Citrix ADM NITRO Developer’s Guide for REST API

Last Updated: March 2018

Document code: March 2018

Contents

Chapter 1

NITRO API

The Citrix® ADM NITRO protocol allows you to configure and monitor the ADM programmatically.

NITRO exposes its functionality through Representational State Transfer (REST) interfaces. This ensures that the NITRO functionality can be accessed by applications developed in any programming language. Additionally, for applications that must be developed in Java or .NET, the NITRO protocol is exposed as Java and .NET libraries that are packaged as separate Software Development Kits (SDKs).

Obtaining the Latest NITRO Package

The latest NITRO package is available as a tar file on the **Downloads** page of the ADM server's configuration utility. You must download and un-tar the file to a folder on your local system. This folder is referred to as <NITRO_SDK_HOME> in this documentation.

The folder contains the NITRO libraries (JARs for Java and DLLs for .NET) in the lib subfolder. The libraries must be added to the client application's classpath to access NITRO functionality. The <NITRO_SDK_HOME> folder also provides samples and documentation that can help you understand the NITRO SDK.

Note: The REST package contains only documentation for using the REST interfaces.

Prerequisites

To use the NITRO protocol, the client application needs only the following:

- Access to an ADM, release 12.1.
- A system to generate HTTP or HTTPS requests (payload in JSON format) to the ADM. You can use any programming language or a tool to generate the requests.
- For Java clients, you must have a system where Java Development Kit (JDK) 1.5 or later is available. The JDK

can be downloaded from http://www.oracle.com/ technetwork/java/javase/downloads/index.html.

- For .NET clients, you must have a system with .NET framework 3.5 or later installed. The .NET framework can be downloaded from http://www.microsoft.com/ downloads/en/default.aspx.

Note: You must have a basic understanding of the ADM server before using the NITRO protocol.

Chapter 2

Execution Flow

The NITRO protocol consists of the client application and the NITRO web service, which runs on the ADM.
The communication between the client application and the NITRO web service is based on REST architecture using HTTP or HTTPS.

Invocation of a NITRO REST request initiates the following processes:

1. The application sends REST request messages to the NITRO web service.
2. The NITRO web service processes the requests and returns a corresponding REST response message to the client application.

To minimize traffic on the ADM network, you retrieve the whole state of a resource from the server, make modifications to the state of the resource locally, and then upload it back to the server in one network transaction. For example, to update a mpsuser resource, you must retrieve the object, update the properties, and then upload the changed object in a single transaction.

Note: Local operations on a resource (changing its properties) do not affect its state on the server until the state of the object is explicitly uploaded.

REST Web Services

REST (REpresentational State Transfer) is an architectural style based on simple HTTP requests and responses between the client and the server. REST is used to query or change the state of objects on the server side. In REST, the server side is modeled as a set of entities where each entity is identified by a unique URL.

Each resource also has a state on which the following operations can be performed:

1. **Create.** Clients can create new server-side resources on a "container" resource. You can think of container resources as folders, and child resources as files or subfolders. The calling client provides the state for the resource to be created. The state can be specified in the request by using XML or JSON format. The client can also specify the unique URL that will identify the new object. Alternatively, the server can choose and return a unique URL identifying the created object. The HTTP method used for create requests is POST.
2. **Read.** Clients can retrieve the state of a resource by specifying its URL with the HTTP GET method. The response message contains the resource state, expressed in JSON format.
3. **Update.** You can update the state of an existing resource by specifying the URL that identifies that object and its new state in JSON or XML, using the PUT HTTP method.
4. **Delete.** You can destroy a resource that exists on the server-side by using the DELETE HTTP method and the URL identifying the resource to be removed.

The general format for NITRO URLs is as follows:

http://<adm-ip-address>/nitro/v2/config/<resource-type>

For example, for an ADM User, <resource-type> can be replaced by mpsuser. Some points to remember:

- In addition to the CRUD operations (Create, Read, Update, and Delete), resources (such as managed_device) can support other operations or actions. These operations use the HTTP POST method, with the URL specifying the operation to be performed and the request body specifying the parameters for that operation.

For more information on the REST objects and the usage, see the documentation provided in the <NITRO_SDK_HOME>/index.html file.

Chapter 3

API Categorization and Usage

Topics:

The NITRO functionality can be grouped into the following:

- System Management on page 12
- Configurations on page 13
- Exception Handling on page 19

Note: All NITRO requests are synchronous. After sending a request, the client application blocks until it receives a response from the NITRO web service.

System Management

This section describes the system level operations that can be performed on the ADM.

- Connect to the ADM on page 12
- Disconnect from the ADM on page 12

The following table describes the system level operations. For a more detailed description of the operations, see the API reference available in the <NITRO_SDK_HOME>/doc folder.

Connect to the ADM (On-Prem)

Before you perform any operation, you must authenticate and establish a session with the appliance.

Description	Sample
To connect to the appliance, specify the username and password in the login object. The Nitro Auth Token that is created must be specified in the cookie field of all requests in the session. To ensure secure communication, use the HTTPS protocol in NITRO requests. You must have a user account on that appliance. The configurations you can perform are limited by the administrative role assigned to your account. **Note:** By default, the connection with the appliance expires after 15 minutes of inactivity. To change the session timeout period, specify the new timeout period in the login object.	To connect to an ADM with IP address 10.102.31.16 by using the HTTP protocol: • **URL.** http://10.102.31.16/nitro/v2/config/login • **HTTP Method.** POST • **Request Payload.** `{ "login": { "username":"nsroot", "password":"verysecret" } }` • **Response:** HTTP Status Code on success - 200 OK HTTP Status Code on Failure - 4xx <string> (for general HTTP errors) or 5xx <string> (for ADM- specific errors). The response payload provides details of the error. • **Response Header** Set-Cookie: NITRO_AUTH_TOKEN=##88374218… • **Response Payload.**

	`{ "login": [ { "token": 0, "tenant_name" :"Owner", "sessionid":"##78C060..." } ] }`

Connect to the ADM (Service)

Before you perform any operation, you must authenticate and establish a session with the appliance.

Description	Sample
To connect to the ADM Service, specify the ID and Secret in the login object. The Nitro Auth Token that is created must be specified in the cookie field of all requests in the session. To ensure secure communication, use the HTTPS protocol in NITRO requests. You must have a user account on that appliance. The configurations you can perform are limited by the administrative role assigned to your account. **Note:** By default, the connection with the appliance expires after 15 minutes of inactivity. To change the session	To connect to the ADM Service by using the HTTPS protocol: • **URL.** https://netscalermas.cloud.com/nitro/v2/config/login • **HTTP Method.** POST • **Request Headers:** isCloud: true • **Request Payload.** `{ "login": { "ID":"Client ID", "Secret":"Secret" } }` • **Response:** HTTP Status Code on success - 200 OK

timeout period, specify the new timeout period in the login object.	HTTP Status Code on Failure - 4xx <string> (for general HTTP errors) or 5xx <string> (for ADM- specific errors). The response payload provides details of the error. • **Response Header** Set-Cookie: NITRO_AUTH_TOKEN=##88374218… • **Response Payload.** `{ “login”:[ { “token”:0, “tenant_name”:”Owner”, “sessionid”:”##78C060…” } ] }`

Disconnect from the ADM On-prem appliance/ADM Service

When the application does not need to interact with the appliance, logging off from the appliance is a good practice.

