ibm.com/redbooks

Exploring the DS8870 RESTful API Implementation

Bert Dufrasne

Adrian Orben

Bob Xiao

Redpaper

Bert Dufrasne
Adrian Orben
Bob Xiao

Exploring the DS8870 RESTful API Implementation

Representational State Transfer (REST) is a stateless architectural style and development approach that uses existing web technology and protocols, essentially Hypertext Transfer Protocol (HTTP).

In the last few years, REST has emerged as a predominant web service design model. It is a simpler alternative to SOAP. The RESTful application programming interface (API) establishes a mapping between create, read, update, and delete operations and the corresponding HTTP actions: POST, GET, PUT, and DELETE.

The RESTful interface focuses on the roles and resources of components, and ignores their internal implementation details. The RESTful API is a client/server model. Therefore, the requests are sent to a server that is component-aware, which masks the intricate details from the users. It also means all information required to process the request by the server is contained along with the request.

This IBM® Redpaper™ publication describes the concepts, architecture, and design of DS8870 RESTful API. It demonstrates how clients can initiate RESTful API communications with their DS8870 storage from a web-browser client, or by using the cURL command-line tool.

The scope of this paper is solely intended for readers who want to explore and experiment with RESTful capabilities implemented in the DS8870. This paper is also a reference for readers who want to understand the internal design points and architecture of DS8870 RESTful API services.

Introduction to DS8870 RESTful API

The DS8870 Release 7.5 provides RESTful API services. This support enables DS8870 improved cloud deployment while using an industry standard API. As we further explain and illustrate in this paper, the DS8870 RESTful API support enables automated custom DS8870 storage operations on volumes, pools, Fibre Channel ports, and also some copy services operations.

Figure 1 shows how RESTful applications or utilities communicate with the DS8870 RESTful services, which are implemented and running on the DS8870 Management Console. By being implemented on the Management Console, rather than in the DS8870 firmware, the DS8870 RESTful API becomes easier for users and developers to maintain and upgrade.

Tip: The IBM Storage Mobile Dashboard, now available for the DS8870 Release 7.5, uses the REST API for its communications with the DS8870.

The communication with the DS8870 Management Console is over Internet Protocol. In its current implementation, the DS8870 RESTful API supports light-weight Secure Hypertext Transfer Protocol (HTTPS) communications in Java Script Object Notation (JSON) format.

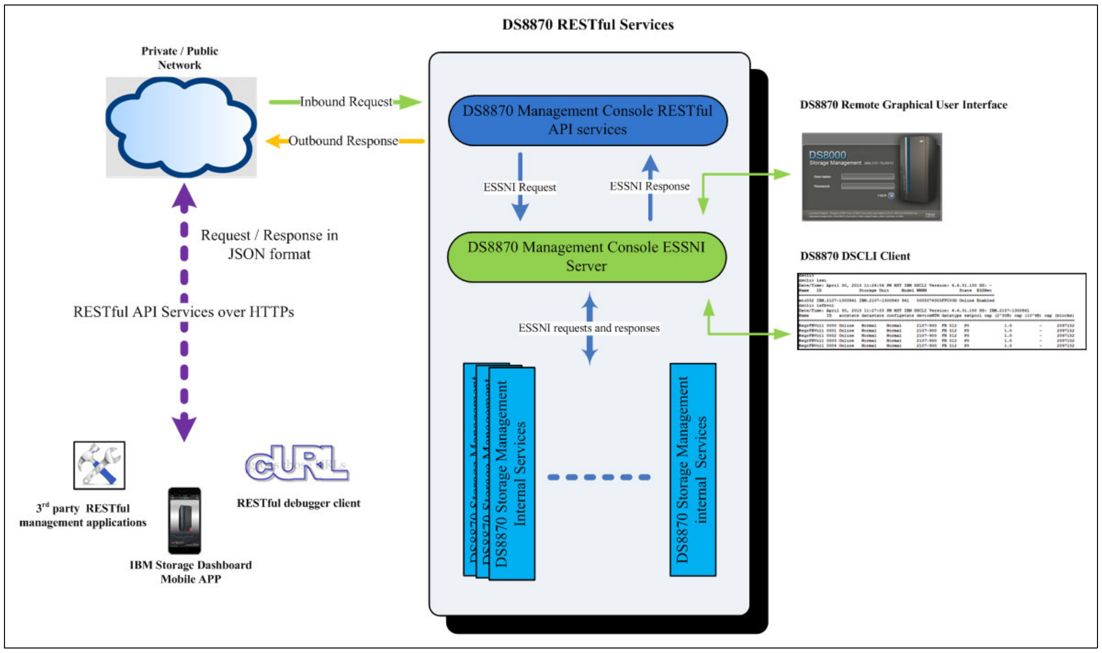

DS8870 RESTful Services
Private / Public Network
Inbound Request
Outbound Response
DS8870 Management Console RESTful API services
ESSNI Request
ESSNI Response
DS8870 Management Console ESSNI Server
ESSNI requests and responses
DS8870 Storage Management Internal Services
DS8870 Storage Management internal Services
Request / Response in JSON format
RESTful API Services over HTTPs
DS8870 Remote Graphical User Interface
DS8870 DSCLI Client
3rd party RESTful management applications
RESTful debugger client
IBM Storage Dashboard Mobile APP

Figure 1 Overview of DS8870 RESTful API Architecture

The DS8870 RESTful API services are enabled by default. There is no additional hardware or software that needs to be deployed to use the DS8870 RESTful API services.

