain.corp/computers/put
Payload	{ "computerIdentifier": "COMPUTER05", "computer": { "description": "modified description 00", "accountThatCanAddComputerToDomain": “CN=Admin,OU=Dept,DC=MyDomain,DC=corp" } }
Response	{"errors":[]}

Update using distinguished name

URI	dra/domains/MyDomain.corp/computers/put
Payload	{ "computerIdentifier": "cn=COMPUTER05,ou=SomeDept,DC=MyDomain,DC=corp", "computer": { "description": "modified description 00", "isDisabled": "true" } }
Response	{"errors":[]}

Update with invalid name in payload

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

Payload	{
	"computerIdentifier": "BadName",
	"computer": {
	"description": "modified description 00",
	"accountThatCanAddComputerToDomain":
	"CN=SharonGroup,OU=SomeDept,DC=MyDomain,DC=corp"
	}
	}

Response	{
	"errors": [{
	"message": "Failed to find the computer named BadName",
	"suggestion": "",
	"stacktrace": ""
	}]
	}

Rename

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

Payload	{
	"computerIdentifier": "SomeObject,OU=SomeDept,DC=MyDomain,DC=corp",
	"computer": {
	"description": "modified description 00",
	"name": "Object_renamed"
	}
	}

Response	{"errors":[]}

To rename an object, supply the identifier for the object and put the new name (and any other properties
you want to change) in the payload.

Contacts

Create

ContactCreate - basic

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

Payload	{
	"contact": {
	"distinguishedName": "cn=Contact001,OU=SomeDept,DC=MyDomain,DC=corp"
	}
	}

Response	{"errors":[]}

ContactCreate – with email

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

Payload	{
	  "contact": {
	    "distinguishedName": "cn=Contact001,OU=SomeDept,DC=MyDomain,DC=corp",
	    "createEmail": true,
	    "legacyExchangeDn": "/o=First/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Recipients/cn=Contact001",
	    "emailAddress": "contact001email@mydomain.corp"
	  }
	}

Response	{"errors":[]}

Notes: To create a contact that has an email address, specify the createEmail=true flag. When the mailNickname is not provided, the alias will be generated by the DRA server to be compliant with configured alias naming policy. If the client supplies a mailNickname attribute that is out of compliance, the request will fail.

Get

ContactGetInfo

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

Payload	{
	  "contactIdentifier": "cn=Test Contact 02,OU=SomeDept,DC=MyDomain,DC=corp"
	}

Response	{
	  "contact": {
	    "mail": "contact02email@MyDomain.corp",
	    "additionalAttributes": {},
	    "name": "Test Contact 02",
	    "class": "Contact",
	    "displayName": "Test Contact 02"
	  },
	  "errors": []
	}

Get a list of contacts in a specific domain

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

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

Response	{
	  "contacts": [{
	      "additionalAttributes": {},
	      "description": "some description",
	      "class": "Contact",
	      "friendlyName": " CONTACT05",

	`"friendlyPath": "MYDOMAIN.CORP/Sharon/CONTACT05",`
	`"isManaged": true,`
	`"distinguishedName": "CN=CONTACT05,OU=SomeDept,`
	`DC = MYDOMAIN,`
	`DC = CORP ",`
	`},`
	Etc.
	`],`
	`"resumeString": "",`
	`"totalNumberOfObjects": 12,`
	`"errors": []`
	`}`

This example gets all contacts in the path Domain\SomeDept. The default container for the domain/contacts/get request is the domain root. To search for contacts across multiple domains, see Enumeration examples.

Delete

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

Payload	`{`
	`"contactIdentifier": "Contact001"`
	`}`

Response	`{"errors":[]}`

Update

Basic

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

Payload	`{`
	`"contactIdentifier": "Contact001",`
	`"contact": {`
	`"description": "modified description",`
	`"employeeId": "A7532Z"`
	`}`
	`}`

Response	`{"errors":[]}`

Enable Email

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

Payload	`{`
	`"contactIdentifier": "contact02cn",`
	`"contact": {`
	`"legacyExchangeDn": "/o=First/ou=Exchange Administrative Group(FYDIBOHF23SPDLT)/cn=Recipients/cn=contact02cn",`
	`"mailNickname": "contact02cnXXX",`
	`"emailAddress": "contact02cnemail@mydomain.corp"`

	} }
Response	{"errors":[]}

This example sets the mailNickname to a specific value. If this parameter is not supplied, the mailNickname will be generated by the DRA server. If the client supplies a mailNickname value that is out of compliance with configured policy, the request will fail.

Disable Email

URI	dra/domains/MyDomain.corp/contacts/disableemail/put

Payload	{
	    "contactIdentifier": "contact02cn"
	}

Response	{"errors":[]}

Rename

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

Payload	{
	    "contactIdentifier": "SomeObject,OU=SomeDept,DC=MyDomain,DC=corp",
	    "contact": {
	        "description": "modified description 00",
	        "name": "Object_renamed"
	    }
	}

Response	{"errors":[]}

To rename an object, supply the identifier for the object and put the new name (and any other properties
you want to change) in the payload.

Delegation Model

The endpoints in this group will allow a client to determine what powers the logged in user has been granted. These endpoints could be useful for developing a custom user interface.

These endpoints would not typically be used to automate object management tasks since the server will fail the request if the requesting user does not have privileges to perform the operation, so there is no need to validate the user's powers prior to sending a request.

Get Powers

For a Group

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

Payload	{
	    "groupIdentifier": "CN=someGroup,CN=Users,DC=MyDomain,DC=corp"
	}
	            Or
	{

	"groupIdentifier": "someGroup" }

Response	{
	"allPowers": [{
	"name": "Associate Virtual Attribute",
	"onePointPath": "OnePoint://pt=Associate Virtual Attribute,ct=Virtual Attributes,module=operations,PowerTemplate",
	"index": "00000000"
	}, {
	"name": "View Temporary Group Assignments",
	"onePointPath": "OnePoint://pt=View Temporary Group Assignments,ct=Temporary Group Assignment,ct=Groups,module=operations,PowerTemplate",
	"index": "0000004c"
	}],
	"errors": []
	}

For a user

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

Payload	{
	"userIdentifier": "CN=someUser,CN=Users,DC=MyDomain,DC=corp"
	}
	Or
	{
	"userIdentifier": "someuser"
	}

Response	{
	"allPowers": [{
	"name": "Associate Virtual Attribute",
	"onePointPath": "OnePoint://pt=Associate Virtual Attribute,ct=Virtual Attributes,module=operations,PowerTemplate",
	"index": "00000000"
	},
	
	{
	"name": "View Temporary Group Assignments",
	"onePointPath": "OnePoint://pt=View Temporary Group Assignments,ct=Temporary Group Assignment,ct=Groups,module=operations,PowerTemplate",
	"index": "0000004c"
	}
	],
	"errors": []
	}

OperationGetPowers

This endpoint will return the effective permission granted to the specified user or group, running the specified operation on each of the fields listed in the attributes object. When writing a custom UI, you could mark a field as read-only if the user does not have write privs on the field.

NOTE: the OperationGetPowers requires the attributes object in the payload. There is no default list of properties to query for these objects.

Requesting operation powers for a group

URI	dra/domains/MyDomain.corp/operations/powers/get

Payload	{
	  "objectIdentifier": "CN=SomeGroup,OU=Accounting,DC=myDomain,DC=corp",
	  "setOperationName": "GroupSetInfo",
	  "getOperationName": "GroupGetInfo",
	  "objectType": "Group",
	  "attributes": [
	    "mailNickname",
	    "description",
	    "managedBy"
	  ]
	}

Response	{
	  "Powers": {
	    "mailNickname": "RW",
	    "description": "RW",
	    "managedBy": "RW"
	  },
	  "errors": []
	}

Requesting operation powers for a user

URI	dra/domains/MyDomain.corp/operations/powers/get

Payload	{
	  "objectIdentifier": "CN=Administrator,CN=Users,DC=myDomain,DC=corp ",
	  "setOperationName": "UserSetInfo",
	  "getOperationName": "UserGetInfo",
	  "objectType": "User",
	  "attributes": ["mailNickname", "description", "employeeNumber"]
	}

Response	{
	  "Powers": {
	    "mailNickname": "RW",
	    "description": "RW",
	    "employeeNumber": "RW"
	  },
	  "errors": []
	}

Get Assignments

For User or Group

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

Payload	{
	"objectIdentifier": " CN=Administrator,CN=Users,DC=MyDomain,DC=corp"
	}
	Or
	{
	" objectIdentifier ": " CN=SomeGroup,OU=Accounting,DC=myDomain,DC=corp"
	}

Response	{
	roles":[
	{
	"sysFlag": true,
	"isAssigned": true,
	"hidden": false,
	"isDynamic": false,
	"activeView": "DRA Security Objects",
	"class": "Role",
	"nameValue": "Audit Users and Groups",
	"objectClass": "Role",
	"path": "OnePoint://role=Audit Users and Groups,module=Security",
	}],
	activeViews":[
	{
	"sysFlag": true,
	"hidden": false,
	"isDynamic": false,
	"class": "ActiveView",
	"nameValue": "DRA Security Objects",
	"objectClass": "ActiveView",
	"path": "OnePoint://av=DRA Security Objects,module=Security",
	}],
	powers":[
	{
	"powerTemplate": "OnePoint://pt=Add Group to Group,ct=Manage Group Membership,ct=Groups,module=operations,PowerTemplate",
	"isDynamic": false,
	"activeView": "DRA Security Objects",
	"class": "Power",
	"friendlyName": "Add Group to Group",
	"nameValue": "Add Group to Group",
	"objectClass": "Power",
	"path": "Add Group to Group",
	}],
	"errors":[]
	}