Description	Sample
To disconnect from the appliance, use the POST HTTP method with delete action.	To disconnect from an ADM with IP address 10.102.31.16: • **URL.** http://10.102.31.16/nitro/v2/config/logout • **HTTP Method**. DELETE • **Cookie.** NITRO_AUTH_TOKEN=##78C060... To disconnect from the ADM Service • **URL.** https://netscalermas.cloud.com/nitro/v2/config/logout • **HTTP Method**. DELETE • **Cookie.** NITRO_AUTH_TOKEN=##78C060...

Modifying the Session Timeout

You can modify the timeout period by specifying a new timeout period (in seconds) in the login object. For example, to modify the timeout period to 60 minutes:

```
{
        "login":
        {
                "username":"<username>",
                "password":"<password>",
                "timeout":"3600"
        }
}
```

Some points to note with regards to session timeout for ADM <VERSION> and later versions:

- When restricted timeout param is enabled, NITRO, by default, uses the timeout value that is configured for the logged in user. You can customize this value but it must be limited to the value specified for the user. If no value is specified for the user, the default timeout value of 15 minutes is used.
- When restricted timeout param is not enabled, NITRO uses the default value of 30 minutes as session timeout.

Configurations

An ADM can support multiple resources. The NITRO protocol can be used to configure these resources.

- **For Java and .NET.** The APIs to configure a resource are grouped into packages or namespaces that have the format com.citrix.mas.nitro.resource.config.<resource_type>. Each of these packages contain a class named <resource_type> that provides the APIs to configure the resource.

  For example, the ADM resource mpsuser is under the package mps com.citrix.mas.nitro.resource.config.mps package or namespace.
- **For REST.** Each ADM resource has an unique URL associated with it, depending on the type of operation to be performed. URLs for configuration operations have the format http://<ADM-IP>/nitro/v2/config/ <resource_type>.

  For example, to configure an ADM resource, the URL is http://10.102.31.16/ nitro/v2/config/mpsuser.

This section describes the configurations that can be performed on the ADM

- Create Resource on page 13
- Retrieve Resource Details on page 14
- Retrieve Resource Statistics on page 16
- Update Resource on page 17
- Delete Resource on page 17
- Bulk Operations on page 17

The following tables describe the configurations operations that can be performed. For a more detailed description of the operations, see the API reference available in the <NITRO_SDK_HOME>/doc folder.

Create resource

Description	Sample
To create a new resource (for example, a managed_device instance) on the ADM, specify the resource name and other related arguments in the specific resource object	To create an instance of managed_device resource where managed_device is used to manage: • **URL.** http://10.102.31.16/nitro/v2/config/managed_deivce*?action=add_device* • **HTTP Method.** POST • **Cookie.** NITRO_AUTH_TOKEN=##78C060... • **Request Payload.** `{ "managed_device": { "ip_address":"10.106.101.112", "profile_name":"ns_nsroot_profile" } }`

Retrieving details of ADM resources

NITRO provides multiple approaches using which you can retrieve resources and their relevant details. The following table explains each of these approaches with the required URL.

Note: A sample format of the request and response is provided below the table.

Header **isCloud:true** is mandatory in case of retrieving details from ADM Service.

Retrieving all details of all resources of a specific type	In the URL, specify the type of resource for which you want to retrieve the details. For example, to retrieve all details of mpsuser resources available on an ADM Appliance http://<adm-ip-address>/nitro/v2/config/mpsuser
Retrieving all details of a specific resource	In the URL, specify the name of resource for which you want to retrieve the details. For example, to retrieve all details of a mpsuser resource whose id is <id_value> http://<adm-ip-address>/nitro/v2/config/mpsuser /<id_value>
Retrieving all details of resources that have Multiple unique identifiers	In the URL, specify the type of resource and use the "args" query parameter to specify the unique attributes and the values for those attributes. For example, to get the mpsuser resources that have same tenant name and belonging to the same group http://<adm-ip-address>/nitro/v2/config/mpsuser?args=tenant_name:Owner,groups:owner
Retrieving specific details of all resources of a specific type	In the URL, specify the type of the resource and use the "attrs" query parameter to specify the resource details that you want to retrieve. For example, to retrieve the "name" and "id" of all mpsuser resources: http://<adm-ip-address>/nitro/v2/config/mpsuser?attrs=name,id
Retrieving specific details of a specific resource	In the URL, specify the type and name of the resource and use the "attrs" query parameter to specify the resource details that you want to retrieve. For example, to retrieve the "name" and "id" of a mpsuser resource whose id is <id_value> : http://<adm-ip-address>/nitro/v2/config/mpsuser/<id_value>?attrs=name,id
Filtering the retrieved resources	In the URL, specify the type of resource and use the "filter" query parameter to specify the attribute(s) and the value(s) of the attributes. The resources fetched will be filtered based on the filter criteria. Note: The filter query parameter supports the use of PCRE regular expressions. For example, to filter the mpuser resources based on the group named “owner” http://<adm-ip-address>/nitro/v2/config/mpsuer?filter=groups:owner

Retrieving resources in paginated manner	If the request is likely to result in a large number of resources, you can divide the results into pages and retrieve them page by page (paginated). For example, if you are retrieving, say 40, managed_device resources of an ADM appliance, instead of retrieving all 40 in one response, you can configure the results to be divided into 4 pages each having 10 results. In the URL, specify the name of the resource and use the following query parameters: • "pageno" - The page number to be displayed. • "pagesize" - The number of resources to be displayed in each page. For example, to retrieve the manage_device in a paginated form, first get a count (using the "count" query parameter shown in below row) of the managed devices. Then, accordingly specify the number of results for each page and then specify the page number to be displayed. http://<adm-ip-address>/nitro/v2/config/managed_device?pagesize=10&pageno=3

Examples:

Retrieving the details of a specific resource

Description	Sample
To retrieve the details of a specific resource, specify the resource along with ID that identifies the particular resource in the URL	To get details of managed device whose ID is 03ba3316-05f9-4dc7-93e5-41d7084a83, • **URL.** http://10.102.31.16/nitro/v2/config/managed_deivce/03ba3316-05f9-4dc7-93e5-41d7084a83 • **HTTP Method.** GET • **Cookie.** NITRO_AUTH_TOKEN=##78C060... • **Response Payload.** `{ "managed_device": [ { "subnet_id": "", "manufacturedate": "2/17/2009", "is_grace": "false", "hostname": "adc_1", "std_bw_config": "0", "gateway_deployment": "false", "gateway_ipv6": "", "ha_master_state": "Primary", … } ] }`

Retrieving specific details of all managed_device resources of ADM

Description	Sample
To retrieve the specific details of all resource of one type, specify the resource along with ?attrs=prop1,prop2 in the URL.	To get ip_address and id of all managed_device resources, • **URL.** http://10.102.31.16/nitro/v2/config/managed_deivce?attrs=ip_address,id • **HTTP Method.** GET • **Cookie.** NITRO_AUTH_TOKEN=##78C060... • **Response Payload.** `{ "managed_device": [ { "id": "19ea448c-d258-4dfc-aecb--a85af6cc1e99", "ip_address": "10.106.101.112" }, { "id": "221a97f5-fd45-457a-8798-28ed93012ab7", "ip_address": "10.102.61.138-p3" } ] }`

Getting the Count of ADM Resources

If you want to have an idea of the number of resources that are likely to be returned by a request, you can use the count query string parameter to ask for a count of the resources to be returned, rather than the resources themselves.
http://<ADM_IP>/nitro/v2/ config/<resource_type>? count=yes

Description	Sample
To get a count of a specific resource type, in the URL specify the count query parameter as "yes".	To get a count of all managed_devices: • **URL.** http://10.102.31.16/nitro/v2/config/managed_device?count=yes • **HTTP Method.** GET • **Cookie.** NITRO_AUTH_TOKEN=##78C060... • **Response Payload.** `{ "managed_device": [ { "__ count": 4 } ] }`

Update Resource

Description	Samp le

To update an existing ADM resource, use the PUT HTTP method. In the HTTP request payload, specify the name and the other arguments that have to be changed.	To change the name of managed_device instance with ID 03ba3316-05f9-4dc7-93e5- 41d7084a83d2 to dev2:
	• **URL.** http://10.102.31.16/nitro/v2/config/managed_d eivce
	• **HTTP Method.** PUT
	• **Cookie.** NITRO_AUTH_TOKEN=##78C060...
	• **Request Payload.**
	{
	"managed_device": [
	{
	"name":"dev2",
	"id":" 03ba3316-05f9-4dc7-93e5- 41d7084a83d2 "…
	}
	]
	}

Note: Some properties in some ADM resources cannot be modified after creation. The username of an ADC instance is an example. If the request payload of the upload operations includes such properties, NITRO does not return an error. The values provided for these properties are ignored.

Delete Resource

Description	Sample
To delete an existing resource, specify the name of the resource to be deleted in the URL.	To delete a managed_device instance with ID 03ba3316-05f9-4dc7-93e5-41d7084a83d2 • **URL.** http://10.102.31.16/nitro/v2/config/managed_deivce/03ba3316-05f9-4dc7-93e5-41d7084a83d2 • **HTTP Method.** DELETE • **Cookie.** NITRO_AUTH_TOKEN=##78C060...

Deleting a User Session

Description	Sample

To delete a user session identified by session id, an instance of mpssession has to be deleted by specifying the session id in the URL.	To delete a mpssession instance with ID 85326d1c-53bc-4836-a265-31489408cdec
	• **URL.** http://10.102.31.16/nitro/v2/config/logout
	• **HTTP Method.** DELETE
	• **Cookie.** NITRO_AUTH_TOKEN=##78C060...
	• **Response Payload.**
	`{`
	`    "logout": [`
	`        {`
	`            "session_timeout": "0",`
	`            "sessionid": "",`
	`            "username": ""`
	`        }`
	`    ]`
	`}`

Bulk Operations

You can query or change the configuration of resources simultaneously by, for example adding multiple mpsusers in the same operation. This minimizes the network traffic.

NITRO supports the following behavior in bulk operations:

- **Exit.** When the first error is encountered, the execution stops. The commands that were executed before the error are committed.
- **Continue.** All the commands in the list are executed even if some commands fail.

Description	Sample

To perform a bulk operation, specify the required parameters in the same request payload.
 	
You can specify the behaviour of the bulk operation in the request header using the X-NITRO-ONERROR parameter
 	
Note: You can also add resources of different types in one request.
	To add 2 ADM resources in one operation:
	
	• **URL.** http://10.102.31.16/nitro/v2/config/mpsuser
	• **HTTP Method.** POST
	• **Cookie.** NITRO_AUTH_TOKEN=##78C060...
	• **Response Payload.**
	`{ "mpsuser": [ { "name":"new_name", "password":"newsceret" }, { "name":"new_name1", "password":"newsecret1" } ] }`
	
	To add multiple resources (two ADC and two mpsusers) in one operation:
	• **URL.** http://10.102.31.16/nitro/v2/config/
	• **HTTP Method.** POST
	• **Cookie.** NITRO_AUTH_TOKEN=##78C060...
	• **Response Payload.**
	`{ "ns": [ { "name":"ns_instance1", "ip-address":"10.70.136.5", "netmask":"255.255.255.0", "gateway":"10.70.136.1" }, { "name":"ns_instance2", "ip-address":"10.70.136.8", "netmask":"255.255.255.0", "gateway":"10.70.136.1" } ],`

```
    "mpsuser":
    [
            {
            "name":"admin",
            "password":"admin",
            },
            {
            "name":"admin1",
            "password":"admin1"
            }
    ]

}
```

Performing File Operations

ADM NITRO allows you to perform file operations such as uploading files, retrieving files, retrieving file content, and deleting files of types: txt, cert, req, xml, lic, and key.

Note:

- Use the "BASE64" value for the file encoding attribute in the request payload. This is the only valid encoding currently supported.
- The file location path must be URL encoded. For example, if the path is /nsconfig/ssl, encode the / and use the file location as %2Fnsconfig%2Fssl.
- When uploading a file, make sure that each directory of the file path has the 755 (read, write, execute) permission.

Uploading a File

Description	Sampl e

To upload a file to the ADM, specify a name for the file, the location where the file must be created on the ADM, and the content of the file.	To upload SSL certificate, ssl_cert resource should be specified in URL and file should be sent as multi – part form data
	
	• **URL.** http://10.102.31.16/nitro/v2/config/ssl_cert
	• **HTTP Method.** POST
	• **Cookie.** NITRO_AUTH_TOKEN=##78C060...
	• **Request Payload:** Multi-part form data with File Stream
	• **Response Payload.**
	{
	"ssl_cert": [
	"file_name": <String_value>
	]
	}

Retrieving the Files

Description	Sample
To retrieve the files from a specific ADM directory, specify the directory path in the URL.	To retrieve SSL certificates, ssl_cert resource and file name should be specified in URL. • **URL.** http://10.102.31.16/nitro/v2/config/ *download/ssl_cert/file_name_value<String>* • **HTTP Method.** GET • **Cookie.** NITRO_AUTH_TOKEN=##78C060... • **Response Payload.** Binary Stream

Deleting a File

Description	Sample

To delete a file from the ADM appliance, specify the filename and the directory path in the URL.	To delete SSL certificate file, ssl_cert resource and file name should be specified in URL.
	• **URL.** http://10.102.31.16/nitro/v2/config/*ssl_cert/file_name_value<String>*
	• **HTTP Method.** DELETE
	• **Cookie.** NITRO_AUTH_TOKEN=##78C060...
	• **Response Payload.**
	`{`
	`"ssl_cert": [`
	`"tenant_name": <String_value>,`
	`"file_name": <String_value>`
	`]`
	`}`

Exception Handling

The errorcode field indicates the status of the operation.