Domains

Get with default property list

URI	dra/domains/get

Payload	{
	"domainIdentifier": "DC=MyDomain,DC=corp"
	}
	Or
	{
	"domainIdentifier": "MyDomain.corp"
	}

Response	{
	"Domain": {
	"dnsFlatName": "MYDOMAIN.CORP(DRDOM610)",
	"lastRefreshEnded": "0001-01-01T00:00:00",
	"lastSuccessfulRefreshEnded": "2014-03-20T17:05:18",
	"isReqManaged": true,
	"additionalAttributes": {},
	"class": "Domain",
	"friendlyPath": "MYDOMAIN.CORP",
	"distinguishedName": "DC=MYDOMAIN,DC=CORP"
	},
	"errors": []
	}

DynamicDistributionGroups

Create

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

Payload	{
	"dynamicDistributionGroup": {
	"distinguishedName":
	"CN=TestGroup02,OU=SomeDept01,OU=SomeDept,DC=MYDOMAIN,DC=CORP",
	"description": "rest test group 02",
	"exchangeToolVersion": "2013"
	}
	}

Response	{"errors":[]}

Notes: When the mailNickname is not provided, the alias will be set to the name of the dynamic distribution group.

Get

Basic

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

Payload	{
	   "ddgIdentifier": "myDDG"
	}

Response	{
	   "dynamicDistributionGroup": {
	      "includedRecipients": "AllRecipients",
	      "recipientContainer": "OnePoint://OU=SomeDept,DC=MYDOMAIN,DC=CORP",
	      "class": "DynamicDistributionGroup",
	      "distinguishedName": "CN=myDDG,OU=SomeDept,DC=MYDOMAIN,DC=CORP",
	      "name": "myDDG",
	      "additionalAttributes": {}
	   },
	   "errors": []
	}

Requesting specific attributes

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

Payload	{
	   "ddgIdentifier": "CN=myDDG,OU=SomeDept,DC=MYDOMAIN,DC=CORP",
	   "attributes": [
	      "samAccountName",
	      "distinguishedName",
	      "description",
	      "recipientLDAPFilter",
	      "includedRecipients"
	   ]
	}

Response	{
	   "dynamicDistributionGroup": {
	      "includedRecipients": "AllRecipients",
	      "description": "some DDG description",
	      "class": "DynamicDistributionGroup",
	      "recipientLDAPFilter": " (&(mailNickname=*) Etc.))) ",
	      "distinguishedName": "CN=myDDG,OU=SomeDept,DC=MYDOMAIN,DC=CORP",
	      "samAccountName": "myDDG",
	      "additionalAttributes": {}
	   },
	   "errors": []
	}

Get a list of dynamic distribution groups in a specific domain

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

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

Response	{
	   "dynamicDistributionGroups": [{
	         "includedRecipients": "AllRecipients",
	         "recipientContainer": "OnePoint://OU=SomeDept,DC=MYDOMAIN,DC=CORP",
	         "class": "DynamicDistributionGroup",
	         "distinguishedName": "CN=myDDG01,OU=SomeDept,DC=MYDOMAIN,DC=CORP",
	         "name": "myDDG",
	         "additionalAttributes": {}
	      },
	      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. To search for groups across multiple domains, see Enumeration examples.

Delete

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

Payload	{
	   "ddgIdentifier": "CN=TestDdg ,OU=SomeDept,DC=MYDOMAIN,DC=CORP"
	}

Response	{"errors":[]}

Update

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

Payload	{
	   "ddgIdentifier": "TestGroup02",
	   " dynamicDistributionGroup": {
	      "description": "updated DESC old4 yyyy",
	      "includedRecipients": "AllRecipients",
	      "moderatorExemptions": [
	         "CN=User001,OU=Engineering,DC=MYDOMAIN,DC=CORP",
	         "CN=User007,OU=HumanResources,DC=MYDOMAIN,DC=CORP"
	      ]
	   }

	}
Response	{"errors":[]}

Copy

URI	dra/domains/MyDomain.corp/dynamicdistributiongroups/copies/post
Payload	{ "ddgIdentifier": "CN=SourceDDG,OU=SomeDept01, OU=SomeDept,DC=MYDOMAIN,DC=CO " dynamicDistributionGroup": { "distinguishedName": "CN=DestinationDDG,OU=SomeDept01,DC=MYDOMAIN,DC=CORP } }
Response	{ "dynamicDistributionGroup": { "class": "DynamicDistributionGroup", "distinguishedName": "cn=DestinationDDG,OU=SomeDept01,DC=MYDOMAIN,DC=CORP" }, "errors": [] }

Move

URI	dra/domains/MyDomain.corp/dynamicdistributiongroups/move
Payload	{ "ddgIdentifier": "cn=TestDDG02,OU=QE,DC=Mydomain,DC=corp", "targetContainer": "OU=Release,DC=Mydomain,DC=corp" }
Response	{"errors":[]}

Restore

To original container

URI	dra/domains/MyDomain.corp/dynamicdistributiongroups/restore
Payload	{ "ddgIdentifier": "CN=TestGroup02,OU=NetIQRecycleBin,DC=MYDOMAIN,DC=CORP" }
Response	{"errors":[]}

To other container

URI	dra/domains/MyDomain.corp/dynamicdistributiongroups/restore
Payload	{ "ddgIdentifier": "CN=TestGroup02,OU=NetIQRecycleBin,DC=MYDOMAIN,DC=CORP", "restoreToContainer": "OU=TempCon,DC=MyDomain,DC=CORP" }
Response	{"errors":[]}

Preview

URI	dra/domains/MyDomain.corp/dynamicdistributiongroups/preview

Payload	{
	  " dynamicDistributionGroup": {
	    "distinguishedName": "OU=SomeDept01, DC=MYDOMAIN,DC=CORP"
	  }
	}

Response	{
	  "members": [{
	      "samAccountName": "USER1",
	      "class": "User",
	      "displayName": "USER1",
	      "distinguishedName": "CN=USER1,OU=SomeDept01,DC=MYDOMAIN,DC=CORP",
	      "friendlyName": null,
	      "friendlyPath": null,
	      "isDisabled": null
	    }
	    Etc.
	  ],
	  "resumeString": "",
	  "totalNumberOfObjects": 1,
	  "numberOfRowsReturned": 1,
	  "isSearchFinished": true,
	  "errors": []
	}

Groups

Create

Basic

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

Payload	{
	  "group": {
	    "distinguishedName": "CN=TestGroup02 ,OU=SomeDept,DC=MYDOMAIN,DC=CORP",
	    "description": "rest test group 02",
	    "groupType": "security",
	    "groupScope": "domain local"
	  }
	}

Response	{"errors":[]}

With email

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

Payload	{
	  "group": {

	"distinguishedName": "CN=TestGroup02 ,OU=SomeDept,DC=MYDOMAIN,DC=CORP", "description": "rest test group 02", "groupType": "security", "groupScope": "domain local", "createEmail": true } }
Response	{"errors":[]}

Notes: To create a group that has an email address, specify the createEmail=true flag. When the mailNickname is not provided, the alias will be generated by the DRA server to be compliant with configured alias naming policy. If the client supplies a mailNickname attribute that is out of compliance, the request will fail.

AD dynamic group

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

Payload	{
	"group": {
	"distinguishedName": "CN=TestGroup02,OU=SomeDept,DC=MYDOMAIN,DC=CORP",
	"description": "rest test group 02",
	"groupType": "security",
	"groupScope": "domain local",
	"whiteList": ["cn=wh1,OU=myOU,DC=mydom,DC=corp"],
	"blackList": [
	"cn=Nov3,cn=Users,DC=mydom,DC=corp",
	"cn=Nov4,cn=Users,DC=mydom,DC=corp"
	],
	"memberFilter": [{
	"includeChildContainers": true,
	"ldapFilter": "(&(objectClass=user)(objectCategory=person))",
	"rootFilter": "DC=DRDOM800,DC=LAB"
	}]
	}
	}

Response	{"errors":[]}

Notes: To create a dynamic group (or to change an existing group to be dynamic), specify the memberFilter object. If you have accounts that you want permanently added to the group, add them to the whiteList. If you have members you want excluded from the group, even if they match the filter, add them to the blacklist attribute. The memberFilter, whiteList, and blackList attributes are all arrays.

Get

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. To search for groups across multiple domains, see Enumeration examples.