- An errorcode of 0 indicates that the operation is successful.
- A non-zero errorcode indicates an error in processing the NITRO request.

The error message field provides a brief explanation and the nature of the failure. By default, NITRO captures only error messages. You can capture warnings by specifying the warning flag while establishing a connection with the appliance.

Description	Sample
The response payload of all operations, specifies the error code and error message.	To get the status of an operation: • **URL.** http://10.102.31.16/nitro/v2/config/*ping* • **HTTP Method.** GET • **Cookie.** NITRO_AUTH_TOKEN=##78C060... • **Response Payload.** `{ "errorcode":-1, "message":"IP address is missing" }`

For a more detailed description of the error codes, see the API reference available in the <NITRO_SDK_HOME>/doc folder.

Error in a Single Resource Operation

HTTP Status Code
4xx <string> (for general HTTP errors) or 5xx <string> (for ADM-specific errors) The

response of a single erroneous operation is as follows:

Response Payload

```
{
        errorcode: <Error code>
        message: "<Error
        message>"
        severity: "ERROR"
}
```

Error in a Bulk Operation

When there is a failure in one of the bulk operations, the response payload gives a combination of success and failure (depends on the value set for X-NITRO-ONERROR in the request header).

HTTP Status Code
    207 Multi Status

Response Payload when X-NITRO-ONERROR is set to continue

When the first operation fails, the request is not terminated. The response payload shows the error details of the failed operation and the success status of the other operations.

```
{
        "errorcode": 1243,
        "message": "Bulk operation failed",
        "severity": "ERROR",
        "response":
        [
                {
                        "errorcode": 273,
                        "message": "Resource already exists",
                        "severity": "ERROR"
                },
                {
                        "errorcode": 0,
                        "message": "Done",
                        "severity": "NONE"
                }
        ]
}
```

Response Payload when X-NITRO-ONERROR is set to exit

When the first operation fails, the request is terminated. The response payload only shows the error details of the failed operation.

```
{
        "errorcode": 1243,
        "message": "Bulk operation failed",
        "severity": "ERROR",
        "response":
        [
            {
                        "errorcode": 273,
                        "message": "Resource already exists", "severity":
                        "ERROR"
            }
         ]
}
```

Java, .NET, and Python API

This section provides basic information for using the Java, .NET, and Python SDKs that are provided for the NITRO API. The API are categorized on their scope and purpose.

Important

All NITRO operations are logged in the /var/mps/log/mps_service.log file on the appliance.

Executable samples are available in the *<NITRO_SDK_HOME>/sample* directory.

Tutorial: Create Your First NITRO Application

After completing this tutorial, you will understand and be able to use NITRO to log in to the ADM appliance, create resources, retrieve details of a resource, delete a resource, schedule configuration jobs, perform configuration audit, save the configurations, and log out of the ADM appliance.

- Using Java API to Create your First NITRO Application
- Using .NET API to Create your First NITRO Application

Note:

Before you begin, make sure that you have the latest ADM NITRO SDK and that the client application satisfies the prerequisites for using the ADM NITRO SDK.

All NITRO exceptions are captured by the *com.citrix.mas.nitro.exception.nitro_exception* class. For a more detailed description, see Exception Handling.

The executable code for the sample is available in the <NITRO_SDK_HOME>/sample/ directory.

Using Java API to Create your First NITRO Application

1. Copy the libraries from <NITRO_SDK_HOME>/lib folder to the project classpath.
2. Create a new class and name it **MyFirstNitroApplication**.
3. Create an instance of com.citrix.mas.nitro.service.nitro_service class. This instance is used to perform all operations on the ADM appliance:

   ```
   nitro_service client = new nitro_service("10.102.29.170","HTTP");
   ```

   This code establishes a connection with an appliance that has IP address 10.102.29.170 and uses the HTTP protocol. Replace 10.102.29.170 with the IP address of the ADM appliance that you have access to. Use “http://netscalermas.cloud.com” for connecting to the ADM Service.

   To use HTTPS connection for incorporating wire level security, use HTTPS protocol by passing “https” instead of “http” to constructor of nitro_service as shown below:

   ```
   nitro_service  client  =  new  nitro_service("10.102.29.170",  "https");
   ```

   SDK takes care of SSL Certificates involved in SSL Handshake.
   Note: MPS currently presents a test certificate not issued by any signing authority, so SDK bypasses the credentials check by using empty Trust Manager

4. Use the nitro_service instance to log in to the appliance using your credentials:

   ```
   client.set_credentials("admin","verysecret");
   #client.set_isCloud(true); #uncomment this line for ADM Service
   #In case of ADM Service, the credentials will be treated as ID and Secret.
   client.login();
   ```

   This code logs into the appliance, with user name as admin and password as verysecret. Replace the credentials with your login credentials.

5. Create an instance of the com.citrix.mas.nitro.resource.config.mps.mpsuser class. You will use this instance to perform operations on the mpsuser. mpsuser my_mpsuser = new mpsuser();

6. Use the mpsuser instance to create a new mpsuser resource on the appliance:

```
my_mpsuser.set_name("nsroot");

my_mpsuser.set_password("mynsroot");

String[] user_groups = new String[1];

user_groups[0] = "owner";

my_mpsuser.set_groups(user_groups);
mpsuser.add(client, my_mpsuser);
```

This code first sets the attributes (name, password, user groups of the user) of the mpsuser locally and then creates the resource by using the corresponding add() method.

7. Retrieve the details of the mpsuser you have created:

```
mpsuser new_mpsuser = mpsuser.get(client, my_mpsuser);
System.out.println("Name : " +new_mpsuser.get_name());
```

This code first retrieves the details of the mpsuser resource as identified by my_mpsuser as an object from the ADM, extracts the required attributes (name) from the object, and displays the results.

8. Delete the mpsuser you created in the above steps:

```
mpsuser.delete(client, my_mpsuser);
```

Deletes the mpsuser resource as identified by my_mpsuser on the ADM appliance.

9. Log out of the ADM appliance using client.logout()

Using .NET API to Create your First NITRO Application

1. Copy the libraries from <NITRO_SDK_HOME>/lib folder to the project classpath.
2. Create a new class and name it **MyFirstNitroApplication**.
3. Create an instance of com.citrix.mas.nitro.service.nitro_service class. This instance is used to perform all operations on the appliance:

```
nitro_service client = new nitro_service("10.102.29.170", "http");
```

This code establishes a connection with an appliance that has IP address 10.102.29.170 and uses the HTTP protocol Replace 10.102.29.170 with the IP address of the ADM appliance that you have access to. Use “http://netscalermas.cloud.com” for connecting to the ADM Service.

To use HTTPS connection for incorporating wire level security, use HTTPS protocol by passing “https” instead of “http” to constructor of nitro_service as shown below:

```
nitro_service client = new nitro_service("10.102.29.170", "https");
```

SDK takes care of SSL Certificates involved in SSL Handshake.
Note: MPS currently presents a test certificate not issued by any signing authority, so SDK bypasses the credentials check by using empty Trust Manager

4. Use the nitro_service instance to log in to the appliance using your credentials:

```
client.set_credentials("admin","verysecret");

 #client.isCloud = true; #uncomment this line for ADM Service

 #In case of ADM Service, the credentials will be treated as ID and Secret.

Client.login();
```

This code logs into the appliance, with user name as admin and password as verysecret. Replace the credentials with your login credentials.