Move

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

Payload	{
	"groupIdentifier": "cn=TestGroup02,OU=QE,DC=Mydomain,DC=corp",
	"targetContainer": "OU=Release,DC=Mydomain,DC=corp"
	}

Response	{"errors":[]}

Delete

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

Payload	{
	"groupIdentifier": "TestGroup02"
	}

Response	{"errors":[]}

Restore

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

Payload	{
	"groupIdentifier": "cn=TestGroup02,OU=NetIQRecycleBin,DC=Mydomain,DC=corp"
	}

Response	{"errors":[]}

Update

Basic

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

	"groupIdentifier": "TestGroup02",
	"group": {
	"description": "updated description",
	"groupScope": "domain local",
	"groupType": "distribution"
	}
	}

Response	{"errors":[]}

Enable Email

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

Payload	{
	"groupIdentifier": "TestGroup02U",
	"group": {
	"legacyExchangeDn": "/o=First/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Recipients/cn=TestGroup02U",
	"mailNickname": "TestGroup02U"
	}
	}

Response	{"errors":[]}

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

Disable Email

URI	dra/domains/MyDomain.corp/groups/disableemail/put

Payload	{
	"groupIdentifier": "TestGroup02U"
	}

Response	{"errors":[]}

Rename

URI – display name	dra/domains/MyDomain.corp/groups/put

Payload	{
	"groupIdentifier": "SomeObject,OU=SomeDept,DC=MyDomain,DC=corp",
	"group": {
	"description": "modified description 00",
	"name": "Object_renamed"
	}
	}

Response	{"errors":[]}

To rename an object, supply the identifier for the object and put the new name (and any other properties
you want to change) in the payload.

Remove dynamic filter (convert to static group)

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

Payload	`{`
	`   "groupIdentifier": "TestGroup02",`
	`   "group": {`
	`      "isDynamic": false`
	`   }`
	`}`

Response	`{"errors":[]}`

Notes: To convert a dynamic group to a static group, send the isDynamic flag with a value of false. To perform the conversion, DRA will run a job that removes the dynamic members. The job runs asynchronously, so there may be a small delay between sending the request and the group showing the correct member list. No further actions can be made on the group while the job is running.

Copy

Copy group and all group members

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

Payload	`{`
	`   "groupIdentifier": "CN=SourceGroup, OU=SomeDept,DC=MYDOMAIN,DC=CORP",`
	`   "group": {`
	`      "name": "DestinationGroup"`
	`   }`
	`}`

Response	`{`
	`   "group": {`
	`      "class": "Group",`
	`      "distinguishedName":`
	`"cn=DestinationGroup,,OU=SomeDept,DC=MYDOMAIN,DC=CORP"`
	`   },`
	`   "errors": []`
	`}`

Do not copy all group members

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

Payload	`{`
	`   "groupIdentifier": "CN=SourceGroup,OU=SomeDept,DC=MYDOMAIN,DC=CORP",`
	`   "group": {`
	`      "name": "DestinationGroup"`
	`   },`
	`   "noCopyMembers": true,`
	`}`

Response	{
	"group": {
	"class": "Group",
	"distinguishedName":
	"cn=DestinationGroup,OU=SomeDept,DC=MYDOMAIN,DC=CORP"
	},
	"errors": []
	}

Member Actions

Add Members to a Group

URI	dra/domains/MyDomain.corp/groups/members/post

Payload	{
	"groupIdentifier": "TestGroup05",
	"groups": [
	{
	"distinguishedName": "CN=Group01,OU=Dept01,OU=Dept,DC=MYDOMAIN,DC=CORP"
	},
	{
	"distinguishedName": "CN=Group06,OU=Dept01,OU=Dept,DC=MYDOMAIN,DC=CORP"
	}
	],
	"computers": [
	{
	"distinguishedName": "CN=COMPUTER05,OU=Dept,DC=MYDOMAIN,DC=CORP"
	}
	]
	}

Response	{"errors":[]}

Valid object arrays are: *computers*, *contacts*, *groups*, and *users*. In the example above, the endpoint processes 2 group objects and 1 computer object. All 3 objects are added as members to TestGroup05.

Add Members to Multiple Groups

Instead of sending a single group identifier in the payload for groupIdentifier, the payload may optionally contain multiple groups in a groupIdentifiers array to add members to several groups at once.

URI	dra/domains/MyDomain.corp/groups/members/post
Payload	{ "groupIdentifiers": ["Group10", "Group11", "Group12"], "groups": [{ "distinguishedName": "CN=Group01,OU=Dept01,OU=Dept,DC=MYDOMAIN,DC=CORP"

	},
	{
	"distinguishedName": "CN=Group06,OU=Dept01,OU=Dept,DC=MYDOMAIN,DC=CORP"
	}
	],
	"computers": [
	{
	"distinguishedName": "CN=COMPUTER05,OU=Dept,DC=MYDOMAIN,DC=CORP"
	}
	]
	}

Response	{"errors":[]}

Valid object arrays are: *computers*, *contacts*, *groups*, and *users*. In the example above, the endpoint processes 2 group objects and 1 computer object. All 3 objects are added as members to Group10, Group11, and Group12.

Remove Members from a Group

The same payload format used to add members to a group is used to remove members from a group.

URI	dra/domains/MyDomain.corp/groups/members/delete

Payload	{
	"groupIdentifier": "TestGroup05",
	"groups": [
	{
	"distinguishedName":
	"CN=Group01,OU=Dept01,OU=Dept,DC=MYDOMAIN,DC=CORP"
	},
	{
	"distinguishedName":
	"CN=Group06,OU=Dept01,OU=Dept,DC=MYDOMAIN,DC=CORP"
	}
	],
	"computers": [
	{
	"distinguishedName": "CN=COMPUTER05,OU=Dept,DC=MYDOMAIN,DC=CORP"
	}
	]
	}

Response	{"errors":[]}

Remove Members from Multiple Groups

Similar to adding members to a group, the payload may optionally contain multiple groups in a groupIdentifiers array to remove members from several groups at once.

URI	dra/domains/MyDomain.corp/groups/members/delete
Payload	{ "groupIdentifiers": ["Group10", "Group11",

	"Group12"
	],
	"groups": [
	{
	"distinguishedName":
	"CN=Group01,OU=Dept01,OU=Dept,DC=MYDOMAIN,DC=CORP"
	},
	{
	"distinguishedName":
	"CN=Group06,OU=Dept01,OU=Dept,DC=MYDOMAIN,DC=CORP"
	}
	],
	"computers": [
	{
	"distinguishedName": "CN=COMPUTER05,OU=Dept,DC=MYDOMAIN,DC=CORP"
	}
	]
	}

Response	{"errors":[]}

Get Members

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

Payload	{
	"groupIdentifier": "TestGroup01"
	}
	Or
	{
	"groupIdentifier": "CN=TestGroup01,OU=SomeDept,DC=MYDOMAIN,DC=CORP"
	}
	Or
	{
	"groupIdentifier": "MYDOMAIN.CORP/SomeDept01/TestGroup01"
	}

Response	{
	"members": [{
	"additionalAttributes": {},
	"name": "User002",
	"class": "User",
	"friendlyParentPath": "MyDomain.corp/Tech",
	"distinguishedName": "CN=User002,OU=Tech,DC=mydomain,DC=corp",
	"samAccountName": "User002",
	"displayName": "User002"
	},
	Etc.
	],
	"errors": []
	}

Get Member-Of

URI	dra/domains/MyDomain.corp/groups/membership/get
Payload	{ "objectIdentifier": "CN=user12,OU=Tech,DC=MyDomain,DC=corp", "objectType": "user", "enumerationOptions": { "objectsPerResponse": 1 } }
Response	{ "members": [{ "groupType": "security", "groupScope": "global", "groupTypeScope": "security - global", "additionalAttributes": {}, "name": "Domain Users", "description": "All domain users", "class": "Group", "friendlyParentPath": "MYDOMAIN.CORP/Users", "distinguishedName": "CN=Domain Users,CN=Users,DC=MyDomain,DC=corp", "samAccountName": "Domain Users" }], "resumeString": "CN=Domain Users,CN=Users,DC=DRDOM610,DC=LAB", "totalNumberOfObjects": 2, "errors": [] }

This payload requests the groups that have User12 as a member. The payload specifies that only one member should be returned in the response. The response shows that the user is a member of 2 total groups, and provides the resumeString value to retrieve more members.

Get Membeship Security

URI	dra/domains/MyDomain.corp/groups/permissions/get
Payload	{ "groupIdentifier": "CN=group12,OU=Tech,DC=MyDomain,DC=corp", }
Response	{ "trustees": [{ "trusteeName": " MyDomain \user11", " classuser": "", "accessmode": 1, " trusteeSid": "S-1-5-21-351726660-2265320152-3547800743-1672", " permissions": 48, " objTypeGuid": "{BF9679C0-0DE6-11D0-A285-00AA003049E2}", "path": "OnePoint://CN= user11,DC= MyDomain,DC= corp " }], "totalNumberOfObjects": 1, "errors": [] }

accessmode: 1 allow; 3 deny
permissions: 16 Read; 32 Write; 48 Read and Write

Set Membership Security

The payload may optionally contain multiple users and groups in an array to add trustees to the group at once.

URI	dra/domains/MyDomain.corp/groups/ permissions /put
Payload	{ "groupIdentifiers": "Group10", "users": [{ "distinguishedName": "CN=User01,OU=Dept01,OU=Dept,DC=MYDOMAIN,DC=CORP", "accessMode": 1, "permissions": 16, "action": 0 }, { "distinguishedName": "CN= User02,OU=Dept01,OU=Dept,DC=MYDOMAIN,DC=CORP", "accessMode": 1, "permissions": 32, "action": 0 }], "groups": [{ "distinguishedName": "CN=Group11,OU=Dept01,OU=Dept,DC=MYDOMAIN,DC=CORP", "accessMode": 1, "permissions": 48, "action": 0 }, { "distinguishedName": "CN=Group06,OU=Dept01,OU=Dept,DC=MYDOMAIN,DC=CORP", "accessMode": 1, "permissions": 16, "action": 1 }] }
Response	{"errors":[]}

accessmode: 1 allow; 3 deny
permissions: 16 Read; 32 Write; 48 Read and Write
action: 0 add; 1 remove

Organizational Unit

Create

URI	dra/domains/MYDOMAIN.CORP/ous/post

Payload	{
	  "ou": {
	    "description": "my new OU description",
	    "distinguishedName": "OU=SomeDept02A,OU=SomeDept,DC=MYDOMAIN,DC=CORP"
	  }
	}

Response	{"errors":[]}

Get

Basic

URI	dra/domains/MYDOMAIN.CORP/ous/get
Payload	{ "ouIdentifier" : "BlueBird02" }

Response	{
	  "ou": {
	    "additionalAttributes": {},
	    "name": "BlueBird02",
	    "description": "test ou grandchild",
	    "class": "OrganizationalUnit",
	    "distinguishedName": "OU=SomeDept01,OU=SomeDept,DC=MYDOMAIN,DC=CORP"
	  },
	  "errors": []
	}

Get a list of OUs in a specific domain

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

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

Response	{
	  "ous": [{
	      "additionalAttributes": {},
	      "description": "some description",
	      "class": "OU",
	      "friendlyName": " OU05",
	      "friendlyPath": "MYDOMAIN.CORP/Sharon/OU05",
	      "isManaged": true,
	      "distinguishedName": "CN=OU05,OU=SomeDept,DC=MYDOMAIN,DC=CORP",
	    },

	Etc.], "resumeString": "", "totalNumberOfObjects": 12, "errors": [] }

This example gets all OUs, including the children, in the path Domain\SomeDept. The default container for the domain/OUs/get request is the domain root. To search for OUs across multiple domains, see Enumeration examples.

Delete

URI	dra/domains/MYDOMAIN.CORP/ous/delete

Payload	{
	"ouIdentifier" : "SomeDept"
	}

Response	{"errors":[]}

Update

URI	dra/domains/MYDOMAIN.CORP/ous /put

Payload	{
	"ouIdentifier": "BlueBird02",
	"ou": {
	"description": "updated OU description"
	}
	}

Response	{"errors":[]}

Move an object from one OU to another

URI	dra/domains/MYDOMAIN.CORP/ous/move

Payload	{
	"objectIdentifier": "BlueBird02",
	"targetContainer": "OU=SomeDept,DC=MYDOMAIN,DC=CORP"
	}

Response	{"errors":[]}

Containers

Get

Basic

URI	dra/domains/MYDOMAIN.CORP/containers/get
Payload	{ "containerIdentifier" : "Users" }

Response	{
	"container": {
	"additionalAttributes": {},
	"name": "Users",

	"description": "Default container for upgraded user accounts", "class": "OrganizationalUnit", "objectClass": "Container", "distinguishedName": "CN=Users,DC=MYDOMAIN,DC=CORP" }, "errors": [] }

Get a list of containers in a specific domain

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

Payload	{
	"enumerationOptions": {
	"includeChildContainers": true,
	"containerDistinguishedName": "DC=MYDOMAIN,DC=CORP"
	}
	}

Response	{
	"containers": [{
	"additionalAttributes": {},
	"name": "Users",
	"description": "Default container for upgraded user accounts",
	"class": "OrganizationalUnit",
	"objectClass": "Container",
	"distinguishedName": "CN=Users,DC=MYDOMAIN,DC=CORP"
	},
	Etc.
	],
	"resumeString": "",
	"totalNumberOfObjects": 12,
	"errors": []
	}

This example gets all containers in domain MYDOMAIN.CORP. For more complicated searches, see Enumeration examples.

Update

URI	dra/domains/MYDOMAIN.CORP/containers /put

Payload	{
	conatinerIdentifier": "Users",
	"container": {
	"description": "Default user accounts for MYDOMAIN.CORP"
	}
	}

Response	{"errors":[]}

Builtin Containers

Get

Basic

URI	dra/domains/MYDOMAIN.CORP/builtinContainers/get
Payload	{ "containerIdentifier" : "Users" }
Response	{ "builtinContainer": { "additionalAttributes": {}, "name": "Builtin", "description": "", "class": "OrganizationalUnit", "objectClass": "Builtin", "distinguishedName": "CN=Builtin,DC=MYDOMAIN,DC=CORP" }, "errors": [] }

Get a list of builtin containers in a specific domain

URI	dra/domains/MyDomain.corp/builtinContainers/get
Payload	{ "enumerationOptions": { "includeChildContainers": true, "containerDistinguishedName": "DC=MYDOMAIN,DC=CORP" } }
Response	{ "builtinContainers": [{ "additionalAttributes": {}, "name": "Builtin", "description": "", "class": "OrganizationalUnit", "objectClass": "Builtin", "distinguishedName": "CN=Builtin,DC=MYDOMAIN,DC=CORP" }], "resumeString": "", "totalNumberOfObjects": 1, "errors": [] }

This example gets all builtin containers in domain MYDOMAIN.CORP. For more complicated searches, see Enumeration examples.

Update

URI	dra/domains/MYDOMAIN.CORP/builtinContainers /put
Payload	{

	conatinerIdentifier": "Builtin",
	"container": {
	"description": "The builtin contiainer"
	}
	}

Response	{"errors":[]}

Resource Mailbox

Enumeration

Get a list of Resource Mailboxes

URI	dra/domains/MYDOMAIN.CORP/resourcemailboxes /get

Payload	{
	"enumerationOptions": {
	"includeChildContainers": true
	}
	}

Response	{
	"equipmentMailboxes": [{
	"includeResource": true,
	"mailboxType": 2,
	"class": "User",
	"distinguishedName": "CN=DefaultEquipmentMB,
	OU = SomeDept,
	DC = MYDOMAIN,
	DC = CORP ",
	},
	Etc.
	],
	"roomMailboxes": [{
	"includeResource": true,
	"mailboxType": 3,
	"class": "User",
	"distinguishedName": "CN=DefaultRoomMB,
	OU = SomeDept,
	DC = MYDOMAIN,
	DC = CORP ",
	"additionalAttributes": {}
	},
	Etc.
	],
	"resumeString": "",
	"totalNumberOfObjects": 5,
	"numberOfRowsReturned": 5,
	"isSearchFinished": true,
	"errors": []

	}

Get a list of Equipment Resource Mailboxes

URI	dra/domains/MYDOMAIN.CORP/equipmentresourcemailboxes /get

Payload	{
	  "enumerationOptions": {
	    "includeChildContainers": true
	  }
	}

Response	{
	  "equipmentMailboxes": [{
	      "includeResource": true,
	      "mailboxType": 2,
	      "class": "User",
	      "distinguishedName": "CN=DefaultEquipmentMB,
	      OU = SomeDept,
	      DC = MYDOMAIN,
	      DC = CORP ",
	    }
	    Etc.
	  ],
	  "roomMailboxes": [],
	  "resumeString": "",
	  "totalNumberOfObjects": 5,
	  "numberOfRowsReturned": 5,
	  "isSearchFinished": true,
	  "errors": []
	}

Get a list of Room Resource Mailboxes

URI	dra/domains/MYDOMAIN.CORP/roomresourcemailboxes /get

Payload	{
	  "enumerationOptions": {
	    "includeChildContainers": true
	  }
	}

Response	{
	  "equipmentMailboxes": [],
	  "roomMailboxes": [{
	      "includeResource": true,
	      "mailboxType": 3,
	      "class": "User",
	      "distinguishedName": "CN=DefaultRoomMB,
	      OU = SomeDept,
	      DC = MYDOMAIN,
	      DC = CORP ",

	"additionalAttributes": {} } Etc.], "resumeString": "", "totalNumberOfObjects": 5, "numberOfRowsReturned": 5, "isSearchFinished": true, "errors": [] }

Create

URI	dra/domains/MYDOMAIN.CORP/roomresourcemailboxes /post or dra/domains/MYDOMAIN.CORP/equipmentresourcemailboxes /post

Payload	{
	"resourceMailbox": {
	"distinguishedName": "CN=MyRMB, OU=EMEA,DC=MyDomain,DC=corp ",
	"addAdditionalResponse": true,
	"additionalResponseText": "some additional text to add",
	"addOrganizerToSubject": true,
	"allowConflicts": true,
	"allowRecurringMeetings": true,
	"allowScheduleOnlyDuringWorkingHours": true,
	"conflictCountMaximum": 22,
	"conflictPercentageAllowed": 22,
	"declineIfMeetingDateIsPassed": true,
	"deleteAttachments": true,
	"deleteComments": true,
	"deleteNonCalendarItems": true,
	"deleteSubject": true,
	"extensionAttribute1": "custom value 22",
	"forwardAndDeliver": true,
	"forwardToRecipient": "CN=User001ME, OU=EMEA,DC=MyDomain,DC=corp ",
	"markPendingAsTentative": true,
	"maximumBookingLeadTime": 22,
	"maximumDurationInMinutes": 22,
	"recipientPolicyUpdatesEmailAddress": false,
	"removePrivateFlagOnAcceptedMeeting": true,
	"resourceAttributes": ["someresource"],
	"resourceCapacity": 22,
	"sendOrganizerInfoWithDecline": true,
	"incomingEmailSize": 2222,
	"outgoingEmailSize": 2222,
	"acceptMessagesFrom": [
	"CN=DDG,CN=MyDDG, OU=EMEA,DC=MyDomain,DC=corp "
	],