5. Create an instance of the com.citrix.mas.nitro.resource.config.mps.mpsuser class. You will use this instance to perform operations on the mpsuser: mpsuser my_mpsuser = new mpsuser();

6. Use the mpsuser instance to create a new mpsuser:

   ```
   my_mpsuser.name = "nsroot";

    my_mpsuser.password = "mynsroot";

    String[] user_groups = {"owner"};

    my_mpsuser.groups = user_groups;

    mpsuser.add(client, my_mpsuser);
   ```

   This code first sets the attributes (name, password, user groups of the user) of the mpsuser locally and then creates the resource by using the corresponding add() method.

7. Retrieve the details of the mpsuser you have created:

   ```
   mpsuser new_mpsuser = mpsuser.get(client, my_mpsuser);
   System.out.println("Name : " +new_mpsuser.name);
   ```

   This code first retrieves the details of the mpsuser resource as identified by my_mpsuser as an object from the ADM, extracts the required attributes (name) from the object, and displays the results.

8. Delete the mpsuser you created in the above steps:

   ```
   mpsuser.delete(client, my_mpsuser);
   ```

9. Log out of the ADM appliance using client.logout()

Connecting to the ADM Appliance

The first step towards using NITRO is to establish a session with the ADM appliance and then authenticate the session by using the ADM user's credentials.

You must create an object of the *com.citrix.mas.nitro.service.nitro_service* class by specifying the ADM IP address and the protocol to connect to the appliance (HTTP or HTTPS). You then use this object and log on to the appliance by specifying the user name and the password of the ADM administrator.

Note:

- For the python SDK, the package path is of the form *massrc.com.citrix.mas...*
- You must have a user account on that appliance. The configuration operations that you perform are limited by the administrative roles assigned to your account.

The following sample code establishes a session with the ADM appliance with IP address 10.102.29.60 by using the HTTPS protocol and also sets a session timeout period (in seconds) of 60 minutes.

Java - Sample code to establish session

```
//Specify the ADM appliance IP address and protocol
nitro_service client = new nitro_service("10.102.29.60","https");

//Specify the login credentials
client.set_credentials("admin","verysecret",360);
#client.set_isCloud(true); #uncomment this line for ADM Service

#In case of ADM Service, the credentials will be treated as ID and Secret.

Client.login();
```

.NET - Sample code to establish session

```
//Specify the ADM appliance IP address and protocol
nitro_service client = new nitro_service("10.102.29.60","https");

//Specify the login credentials
client.set_credentials("admin","verysecret",360);
#client.isCloud(true); #uncomment this line for ADM Service

#In case of ADM Service, the credentials will be treated as ID and Secret.

Client.login();
```

Python - Sample code to establish session

```
#Specify the ADM appliance IP address and protocol client =
nitro_service("10.102.29.60","https")

#Specify the login credentials
client.set_credentials("admin","verysecret",360)
#client.isCloud(true) #uncomment this line for ADM Service

#In case of ADM Service, the credentials will be treated as ID and Secret.

Client.login()
```

General API Usage

ADM resources are organized into a set of packages or namespaces. Each package or namespace corresponds to a ADM feature. For example, all ADM resources are available in *com.citrix.mas.nitro.resource.config.ns*.

Note: For the python SDK, the package path is of the form *massrc.com.citrix.mas......*

Each ADM resource is represented by a class. For example, the class that represents a user is called **mpsuser** (in *com.citrix.mas.nitro.resource.config.mps*). The state of a resource is represented by properties of a class. You can get and set the properties of the class.

Note: The setter and getter properties are always executed locally on the client. They do not involve any network interaction with the NITRO web service. All properties have basic simple types: integer, long, boolean, and string.

Examples

Adding an ADM Resource

To create a new resource, instantiate the resource class, configure the resource by setting its properties locally, and then upload the new resource instance to the ADM appliance.

The following sample code creates a mpsuser resource

Java - Sample code to add an ADM resource

```
//Create an instance of the mpsuser
mpsuser my_mpsuser = new mpsuser();

my_mpsuser.set_name("nsroot");
my_mpsuser.set_password("mynsroot");
String[] user_groups = new String[1];
user_groups[0] = "owner";
my_mpsuser.set_groups(user_groups);


//Upload the resource to ADM
mpsuser.add(client, my_mpsuser);
```

.NET - Sample code to add an ADM resource

```
//Create an instance of the mpsuser
mpsuser my_mpsuser = new mpsuser();
my_mpsuser.name = "nsroot";
my_mpsuser.password = "mynsroot");
String[] user_groups = {"owner"};
my_mpsuser.groups = user_groups;

//Upload the resource to ADM
mpsuser.add(client, my_mpsuser);
```

Python - Sample code to add an ADM resource

```
#Create an instance of the mpsuser
my_mpsuser = mpsuser()
my_mpsuser.name = "nsroot"
my_mpsuser.password = "mynsroot"
user_groups = []
user_groups.append("owner")
my_mpsuser.groups = user_groups

#Upload the resource to ADM
mpsuser.add(client, my_mpsuser)
```

Retrieving Properties of ADM Resources

To retrieve the properties of a resource, you retrieve the resource object from the ADM appliance. Once the object is retrieved, you can extract the required properties of the resource locally, without further network traffic.

The following sample code retrieves the details of a mpsuser resource.

Java - Sample code to get details of resource

```
//Retrieve the id of the resource object from the ADM mpsuser
my_mpsuser = new mpsuser();
mpsuser[] simplelist = mpsuser.get(client);
for(mpsuser item : simplelist ){
if (item.get_name().equals("nsroot")) {
        my_mpsuser.set_id(item.get_id());
}
 }
 my_mpsuser.set_name("nsroot");

//Retreive the resource object identified by the id
my_mpsuser_retreived = mpsuser.get(client,my_mpsuser);

//Extract the properties of the resource from the object //locally
System.out.println(my_mpsuser_retreived.get_name());
System.out.println(my_mpsuser_retreived.get_id());
```

.NET - Sample code to get details of resource

```
//Retrieve the id of the resource object from the ADM mpsuser
my_mpsuser = new mpsuser();
mpsuser[] simplelist = mpsuser.get(client);

for(int i=0; i< simplelist.Length; i++){
        if (simplelist[i].name.Equals("nsroot")){
                my_mpsuser.id = simplelist[i].id;
    }
}
my_mpsuser.name = "nsroot";

//Retreive the resource object identified by the id
my_mpsuser_retreived = mpsuser.get(client,my_mpsuser);

//Extract the properties of the resource from the object //locally
Console.WriteLine(my_mpsuser_retreived.name);
Console.WriteLine(my_mpsuser_retreived.id);
```

Python - Sample code to get details of resource

```
#Retrieve the id of the resource object from the ADM
my_mpsuser = mpsuser()
simplelist = mpsuser.get(client)
for item in simplelist :
if item.name == "nsroot" :
        mympsuser.id = item.id;
 mympsuser.name = "nsroot"

#Retreive the resource object identified by the id
my_mpsuser_retreived = mpsuser.get(client,mympsuser);

#Extract the properties of the resource from the object //locally
print(my_mpsuser_retreived.name)
print(my_mpsuser_retreived.id)
```

Filtering Results

You can also retrieve resources by specifying a filter on the value of their properties by using the *com.citrix.mas. nitro.util.filtervalue* class or calling get_filtered() method defined for the resource. For example, you can retrieve all mpsuser resources whose tenant_name or tenant_id is same and belonging to a same user group.

Java - Sample code to get filtered results

```
filtervalue[] filter = new filtervalue[2]; filter[0] = new
filtervalue("tenant_name","Owner");
filter[1] = new filtervalue("groups","owner");
mpsuser [] result =mpsuser.get_filtered(client,filter);
```

.NET - Sample code to get filtered results

```
filtervalue[] filter = new filtervalue[2]; filter[0] = new
filtervalue("tenant_name","Owner");
filter[1] = new filtervalue("groups","owner”);
mpsuser [] result =mpsuser.get_filtered(client,filter);
```

Python - Sample code to get filtered results

```
filter_params = []
filter_params = [ filtervalue() for _ in range(2)]

filter_params[0] = filtervalue("groups","owner") filter_params[1] =
filtervalue("tenant_name","Owner")
result =mpsuser.get_filtered(client, filter_params)
```

Updating an ADM Resource

To update the properties of a resource, instantiate the resource class, specify the id of the resource to be updated, configure the resource by updating its properties locally, and then upload the updated resource instance to the ADM appliance.

Note: Some properties in some ADM resources are not allowed to be modified after creation. The location or tenant_id of the managed_device resource, are examples of such properties. Even though the update method appears to succeed, these properties retain their original values on the appliance.

The following sample code updates the password of a mpsuser resource

Java - Sample code to update an ADM resource

```
mpuser update_mpsuser = new mpuser();

//Specify the id of the mpuser to be updated
update_mpsuser.set_id("9b304030-4233-45df-be3b-0d73d3cd59a9 ");

//Specify the updated password
update_mpsuser.set_password("verysecret");

//Upload the resource to ADM
mpuser.update(client,update_mpsuser);
```

.NET - Sample code to update an ADM resource

```
mpuser update_mpsuser = new mpuser();

//Specify the id of the mpuser to be updated
update_mpsuser.id = "9b304030-4233-45df-be3b-0d73d3cd59a9 ";

//Specify the updated password
update_mpsuser.password = "verysecret";

//Upload the resource to ADM
mpuser.update(client,update_mpsuser);
```

Python - Sample code to update an ADM resource

```
update_mpsuser = mpuser()

#Specify the id of the mpuser to be updated
update_mpsuser.id = "9b304030-4233-45df-be3b-0d73d3cd59a9 "

#Specify the updated password
update_mpsuser.password = "verysecret"

#Upload the resource to ADM
mpuser.update(client,update_mpsuser)
```

Deleting an ADM Resource

To delete an existing resource, invoke the static method **delete ()** on the resource class, by passing the id of the resource.

The following sample code deletes mpsuser resource whose id is 9b304030-4233-45df-be3b-0d73d3cd59a9

Java - Sample code to delete an ADM resource

```
mpsuser remove_mpsuser = new mpsuser();
remove_mpsuser.set_id("9b304030-4233-45df-be3b-0d73d3cd59a9 ");
mpsuser.delete(client, remove_mpsuser);
```

.NET - Sample code to delete an ADM resource

```
mpsuser remove_mpsuser = new mpsuser();
remove_mpsuser.id = "9b304030-4233-45df-be3b-0d73d3cd59a9 ";
mpsuser.delete(client, remove_mpsuser);
```

Python - Sample code to delete an ADM resource

```
remove_mpsuser = mpsuser()
remove_mpsuser.id = "9b304030-4233-45df-be3b-0d73d3cd59a9 "
mpsuser.delete(client, remove_mpsuser)
```

Usage Scenarios

In this section, we provide NITRO API specific to certain resources and scenarios. We will be adding more scenarios in future updates to this documentation.

Adding a managed device to ADM

To add a managed device, first create an instance of managed_device class and set the ip address for the device being added. This must be set as it is a mandatory property. You can also set other optional properties by using the corresponding setter methods. Then invoke the static add_device method on the managed_device class by passing this instance.

Java – Sample code to add a managed_device

```
managed_device managed_device_obj = new managed_device();
managed_device_obj.set_ip_address ("10.106.101.17");
managed_device_obj.set_profile_name ( "ns_nsroot_profile");
managed_device.add_device(adm_client, managed_device_obj);
```

Csharp – Sample code to add a managed_device

```
managed_device managed_device_obj = new managed_device();
managed_device_obj.ip_address = "10.106.101.17";
managed_device_obj.profile_name = "ns_nsroot_profile";
managed_device.add_device(adm_client, managed_device_obj);
```

Python – Sample code to add a managed_device

```
managed_device_obj = managed_device()
managed_device_obj.ip_address = "10.106.101.17"
managed_device_obj.profile_name = "ns_nsroot_profile"
managed_device.add_device(adm_client, managed_device_obj)
```

Retrieving list of all managed devices from ADM

To get the list of all managed devices from ADM, just invoke the static get_filtered method by passing an instance of nitro_service and empty filter value.

Java – Sample code to retrieve list of all managed_devices

```
managed_device[] simplelist = managed_device.get_filtered(adm_client, "");
```

Csharp – Sample code to retrieve list of all managed_devices

```
managed_device[] simplelist = managed_device.get_filtered(adm_client, "");
```

Python – Sample code to retrieve list of all managed_devices

```
simplelist = managed_device.get_filtered(adm_client, "")
```

Retrieve details of a managed device from ADM

To get the details of managed device from ADM, invoke static get_filtered method by passing the ip address of the device.

Java – Sample code to retrieve details of a managed_device

```
String _filter = “ip address: 10.106.101.112“;
managed_device retrieved_device = managed_device.get_filtered(adm_client, _filter);
```

Csharp – Sample code to retrieve details of a managed_device

```
String _filter = “ip address: 10.106.101.112“;
managed_device retrieved_device = managed_device.get_filtered(adm_client,_filter);
```

Python – Sample code to retrieve details of a managed_device

```
_filter = “ip address: 10.106.101.112“
retrieved_device = managed_device.get_filtered(adm_client, _filter)
```

Deleting a managed device from ADM

To delete a managed device from ADM, create an instance of managed_device class and set the device id of the device you want to delete and invoke the static delete method by passing this instance.

Java – Sample code to delete a managed_device

```
Managed_device device_obj = managed_device();
device_obj.set_id (Object.toString(device_id));
managed_device.delete(adm_client,device_obj);
```

Csharp – Sample code to delete a managed_device

```
Managed_device device_obj = managed_device();
device_obj.id = Convert.ToString(device_id);
managed_device.delete(adm_client,device_obj);
```

Python – Sample code to delete a managed_device

```
device_obj = managed_device()
device_obj.id = str(device_id)
managed_device.delete(adm_client,device_obj)
```

where device_id is the id of the device you want to delete. To know about the specific device id, get the details of the device.