	"forwardEmailRecipient": "CN=User001ME, OU=EMEA,DC=MyDomain,DC=corp ", "quotaIssueWarning": 2522, "quotaUseMBStoreDefaults": false, "maximumRecipients": 822, "requireAuthenticationToSendTo": true, "proxyAddresses": ["ZZZ:RoomEquiq22zzz@mydomain.corp", "SMTP:RoomEquiq22@mydomain.corp"], "rejectMessagesFrom": ["CN=UnivSecGrp01,OU=USA,DC=MyDomain,DC=corp", "CN=UnivDistGrp,OU=USA,DC=MyDomain,DC=corp"] } }

Response	{
	"mailboxTypeName": "Room",
	"resourceMailbox": {
	"distinguishedName": "CN=MyRMB,OU=EMEA,DC=MyDomain,DC=corp",
	"class": "ResourceMailbox",
	},
	"draServerAndPort": "HOUDVDR164:11192",
	"errors": []
	}
	
	Or
	{
	"mailboxTypeName": "Equipment",
	"resourceMailbox": {
	"distinguishedName": "CN=MyRMB,OU=EMEA,DC=MyDomain,DC=corp",
	"class": "ResourceMailbox",
	},
	"draServerAndPort": "HOUDVDR164:11192",
	"errors": []
	}

Get

URI	dra/domains/MYDOMAIN.CORP/resourcemailboxes /get

Payload	{
	"resourceMailboxIdentifier": "CN=SomeRoom,OU=EMEA,DC=MyDomain,DC=CORP"
	}

Response	{
	"resourceMailbox": {
	"domainDistinguishedName": "OnePoint://DC=MyDomain,DC=corp",
	"mailboxTypeName": "Room",
	"mailboxType": 3,
	"homeMdb": "CN=exchange,CN=Databases, etc… ",

	"proxyAddresses": ["SMTP:DefaultRoomMB@drdom150.lab"], "class": "ResourceMailbox", "distinguishedName": "CN=SomeRoom,OU=EMEA,DC=MyDomain,DC=corp", }, "draServerAndPort": "HOUDVDR164:11192", "errors": [] }

Update

URI	dra/domains/MYDOMAIN.CORP/resourcemailboxes /put

Payload	{
	"resourceMailboxIdentifier": "CN=MyRMB",
	"resourceMailbox": {
	"addAdditionalResponse": true,
	"additionalResponseText": "some additional text to add",
	"addOrganizerToSubject": true,
	"allowConflicts": true,
	"allowRecurringMeetings": true,
	"allowScheduleOnlyDuringWorkingHours": true,
	"conflictCountMaximum": 22,
	"conflictPercentageAllowed": 22,
	"declineIfMeetingDateIsPassed": true,
	"deleteAttachments": true,
	"deleteComments": true,
	"deleteNonCalendarItems": true,
	"deleteSubject": true,
	"extensionAttribute1": "custom value 22",
	"forwardAndDeliver": true,
	"forwardToRecipient": "CN=User001ME, OU=EMEA,DC=MyDomain,DC=corp ",
	"markPendingAsTentative": true,
	"maximumBookingLeadTime": 22,
	"maximumDurationInMinutes": 22,
	"recipientPolicyUpdatesEmailAddress": false,
	"removePrivateFlagOnAcceptedMeeting": true,
	"resourceAttributes": ["someresource"],
	"resourceCapacity": 22,
	"sendOrganizerInfoWithDecline": true,
	"incomingEmailSize": 2222,
	"outgoingEmailSize": 2222,
	"acceptMessagesFrom": [
	"CN=DDG,CN=MyDDG, OU=EMEA,DC=MyDomain,DC=corp "
	],
	"forwardEmailRecipient": "CN=User001ME, OU=EMEA,DC=MyDomain,DC=corp ",
	"quotaIssueWarning": 2522,
	"quotaUseMBStoreDefaults": false,

	"maximumRecipients": 822, "requireAuthenticationToSendTo": true, "proxyAddresses": ["ZZZ:RoomEquiq22zzz@mydomain.corp", "SMTP:RoomEquiq22@mydomain.corp"], "rejectMessagesFrom": ["CN=UnivSecGrp01,OU=USA,DC=MyDomain,DC=corp", "CN=UnivDistGrp,OU=USA,DC=MyDomain,DC=corp"] } }
Response	{"errors":[]}

This example uses the display name to identify the resource mailbox to update. When the display name is used, the REST endpoint will make an additional call to the DRA server to obtain the distinguished name. If more than one resource mailbox is found in that domain with the same display name, the request will fail.

Delete

URI	dra/domains/MYDOMAIN.CORP/resourcemailboxes /delete
Payload	{ "resourceMailboxIdentifier": "CN=SomeRoom,OU=EMEA,DC=MyDomain,DC=CORP" }
Response	{ "errors": [ ]}

Restore

To original container

URI	dra/domains/MyDomain.corp/resourcemailboxes/restore
Payload	{ "resourceMailboxIdentifier": "CN=ConfRoom02,OU=NetIQRecycleBin,DC=MYDOMAIN,DC=CORP" }
Response	{"errors":[]}

To other container

URI	dra/domains/MyDomain.corp/resourcemailboxes/restore
Payload	{ "resourceMailboxIdentifier": "CN=ConfRoom02,OU=NetIQRecycleBin,DC=MYDOMAIN,DC=CORP", "restoreToContainer": "OU=TempCon,DC=MyDomain,DC=CORP" }
Response	{"errors":[]}

Move

URI	dra/domains/MYDOMAIN.CORP/resourcemailboxes/move
Payload	{ "resourceMailboxIdentifier": "CN=RoomRMB02,OU=MainConf,DC=MYDOMAIN,DC=CORP", "targetContainer": "OU=SpecialConf,DC=MYDOMAIN,DC=CORP" }
Response	{"errors":[]}

Copy

Change only the name in the target object

URI	dra/domains/MyDomain.corp/resourcemailboxes/copies/post
Payload	{ "resourceMailboxIdentifier": "CN=SourceUser,OU=SomeDept,DC=MYDOMAIN,DC=CORP", "user": { "name": "DestinationUser" } }
Response	{ "user": { "mailboxType": 3, "class": "User", "distinguishedName": "cn=DestinationUser,OU=SomeDept,DC=MYDOMAIN,DC=CORP" }, "errors": [] }

Change the name and some other properties in the target object

URI	dra/domains/MyDomain.corp/resourcemailboxes/copies/post
Payload	{ "userIdentifier": "CN=SourceUser,OU=SomeDept,DC=MYDOMAIN,DC=CORP", "user": { "name": "DestinationUser", "description": "new desc", "addToGroups": "CN=somegroup,OU=Admin,DC=MYDOMAIN,DC=CORP" } }

Response	{ "user": { "class": "User", "distinguishedName": "cn=DestinationUser,OU=SomeDept01,DC=MYDOMAIN,DC=CORP" }, "errors": [] }

User

Create

Basic

URI	dra/domains/MyDomain.corp/users/post
Payload	{ "user": { "distinguishedName": "cn=User09,OU=SomeDept,DC=MyDomain,DC=corp" } }
Response	{ "user": { "userPassword": "HJv5MYkz$", "additionalAttributes": {}, "class": "User" }, "errors": [] }

Notes: A userPassword generated by the DRA server will be returned to the client. If the client provides the password, then no password is returned. When the DRA server generates the password, the user is set to require the password to be changed at the next logon.

With Email

URI	dra/domains/MyDomain.corp/users/post
Payload	{ "user": { "distinguishedName": "cn=User09,OU=SomeDept,DC=MyDomain,DC=corp", "emailAddress": "User09@mydomain.com", "createEmail": true } }
Response	{ "user": { "userPassword": "HJv5MYkz$", "additionalAttributes": {},

	"class": "User" }, "errors": [] }

Notes: To create a user that has an email address, specify the createEmail=true flag. When the mailNickname is not provided, the alias will be generated by the DRA server to be compliant with configured alias naming policy. If the client supplies a mailNickname attribute that is out of compliance, the request will fail.

With Mailbox

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

Payload	{
	"user": {
	"distinguishedName": "cn=User09,OU=SomeDept,DC=MyDomain,DC=corp",
	"mailboxStore": "LDAP://CN=Mailbox Database 0565232367,CN=Databases,CN=Exchange Administrative Group (FYDIBOHF23SPDLT),CN=Administrative Groups,CN=First,CN=Microsoft Exchange,CN=Services,CN=Configuration,DC=MYDOMAIN,DC=CORP",
	"createMailbox": true
	}
	}

Response	{
	"user": {
	"userPassword": "HJv5MYkz$",
	"additionalAttributes": {},
	"class": "User"
	},
	"errors": []
	}

Notes: To create a user that has an email address, specify the createMailbox=true flag.

With Mailbox and Default Mailbox Store

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

Payload	{
	"user": {
	"distinguishedName": "cn=User09,OU=SomeDept,DC=MyDomain,DC=corp",
	"mailboxStore": "LDAP://CN=Mailbox Database 0565232367,CN=Databases,CN=Exchange Administrative Group (FYDIBOHF23SPDLT),CN=Administrative Groups,CN=First,CN=Microsoft Exchange,CN=Services,CN=Configuration,DC=MYDOMAIN,DC=CORP",
	"createMailbox": true,
	"useDefaultMailboxStore": true
	}
	}

Response	{
	"user": {

	"userPassword": "HJv5MYkz$", "additionalAttributes": {}, "class": "User" }, "errors": [] }

Notes: To create a user that has an email address, specify the createMailbox=true flag.