Uploading a SSL Certificate to ADM

To upload a SSL certificate, first create an instance of ns_ssl_cert class and set the file name and path to the SSL file. Then invoke the static upload method by passing the created instance.

Java – Sample code to upload a SSL certificate

```
ns_ssl_cert certificate_obj = new ns_ssl_cert();
certificate_obj.set_file_location_path("/root/random_scripts/");
certificate_obj.set_file_name ("mydomain29282.com.pem");
ns_ssl_cert.upload(adm_client, certificate_obj);
```

Csharp – Sample code to upload a SSL certificate

```
ns_ssl_cert certificate_obj = new ns_ssl_cert();
certificate_obj.file_location_path = "/root/random_scripts/";
certificate_obj.file_name = "mydomain29282.com.pem";
ns_ssl_cert.upload(adm_client, certificate_obj);
```

Python – Sample code to upload a SSL certificate

```
certificate_obj = ns_ssl_cert()
certificate_obj.file_location_path = "/root/random_scripts/"
certificate_obj.file_name = "mydomain29282.com.pem"
ns_ssl_cert.upload(adm_client, certificate_obj)
```

Replace the file name and file path to appropriate values.

Downloading the SSL certificate from ADM

To download a ssl certificate, first create an instance of ns_ssl_cert class and set the file name and path to the ssl file. Then invoke the static download method by passing the created instance.

Java – Sample code to download a SSL certificate

```
ns_ssl_cert certificate_obj = new ns_ssl_cert();
certificate_obj.set_file_location_path("/root/random_scripts/");
certificate_obj.set_file_name ("mydomain29282.com.pem");
ns_ssl_cert retrieved_certificate_obj = ns_ssl_cert.download (adm_client, certificate_obj);
```

Csharp – Sample code to download a SSL certificate

```
ns_ssl_cert certificate_obj = new ns_ssl_cert();
certificate_obj.file_location_path = "/root/random_scripts/";
certificate_obj.file_name = "mydomain29282.com.pem";
ns_ssl_cert retrieved_certificate_obj = ns_ssl_cert.download(adm_client, certificate_obj);
```

Python – Sample code to download a SSL certificate

```
certificate_obj = ns_ssl_cert()
certificate_obj.file_location_path = "/root/random_scripts/"
certificate_obj.file_name = "mydomain29282.com.pem"
retrieved_certificate_obj = ns_ssl_cert.download(adm_client, certificate_obj)
```

Replace the file name and file path to appropriate values.

Deleting the SSL certificate from ADM

To delete a SSL certificate, first create an instance of ns_ssl_cert class and set the file name and path to the SSL file. Then invoke the static delete method by passing the created instance.

Java – Sample code to delete a SSL certificate

```
ns_ssl_cert certificate_obj = new ns_ssl_cert();
certificate_obj.set_file_location_path("/root/random_scripts/");
certificate_obj.set_file_name ("mydomain29282.com.pem");
ns_ssl_cert.delete (adm_client, certificate_obj);
```

Csharp – Sample code to delete a SSL certificate

```
ns_ssl_cert certificate_obj = new ns_ssl_cert();
certificate_obj.file_location_path = "/root/random_scripts/";
certificate_obj.file_name = "mydomain29282.com.pem";
ns_ssl_cert.delete(adm_client, certificate_obj);
```

Python – Sample code to delete a SSL certificate

```
certificate_obj = ns_ssl_cert()
certificate_obj.file_location_path = "/root/random_scripts/"
certificate_obj.file_name = "mydomain29282.com.pem"
ns_ssl_cert.delete(adm_client, certificate_obj)
```

Retrieve details of SSL certificate from ADM

To retrieve details of a SSL certificate, invoke the static get_filtered method by passing the filename of certificate.

Java – Sample code to retrieve details of a SSL certificate

```
String json_filter = "file_name: mydomain29282.com.pem";
retrieved_cert = ns_ssl_cert.get_filtered(adm_client, json_filter);
```

Csharp – Sample code to retrieve details of a SSL certificate

```
String json_filter = "file_name: mydomain29282.com.pem";
retrieved_cert = ns_ssl_cert.get_filtered(adm_client, json_filter);
```

Python – Sample code to retrieve details of a SSL certificate

```
json_filter = "file_name: mydomain29282.com.pem"
retrieved_cert = ns_ssl_cert.get_filtered(adm_client, json_filter)
```

Retrieve list of all SSL certificates from ADM

To retrieve list of all SSL certificates, invoke the static get_filtered method by passing empty filter value.

Java – Sample code to retrieve list of all SSL certificates

```
        retrieved_cert = ns_ssl_cert.get_filtered(adm_client, "");
```

Csharp – Sample code to retrieve list of all SSL certificates

```
        retrieved_cert = ns_ssl_cert.get_filtered(adm_client, "");
```

Python – Sample code to retrieve list of all SSL certificates

```
        retrieved_cert = ns_ssl_cert.get_filtered(adm_client, "")
```

Performing inventory check from ADM

This involves rediscovering that device that was added to the ADM. To do this, create an instance of inventory class and set device IP address and the name of the partition in the device. Then invoke static get method by passing this instance.

Java – Sample code to retrieve activity id of rediscovery process

```
        inventory inventory_obj = new inventory();
        String device_ip = device_ip + "-" + partition;
        inventory_obj.set_device_ipaddress(device_ip);
        inventory[] simplelist = inventory.get(adm_client, inventory_obj);
        String act_id = simplelist[simplelist.length-1].act_id;
```

Csharp – Sample code to retrieve activity id of rediscovery process

```
        inventory inventory_obj = new inventory();
        String device_ip = device_ip + "-" + partition;
        inventory_obj.device_ipaddress = device_ip;
        inventory[] simplelist = inventory.get(adm_client, inventory_obj);
        String act_id =  simplelist.Last().act_id;
```

Python – Sample code to retrieve activity id of rediscovery process

```
        inventory_obj = inventory()
        device_ip = device_ip + "-" + partition
        inventory_obj.device_ipaddress = device_ip
        simplelist = inventory.get(adm_client, inventory_obj)
        act_id = simplelist[-1].act_id
```

Finally, activity ID of the rediscovery process is fetched.

Retrieve list of all events from ADM

To retrieve list of all events, invoke get method of the event class. To get the count of no of events, create an instance of event class and set the count property to yes and pass it as argument to the static get method.

Java – Sample code to retrieve list of all events and its count

```
event[] simplelist = event.get(adm_client);

event event_obj =new event();
event_obj.set_count(yes);
int event_count = event.get(adm_client,event_obj);
```

Csharp – Sample code to retrieve list of all events and its count

```
event[] simplelist = event.get(adm_client);

event event_obj =new event();
event_obj.count = yes;
int event_count = event.get(adm_client,event_obj);
```

Python – Sample code to retrieve list of all events and its count

```
simplelist = event.get(adm_client)

event_obj = event()
event_obj.count = yes
event_count = event.get(adm_client,event_obj)
```

Delete an event from ADM

To delete an event from ADM, create an instance of event class and set the id of the event and invoke static delete method by passing this instance.