Get

Basic

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

Payload	{
	"userIdentifier": "cn=User02,OU=SomeDept,DC=MYDOM,DC=CORP"
	}

Response	{
	"user": {
	"userPrincipalName": "User02@MYDOM.CORP",
	"fullName": "User 2 Jones",
	"description": "some account in the domain",
	"isDisabled": false,
	"class": "User",
	"distinguishedName": "cn=User02,OU=SomeDept,DC=MYDOM,DC=CORP ",
	"samAccountName": "user02"
	},
	"errors": []
	}

NOTE: When listing specific attributes, 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 users in a specific domain

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

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

Response	{
	"users": [{
	"additionalAttributes": {},
	"description": "some description",
	"class": "OU",
	"friendlyName": " OU05",
	"friendlyPath": "MYDOMAIN.CORP/Sharon/OU05",
	"isDisabled": true,
	"class": "User",

	"friendlyName": " USER05",
	"friendlyPath": "MYDOMAIN.CORP/Sharon/USER05",
	"isManaged": true,
	"distinguishedName": "CN=USER05,OU=SomeDept,DC=MYDOMAIN,DC=CORP",
	"sAMAccountName": "USER05",
	"displayName": "USER05",
	"winNTPath": "MYDOMAIN\\USER05"
	},
	Etc.
	],
	"resumeString": "",
	"totalNumberOfObjects": 12,
	"errors": []
	}

This example gets all users in the path Domain\Users. The default container for the domain/users/get request is the domain root. To search for users across multiple domains, see Enumeration examples.

Delete

URI	dra/domains/MyDomain.corp/users/delete
Payload	{
	"userIdentifier": "user10"
	}
Response	{"errors":[]}

Restore

To original container

URI	dra/domains/MyDomain.corp/users/restore
Payload	{
	"userIdentifier": "CN=User02,OU=NetIQRecycleBin,DC=MYDOMAIN,DC=CORP"
	}
Response	{"errors":[]}

To other container

URI	dra/domains/MyDomain.corp/users/restore
Payload	{
	"userIdentifier": "CN=user02,OU=NetIQRecycleBin,DC=MYDOMAIN,DC=CORP",
	"restoreToContainer": "OU=TempCon,DC=MyDomain,DC=CORP"
	}
Response	{"errors":[]}

Move

URI	dra/domains/MYDOMAIN.CORP/users/move

Payload	{ "userIdentifier": "CN=User02,OU=Tax,DC=MYDOMAIN,DC=CORP", "targetContainer": "OU=FINANCE,DC=MYDOMAIN,DC=CORP" }
Response	{"errors":[]}

Update

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 the 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":[]}

Notes: When the mailNickname is not provided, the alias will be generated by the DRA server to be compliant with 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":[]}

CreateMailbox

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

Payload	{
	"userIdentifier": "user06",
	"user": {
	"mailboxStore": "LDAP://CN=Mailbox Database 0565232367,CN=Databases,CN=Exchange Administrative Group (FYDIBOHF23SPDLT),CN=Administrative Groups,CN=First,CN=Microsoft Exchange,CN=Services,CN=Configuration,DC=MYDOMAIN,DC=CORP",
	"mailNickname": "User06"
	}
	}

Response	{"errors":[]}

CreateMailbox using the default Mailbox Store

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

Payload	{
	  "userIdentifier": "user06",
	  "user": {
	    "mailboxStore": "LDAP://CN=Mailbox Database 0565232367,CN=Databases,CN=Exchange Administrative Group (FYDIBOHF23SPDLT),CN=Administrative Groups,CN=First,CN=Microsoft Exchange,CN=Services,CN=Configuration,DC=MYDOMAIN,DC=CORP",
	    "mailNickname": "User06",
	    "useDefaultMailboxStore": true
	  }
	}

Response	{"errors":[]}

DeleteMailbox

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

Payload	{
	  "userIdentifier": "user06"
	}

Response	{"errors":[]}

Rename

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

Payload	{
	  "userIdentifier": "SomeObject,OU=SomeDept,DC=MyDomain,DC=corp",
	  "user": {
	    "description": "modified description 00",
	    "name": "Object_renamed"
	  }
	}

Response	{"errors":[]}

To rename an object, supply the identifier for the object and put the new name (and any other properties
you want to change) in the payload.

Copy

Change only the name in the target object

URI	dra/domains/MyDomain.corp/users/copies/post

Payload	{
	  "userIdentifier": "CN=SourceUser,OU=SomeDept,DC=MYDOMAIN,DC=CORP",
	  "user": {
	    "name": "DestinationUser"
	  }
	}

Response	{
	  "user": {
	    "class": "User",
	    "distinguishedName":
	"cn=DestinationUser,OU=SomeDept01,OU=SomeDept,DC=MYDOMAIN,DC=CORP"
	  },
	  "errors": []
	}

Change the name and some other properties in the target object

URI	dra/domains/MyDomain.corp/users/copies/post

Payload	{
	  "userIdentifier": "CN=SourceUser,OU=SomeDept,DC=MYDOMAIN,DC=CORP",
	  "user": {
	    "distinguishedName": "CN=DestinationUser,DC=MYDOMAIN,DC=CORP",
	    "description": "new desc",
	    "addToGroups": ["CN=somegroup,OU=Admin,DC=MYDOMAIN,DC=CORP"]
	  }
	}

Response	{
	  "user": {
	    "class": "User",
	    "distinguishedName": "cn=DestinationUser,DC=MYDOMAIN,DC=CORP"
	  },
	  "errors": []
	}

This payload will change the name, location, description of the new user. Also, the copied user will be added to the specified group.

Mail-enabled user

URI	dra/domains/MyDomain.corp/users/copies/post

Payload	{
	  "userIdentifier": "CN=User003ME,OU=Dept27,DC=MYDOMAIN,DC=CORP",
	  "user": {
	    "distinguishedName": "CN=User003MEDUP,OU=Dept88,DC=drdom150,DC=lab",
	    "emailAddress": "User003MEDUP@testmail.com"
	  }
	}

Response	{
	  "user": {
	    "class": "User",
	    "distinguishedName": "cn=User003MEDUP,OU=Dept88,DC=MYDOMAIN,DC=CORP"
	  },
	  "errors": []
	}

This payload shows the minimum properties required to copy an email-enabled user. For this copied user, the samAccountName, userPrincipalName, and mailNickName will be generated based on the name and domain of the target user.

User with a mailbox

URI	dra/domains/MyDomain.corp/users/copies/post

Payload	{
	  "userIdentifier": "CN=User003MB,OU=Dept27,DC=MYDOMAIN,DC=CORP",
	  "user": {
	    "distinguishedName": "CN=User003MBDUP,OU=Dept88,DC=drdom150,DC=lab",
	    "mailboxStore": "LDAP://MYDOMAIN.CORP/CN=Exchange etc. "
	  }
	}

Response	{
	  "user": {
	    "class": "User",
	    "distinguishedName": "cn=User003MBDUP,OU=Dept88,DC=MYDOMAIN,DC=CORP"
	  },
	  "errors": []
	}

The copied user’s mailbox will be created in the specified mailboxStore. If mailboxStore is not specified, the copied user’s mailbox will be created in the same store as the source user.

User with a mailbox using the Default Mailbox Store

URI	dra/domains/MyDomain.corp/users/copies/post

Payload	{
	  "userIdentifier": "CN=User003MB,OU=Dept27,DC=MYDOMAIN,DC=CORP",
	  "user": {
	    "distinguishedName": "CN=User003MBDUP,OU=Dept88,DC=drdom150,DC=lab",
	    "mailboxStore": "LDAP://MYDOMAIN.CORP/CN=Exchange etc. ",
	    "useDefaultMailboxStore": true
	  }
	}

Response	{ "user": { "class": "User", "distinguishedName": "cn=User003MBDUP,OU=Dept88,DC=MYDOMAIN,DC=CORP" }, "errors": [] }

The copied user’s mailbox will be created in the default mailbox Store.

Convert to Resource Mailbox

URI	dra/domains/MyDomain.corp/users/createmailbox/put
Payload	{ "userIdentifier": "CN=UserSource,OU=ConfAdmin,DC=MYDOMAIN,DC=CORP", "user": { "mailboxStore": "LDAP://MYDOMAIN.CORP/CN=Exchange etc. ", "resourceMailboxType": "Room" } }
Response	{ "user": { "class": "User", "distinguishedName": "cn=User003MBDUP,OU=Dept88,DC=MYDOMAIN,DC=CORP" }, "errors": [] }

To convert an existing user to a Resource Mailbox, you must add the parameter ‘resourceMailboxType’ to the request for creating a mailbox. Valid values for resourceMailboxType are “room” and “equipment”.

Get Mailbox Security Permissions

URI	dra/domains/MyDomain.corp/users/mailbox/get
Payload	{ "userIdentifier": "CN=UserSource,OU=ConfAdmin,DC=MYDOMAIN,DC=CORP" }

Response	{
	  "mailbox": {
	    "trustees": [{
	      "objectClass": "Group",
	      "preWindows2000Name": "NETWORK SERVICE",
	      "distinguishedName": "",
	      "inheritAllow": [
	        "ExRPerms"
	      ]
	    }],
	    "class": "Mailbox",
	  },
	  "errors": []
	}

To retrieve the mailbox NT security permissions, specify “isSecurity”:”true” immediately after the userIdentifier.
Without this flag, the Exchange security permissions are retrieved.
For mailbox Exchange security permissions, possible values for allow/deny are:
ExDelStor, ExCPerms, ExFullControl, ExAssocX, ExRPerms, ExTakeOwn.
For mailbox NT security permissions, possible values for deny/allow are: ntSendAs, ntReceiveAs.

Set Mailbox Security Permissions

URI	dra/domains/MyDomain.corp/users/mailbox/post

Payload	{
	  "userIdentifier": "CN=User7,DC=MYDOMAIN,DC=CORP",
	  "mailbox": {
	    "trustees": [{
	      "preWindows2000Name": "MYDOMAIN\\\\User18",
	      "allow": ["ExRPerms", "ExTakeOwn"],
	      "deny": ["ExAssocX"]
	    }, {
	      "preWindows2000Name": "MYDOMAIN\\\\User28B",
	      "allow": ["ExDelStor"]
	    }],
	  }
	}

Response	{
	  "draServerAndPort": "MYDOMAIN:11192",
	  "errors": []
	}

To set the mailbox NT security permissions, specify “isSecurity”:”true” immediately after the userIdentifier.
Without this flag, the Exchange security permissions are set.
For mailbox Exchange security permissions, possible values for allow/deny are:
ExDelStor, ExCPerms, ExFullControl, ExAssocX, ExRPerms, ExTakeOwn.
For mailbox NT security permissions, possible valuesfor deny/allow are: ntSendAs, ntReceiveAs.

VirtualAttributes

Create

URI	dra/virtualAttributes/post

Payload	{
	   "virtualAttribute": {
	      "name": "MyVA",
	      "description": "some good description",
	      "isMultiValued": true,
	      "attributeType": "string"
	   }
	}

Response	{"errors":[]}

VirtualAttributes can only be disabled, they cannot be deleted. Valid values for the *attributeType* are “string”, “bool”, and “int”.

Associate – Associate a virtual attribute to a class

URI	dra/virtualAttributes/associations/post

Payload	{
	   "virtualAttributeName": "MyVA",
	   "classToAssociateName": "Computer"
	}

Response	{"errors":[]}

Valid values for classToAssociateName: Computer, Group, OU, or User

Disassociate – Remove the association between a virtual attribute and a class

URI	dra/virtualAttributes/associations/delete

Payload	{
	   "virtualAttributeName": "MyVA",
	   "classToAssociateName": "Computer"
	}

Response	{"errors":[]}

Valid values for classToAssociateName: Computer, Contact, Group, OrganizationalUnit or User

Disable

URI	dra/virtualAttributes/disable/put

Payload	{
	   "virtualAttributeName": "MyVA"
	}

Response	{"errors":[]}

Enable

URI	dra/virtualAttributes/enable/put
Payload	{

	"virtualAttributeName": "MyVA" }
Response	{"errors":[]}

Enumeration

List all defined virtual attributes

URI	dra/virtualAttributes/get
Payload	{ }

Response	{
	"totalNumberOfObjects": 8,
	"virtualAttributes": [{
	"name": "Attribute03",
	"description": "some description",
	"attributeType": "string",
	"isMultiValued": true,
	"isEnabled": true
	},
	Etc.
	],
	"errors": []
	}

Get Virtual Attributes Associated With a Class

URI	dra/virtualAttributes/associations/get

Payload	{
	"className": "User",
	"isAssociated": true
	}

Response	{
	"totalNumberOfObjects": 8,
	"virtualAttributes": [{
	"name": "Attribute03",
	"description": "some description",
	"attributeType": "string",
	"isMultiValued": true,
	"isEnabled": true
	},
	Etc.
	],
	"errors": []
	}

Get Virtual Attributes Not Associated With a Class

URI	dra/virtualAttributes/associations/get

Payload	{
	"className": "User",
	"isAssociated": false
	}

Response	{
	"totalNumberOfObjects": 8,
	"virtualAttributes": [{
	"name": "Attribute07",
	"description": "some description",
	"attributeType": "string",
	"isMultiValued": true,
	"isEnabled": true
	},
	Etc.
	],
	"errors": []
	}

Exchange

Get Exchange Configuration

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

Response	{
	"exchangeConfiguration": {
	"class": "Exchange",
	"name": "module=Exchange",
	"nameValue": "Exchange",
	"parentPath": "OnePoint://",
	"path": "OnePoint://module=Exchange",
	"draSchemaPath": "OnePoint://cn=Exchange,cn=Exchange,module=Schema",
	"logCategoryMask": 0,
	"enableExchangePolicy": 1,
	"enableExchange2000SupportPolicy": 0,
	"enableExchange2007SupportPolicy": 0,
	"enableExchange2010SupportPolicy": 1,
	"enableExchange2013SupportPolicy": 1,
	"enableExchangeOnlineSupportPolicy": 0,
	"enforceExch2KAliasNamingConvention": 0,
	"enforceExchArchiveNamingConvention": 0,
	"enforceExchResourceNamingConvention": 0,
	"exch2KAliasNamingConvention": "",
	"exch2KAliasNamingConventionFlags": 0,
	"exchArchiveNamingConvention": "",
	"exchArchiveNamingConventionFlags": 0,
	"exchResourceNamingConvention": "",

	"exchResourceNamingConventionFlags": 0, "isExchangeBackdoorEnabled": false, "exchangeToolsInstalled": 20, "isExch2000Enabled": 0, "isExch2000SupportInstalled": 0, "isExch2007Enabled": 0, "isExch2007SupportInstalled": 0, "isExch2010Enabled": 1, "isExch2010SupportInstalled": 1, "isExch2013Enabled": 1, "isExch2013SupportInstalled": 1, "isExchOnlineEnabled": 0, "isExchOnlineSupportInstalled": 0, "isExch55Enabled": 0, "isX2KSP2OrLaterEnabled": 0, "mailboxPolicyDuringCreate": 0, "mailboxPolicyDuringDelete": 0, "mailboxPolicyDuringUpdate": 0, "prohibitSecurityGroupMailbox": 0, "additionalAttributes": {} }, "errors": [] }

GenerateExchangeName

URI	dra/domains/MyDomain.corp/exchangeName/alias/get

Payload	{
	"firstName": "Charlie",
	"lastName": "Winston",
	"initials": "CAW",
	"samAccountName": "WinstonCharlie"
	}

Response	{
	"generatedName": "WinstCh",
	"errors": []
	}

Notes:

- The alias is generated according to the naming policy configured on the DRA server. If no policy is configured, the samAccountName is returned.
- If the server is configured to enforce the naming convention, and sufficient properties are not sent in the request to generate a policy-compliant name, the request will fail with a general ‘operation failed’ server error.
- When enabling email or creating a mail-enabled object (user, group or contact), if you do not supply the mailNickname property, the REST endpoint will call this method to obtain a value from the server for the mailNickname property.

List Exchange Servers

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

Response	{
	  "ExchangeServers": [{
	      "distinguishedName": "CN=IDCDVDR152,CN=Servers,CN=Exchange Administrative Group (FYDIBOHF23SPDLT),CN=Administrative Groups,CN=NetIQ Organization,CN=Microsoft Exchange,CN=Services,CN=Configuration,DC=MyDomain,DC=Corp",
	      "legacyExchangeDN": "/o=NetIQ Organization/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Configuration/cn=Servers/cn=IDCDVDR152",
	      "ldapPath": "LDAP://IDCDVDR150.MYDOMAIN.CORP/CN=IDCDVDR152,CN=Servers,CN=Exchange Administrative Group (FYDIBOHF23SPDLT),CN=Administrative Groups,CN=NetIQ Organization,CN=Microsoft Exchange,CN=Services,CN=Configuration,DC=MYDOMAIN,DC=CORP",
	      "name": "IDCDVDR152",
	      "additionalAttributes": {}
	    },
	    Etc.
	  ],
	  "numberOfRowsReturned": 4,
	  "errors": []
	}

The value of ***ldapPath*** can be used to request the mailbox stores on this server.

List Exchange Admin Groups

URI	dra/domains/MyDomain.corp/exchangeServers/adminGroups/get
Payload	{ }

Response	{
	  "AdminGroups": [{
	    "distinguishedName": "CN=Exchange Administrative Group (FYDIBOHF23SPDLT),CN=Administrative Groups,CN=NetIQ Organization,CN=Microsoft Exchange,CN=Services,CN=Configuration,DC=DRDOM150,DC=LAB",
	    "legacyExchangeDN": "/o=NetIQ Organization/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)",
	    "ldapPath": "LDAP://IDCDVDR150.DRDOM150.LAB/CN=Exchange Administrative Group (FYDIBOHF23SPDLT),CN=Administrative Groups,CN=NetIQ Organization,CN=Microsoft Exchange,CN=Services,CN=Configuration,DC=DRDOM150,DC=LAB",
	    "name": "Exchange Administrative Group (FYDIBOHF23SPDLT)",
	    "additionalAttributes": {}
	  }],
	  "numberOfRowsReturned": 1,
	  "errors": []
	}