Java – Sample code to delete an event

```
event event_obj = new event();
event_obj.set_id (event_id);
event.delete(adm_client, event_obj);
```

Csharp – Sample code to delete an event

```
event event_obj = new event();
event_obj.id = event_id;
event.delete(adm_client, event_obj);
```

Python – Sample code to delete an event

```
event_obj = event()
event_obj.id = event_id
event.delete(adm_client, event_obj)
```

Creating a configuration job on ADM

To create a config job, first create an instance of config_job class, then set all the mandatory and relevant properties and invoke the “add” static method by passing this instance to the method.

Here the mandatory parameter is name of the config job but to demonstrate how a job can be created, other parameters are set.

Java – Sample code to create a config job

```
String[] device_list=new String[10];
device_list[0] = '10.102.201.86-p1';
config_command config_commandObj= new config_command();
config_commandObj.set_protocol('SSH');
config_commandObj.set_command('show ns config');
String command_list= new String[10];
command_list[0] = config_commandObj;
configuration_template config_temp= new configuration_template();
config_temp.set_commands(command_list);
config_job config_job_obj= new config_job();
config_job_obj.set_name('test2');
config_job_obj.set_devices(device_list);
config_job_obj.set_device_family('ns');
config_job_obj.set_template_info(config_temp);
config_job[] simplelist = config_job.add(client, config_job_obj);
```

Csharp – Sample code to create a config job

```
string[] device_list=new string[10];
device_list[0] = '10.102.201.86-p1';
config_command config_commandObj= new config_command();
config_commandObj.protocol='SSH';
config_commandObj.command='show ns config';
string command_list= new string[10];
command_list[0] = config_commandObj;
configuration_template config_temp= new configuration_template();
config_temp.commands=command_list;
config_job config_job_obj= new config_job();
```

```
        config_job_obj.name='test2';
        config_job_obj.devices=device_list;
        config_job_obj.device_family='ns';
        config_job_obj.template_info=config_temp;
        config_job[] simplelist = config_job.add(client, config_job_obj);
```

Python – Sample code to create a config job

```
        device_list=[]
        device_list.append('10.102.201.86-p1')
        config_commandObj=config_command()
        config_commandObj.protocol='SSH'
        config_commandObj.command='show ns config'
        command_list=[]
        command_list.append(config_commandObj)
        config_temp=configuration_template()
        config_temp.commands=command_list
        config_job_obj= config_job()
        config_job_obj.name='test2'
        config_job_obj.devices=device_list
        config_job_obj.device_family='ns'
        config_job_obj.template_info=config_temp
        simplelist = config_job.add(client, config_job_obj)
```

Adding datacenter through ADM

To add a new datacenter, create an instance of mps_datacenter class and set latitude, longitude and name of the datacenter and invoke “add” static method by passing this instance.

Mandatory parameters are latitude, longitude and name.

Java – Sample code to add a datacenter

```
mps_datacenter my_datacenter = new mps_datacenter();

my_datacenter.set_name("new_dc");

my_datacenter.set_latitude(12.9716);

my_datacenter.set_longitude(77.5946);

mps_datacenter simplelist = mps_datacenter.add(client,my_datacenter);
```

Csharp – Sample code to add a datacenter

```
mps_datacenter my_datacenter = new mps_datacenter();

my_datacenter.name = "new_dc";

my_datacenter.latitude = 12.9716;

my_datacenter.longitude = 77.5946;

mps_datacenter simplelist = mps_datacenter.add(client,my_datacenter);
```

Python – Sample code to add a datacenter

```
my_datacenter = mps_datacenter()

my_datacenter.name = "new_dc"

my_datacenter.latitude = float(12.9716)

my_datacenter.longitude = float(77.5946)

simplelist = mps_datacenter.add(client,my_datacenter)
```

Delete a datacenter from ADM

To delete a datacenter from ADM, create an instance of mps_datacenter class and set the id of the datacenter and invoke static delete method by passing this instance.

Here id of the datacenter named “new_dc” is fetched and then the datacenter is deleted using this id.

Java – Sample code to delete a datacenter

```
mps_datacenter my_datacenter = new mps_datacenter();
String filter_value = "name:new_dc";
mps_datacenter simplelist = mps_datacenter.get_filtered(client,filter_value);
mps_datacenter my_datacenter = new mps_datacenter();
my_datacenter.set_id(simplelist[0].id);
mps_datacenter simplelist = mps_datacenter.delete(client,my_datacenter);
```

Csharp – Sample code to delete a datacenter

```
mps_datacenter my_datacenter = new mps_datacenter();
string filter_value = "name:new_dc";
mps_datacenter simplelist = mps_datacenter.get_filtered(client,filter_value);
mps_datacenter my_datacenter = new mps_datacenter();
my_datacenter.id= simplelist[0].id;
mps_datacenter simplelist = mps_datacenter.delete(client,my_datacenter);
```

Python – Sample code to delete a datacenter

```
my_datacenter = mps_datacenter()
filter_value = "name:new_dc"
simplelist = mps_datacenter.get_filtered(client,filter_value)
my_datacenter = mps_datacenter()
my_datacenter.id= str(simplelist[0].id)
simplelist = mps_datacenter.delete(client,my_datacenter)
```

Retrieving ADC SDX instances from ADM

To retrieve ADC SDX instances from ADM, invoke static “get” method.

Java – Sample code to retrieve ADC SDX instances

```
nssdx simplelist = nssdx.get(client);
for(nssdx item : simplelist){
        print("name of the ADC SDX: " + item.get_name());
        print("id of the ADC SDX: " + item.get_id());
}
```

Csharp – Sample code to retrieve ADC SDX instances

```
nssdx simplelist = nssdx.get(client);
foreach (nssdx item in simplelist){
        print("name of the ADC SDX: " + item.name);
        print("id of the ADC SDX: " + item.id);
}
```

Python – Sample code to retrieve ADC SDX instances

```
simplelist = nssdx.get(client)
for item in simplelist:
        print("name of the ADC SDX: " + item.name)
        print("id of the ADC SDX: " + item.id)
```

Retrieving details of stylebooks from ADM

To retrieve details of stylebooks from ADM, invoke static “get” method.

Java – Sample code to retrieve details of stylebooks

```
stylebooks simplelist = stylebooks.get(client) ;
for(stylebooks item : simplelist){
print("Name of the stylebook: "+ item.get_display_name());
print("Source of the stylebook: "+ item.get_source());
}
```

Csharp – Sample code to retrieve details of stylebooks

```
stylebooks simplelist = stylebooks.get(client) ;
foreach(stylebooks item in simplelist){
print("Name of the stylebook: "+ item.display_name);
print("Source of the stylebook: "+ item.source);
}
```

Python – Sample code to retrieve Details of stylebooks

```
simplelist = stylebooks.get(client)
for item in simplelist:
        print("Name of the stylebook: "+ item.display_name)
        print("Source of the stylebook: "+ str(item.source
```