List Mailbox Stores on an Exchange Server

URI	dra/domains/MyDomain.corp/exchangeServers/mailboxStores/get
Payload	{ "exchangeServerLdapPath": "LDAP://IDCDVDR150.MYDOMAIN.CORP/CN=IDCDVDR152,CN=Servers,CN=Exchange Administrative Group (FYDIBOHF23SPDLT),CN=Administrative Groups,CN=NetIQ Organization,CN=Microsoft Exchange,CN=Services,CN=Configuration,DC=MYDOMAIN,DC=CORP" }
Response	{ "MailboxStores": [{ "distinguishedName": "CN=Exchange 2010,CN=Databases,CN=Exchange Administrative Group (FYDIBOHF23SPDLT),CN=Administrative Groups,CN=NetIQ Organization,CN=Microsoft Exchange,CN=Services,CN=Configuration,DC=DRDOM150,DC=LAB", "legacyExchangeDN": "/o=NetIQ Organization/ou=Exchange Administrative Group (FYDIBOHF23SPDLT)/cn=Configuration/cn=Servers/cn=IDCDVDR152/cn=Microsoft Private MDB", "ldapPath": "LDAP://IDCDVDR150.DRDOM150.LAB/CN=Exchange 2010,CN=Databases,CN=Exchange Administrative Group (FYDIBOHF23SPDLT),CN=Administrative Groups,CN=NetIQ Organization,CN=Microsoft Exchange,CN=Services,CN=Configuration,DC=DRDOM150,DC=LAB", "name": "Exchange 2010", "additionalAttributes": {} }, Etc.], "numberOfRowsReturned": 4, "errors": [] }

Office 365

Enable Office 365 Archive mailbox

URI	dra/domains/MyDomain.corp/users/office365/mailbox/enablearchive
Payload	{ "userIdentifier": "office365User22" }
Response	{"errors":[]}

Disable Office 365 Archive mailbox

URI	dra/domains/MyDomain.corp/users/office365/mailbox/disablearchive
Payload	{ "userIdentifier": "office365User22" }

Response	{"errors":[]}

Search (Enumeration)

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 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 in all of 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 of 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": []
}
```

Handling Multi-Page Results

Enumeration specifying paging values and specific attributes to retrieve

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

Payload

```
{
   "enumerationOptions": {
      "objectsPerResponse": "5",
      "managedContainerEnum": "false",
      "resumeString": "CN=Accounting-DG,DC=MYDOMAIN,DC=CORP"
   },
   "attributes": ["info", "displayNamePrintable", "groupType", "managedBy"]
}
```

Response

```
{
   "groups": [{
         "info": null,
         "displayNamePrintable": null,
         "groupType": 2,
         "managedBy": "CN=Christa Daugherty,OU=Sales,DC=MYDOMAIN,DC=CORP",
         "class": "Group"
      },
      Etc.
   ]
}
```

Note: *managedContainerEnum* is optional. Default is false. When “true”, the DRA server will enumerate only managed containers. This is needed when the admin has rights over only some of the domain tree.

Searching the Recycle Bin

Recycle bins can be searched for any object type that can be placed in a recycle bin. Organizational units, containers, domains, etc. cannot be placed in the DRA Recycle Bin. Sending filters for these types of objects does not cause an error, they are just ignored.

All objects in a recycle bin exist as a flat list in the root of that recycle bin. Therefore setting the includeChildContainers member of of the enumerationOptions object has no effect on the search.

Individual recycle bins may be searched using the dra/managedObjects endpoint and specifying the recycle bin to search in the enumerationOptions.

Searching a Specific Recycle Bin

URI	dra/recyclebins/managedObjects/get

Payload

```
{
   "attributes": [
      "friendlyName",
      "distinguishedName",
   ],
   "enumerationOptions": {
      "containerDistinguishedName": "OU=NetIQRecycleBin,DC=DRDOM150,DC=LAB"
   },
   "userOrFilter": {
      "friendlyName": "*"
   },
   "groupOrFilter": {
      "friendlyName": "*"
   }
}
```

Response

```
{
   "computers": [],
   "contacts": [],
   "domains": [],
   "groups": [
      {
         "distinguishedName": "CN=group1,OU=NetIQRecycleBin,DC=DRDOM150,DC=LAB",
         "friendlyName": "group1",
         "additionalAttributes": {
         }
      },
      {
         "distinguishedName": "CN=group5,OU=NetIQRecycleBin,DC=DRDOM150,DC=LAB",
         "friendlyName": "group5",
         "additionalAttributes": {
         }
      }
   ],
   "dynamicDistributionGroups": [],
   "ous": [],
```

	"users": [
	{
	"distinguishedName":
	"CN=Employee_1,OU=NetIQRecycleBin,DC=DRDOM150,DC=LAB",
	"friendlyName": "Employee_1",
	"additionalAttributes": {
	}
	},
	],
	"equipmentMailboxes": [],
	"roomMailboxes": [],
	"resumeString": "",
	"totalNumberOfObjects": 3,
	"numberOfRowsReturned": 3,
	"isSearchFinished": true,
	"errors": []
	}

This search looks for all users and groups in the Recycle Bin for drdom150.lab and returns the distinguished name and friendly name of each.

Searching Across All Recycle Bins

URI	dra/recyclebins/managedObjects/get

Payload	{
	"attributes": [
	"distinguishedName",
	],
	"userOrFilter": {
	"friendlyName": "*"
	}
	}

Response	{
	"computers": [],
	"contacts": [],
	"domains": [],
	"groups": [],
	"dynamicDistributionGroups": [],
	"ous": [],
	"users": [
	{
	"distinguishedName":
	"CN=033120130751,OU=NetIQRecycleBin,DC=drdom820,DC=lab",
	"additionalAttributes": {
	}
	},
	{
	"distinguishedName":
	"CN=033120150757,OU=NetIQRecycleBin,DC=drdom820,DC=lab",

```
          "additionalAttributes": {
          }
        },
        {
          "distinguishedName": "CN=04032015-
0920,OU=NetIQRecycleBin,DC=DRDOM691,DC=LAB",
          "additionalAttributes": {
          }
        },
       {
          "distinguishedName": "CN=04032015-
1920,OU=NetIQRecycleBin,DC=DRDOM150,DC=LAB",
          "additionalAttributes": {
          }
        }
    ],
    "equipmentMailboxes": [],
    "roomMailboxes": [],
    "resumeString": "",
    "totalNumberOfObjects": 4,
    "numberOfRowsReturned": 4,
    "isSearchFinished": true,
    "errors": []
}
```

This search looks for all users in all Recycle Bins and returns the distinguished name of each.

Requesting other DRA operations

It’s possible to request any DRA operation by providing a payload that contains all of the values the DRA server expects in the varset. All values must be formatted exactly as the server expects them. For example, path variables will typically need to be prefixed by “OnePoint://”.

The data returned in the response is likewise presented exactly as the server sends it.

URI	/dra/operations/varset/post

Payload (example)

```
{
   "varsetData": {
      "OperationName": "SecurityAssignmentEnum",
      "Type": "AdminAssignment",
      "AA": "OnePoint://CN=Administrator,CN=Users,DC=MYDOMAIN,DC=CORP",
      "ResumeStr": "",
      "nextrows": 250,
      "Scope": 0,
      "Hints": [
         "$McsNameValue",
         "$McsClass",
         "$McsPath",
         "Comment",
```

	"Description", "$McsSysFlag", "$McsIsAssigned", "$McsHidden", "PowerTemplate", "OperationDescription", "IsClonable", "$McsFriendlyName", "$McsFriendlyPath", "$McsHasMailbox", "AccountDisabled", "sAMAccountName", "$McsObjectClass", "groupType", "Name", "Status"] } }

Response (example)	{
	"returnedVarset": {
	"ClientName": "CN=Administrator",
	"ClientPath": "OnePoint://CN=Administrator,CN=Users,DC=MYDOMAIN,DC=CORP",
	"Errors": null,
	"Errors.LastError": 0,
	"Errors.NumErrors": 0,
	"IEaEnumerateBuf": [{
	"$McsNameValue": "Audit All Objects",
	"$McsClass": "Role",
	"$McsPath": "OnePoint://role=Audit All Objects,module=Security",
	"Comment": null,
	"Description": "Allows you to view the properties of all objects, whether security, policy, triggers, or configuration.",
	"$McsSysFlag": true,
	"$McsIsAssigned": true,
	"$McsHidden": false,
	"PowerTemplate": null,
	"OperationDescription": null,
	"IsClonable": null,
	"$McsFriendlyName": null,
	"$McsFriendlyPath": null,
	"$McsHasMailbox": null,
	"AccountDisabled": null,
	"sAMAccountName": null,
	"$McsObjectClass": null,
	"groupType": null,
	"Name": null,
	"Status": null

```
        }, {
            "$McsNameValue": "All Objects",
            "$McsClass": "ActiveView",
            "$McsPath": "OnePoint://av=All Objects,module=Security",
            "Comment": null,
            "Description": "Includes managing any aspect of your enterprise including all objects
in all managed domains. You can assign this ActiveView to the administrator or to an
Assistant Admin who needs auditing powers across the enterprise.",
            "$McsSysFlag": true,
            "$McsIsAssigned": null,
            "$McsHidden": false,
            "PowerTemplate": null,
            "OperationDescription": null,
            "IsClonable": null,
            "$McsFriendlyName": null,
            "$McsFriendlyPath": null,
            "$McsHasMailbox": null,
            "AccountDisabled": null,
            "sAMAccountName": null,
            "$McsObjectClass": null,
            "groupType": null,
            "Name": null,
            "Status": null
        }],
        "NumberOfRows": 2,
        "Result": "All Done",
        "ResumeStr": "",
        "TotalNumberObjects": 1,
        "Warnings": null,
        "Warnings.LastWarning": 0,
        "Warnings.NumWarnings": 0
    },
    "errors": []
}
```