The RESTful API in turn communicates with IBM DS8870 Enterprise Storage Server® Network Interface (ESSNI) server which is also running on the Management Console. ESSNI server provides the communication interfaces for RESTful services and other external clients including DS8870 GUI and DSCLI that can manage the DS8870 storage resources.

DS8870 RESTful API design points and architecture

The DS8870 RESTful API implementation complies with industry-standard REST architecture and design principles. This section first reviews the RESTful API design points and then has details of the RESTful architecture implementation for the DS8870.

- Standard client/server interface

  The standard client/server interface separates the implementation of both server and client. With the comprehensive and uniformed RESTful API interfaces, client applications can be built regardless of DS8870 internal software architecture.

- Stateless

  Being stateless means that the server (DS8870) has no memory of previous communication with any of its clients. This design shifts the responsibility of maintaining state from server system to client applications. The client is the only entity that can save its session states.

- Layered system

  In principle, the client should not be able to determine whether it is connected directly to the end server, or through an intermediate middleware. Therefore, servers at various layers can be used to improve system scalability, and can also be used to enforce security policies. This principle applies to both client and server.

- Code on demand (optional)

  This optional principle allows RESTful clients to request and execute code from a server. It not only improves extensibility and configuration for servers, but also results in improved performance and efficiency for clients.

RESTful operations

The RESTful API implements four basic program operations, which are create, read, update and delete (CRUD), that map to corresponding HTTP methods:

- HTTP POST is mapped to create operations. For instance, create a Fixed Block volume in the DS8870.
- HTTP GET is mapped to the query (read) operation. It is used to retrieve information from specified resources. Upon successful execution, resource attributes are returned to RESTful client.
- HTTP PUT is mapped to update operations, such as modification of DS8870 volume capacity.
- HTTP DELETE is mapped to delete operations.

In addition to these HTTP methods, HTTP Uniform Resource Identifier (URI) defines the resource names available to REST applications.

The DS8870 RESTful API service contains a set of easy-to-reference URIs for specific system resources. Be aware that the DS8870 RESTful version-1 (v1) supports only JSON format. Therefore, you must specify Content-Type of `application/json` in the HTTP header when an API request is initiated.

All currently supported URIs are defined according to the following generic format:

```
https://DS8870_HMC_hostname:8452/api/v1/resource_name
```

The format specifies the following information:

- `DS8870_HMC_hostname` specifies the host name or IP address of a DS8870 Management Console.
- `8452` is the HTTP port utilized by the DS8870 for RESTful API communication.
- `/api/v1` indicates the version-1 RESTful API
- `resource_name` designates the specific resource targeted by the operation.

Note: Resource name of URI in RESTful API is case sensitive.

The following URI example retrieves DS8870 general system information by the HTTP GET method:

```
https://IBM_DS8870.tuc.ibm.com:8452/api/v1/systems
```

Paging

Paging is useful for restricting the number of API records returned from the DS8870 system. It provides a better control with finer indexing to response data. Some queries, such as a volume list request, can potentially return a huge number of volume entities. A client application can traverse only a specific Storage Pools record by using paging.

Two parameters are used for paging control:

- Offset

  Defines the index of the first entity to be retrieved. The default offset is 0.
- Limit

  Defines the maximum number of elements the server should return.

An easy way to implement paging in your application is to embed the offset and limit parameters in your URI, as in the following example:

```
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/pools?limit=1&offset=1"
```

This URI tells the DS8870 system only to return the attributes of the second extent pool, if defined.

Response data to RESTful request

Returned data for a RESTful request is included, in JSON format, into the body of an HTTP response.

RESTful clients must first verify the response state from the HTTP header, that is, an HTTP status code that indicates whether the HTTP communication completed successfully. If the HTTP return status is good, clients can further parse the HTTP response for the details of the server response.

Response data structure

The actual response data is included in the body section of HTTP response to an HTTP request.

At a high level, response data includes the following three fields:

- server

  The server field provides a quick view of the API execution status, with a brief message. It consists of three subfields:

  - status

    This specifies the status of RESTful API request from DS8870. The returned value is either `ok` or `failed`.

  - code

    This is the abbreviation of `error code`. If the request response is a good status (the status field indicates `ok`), then the code field is empty. Otherwise, a specific error code is returned, and includes an error message for reference.

  - message

    This is a brief description in terms of RESTful API response.

- counts

  The counts field includes two subfields with reference to returned objects:

  - data_counts

    This subfield indicates the number of objects returned in the current response data.

  - total_counts

    This gives you the total counts of objects in the system (DS8870) that match the request. The data_counts might be less than the total_counts, if paging was used. For example, if you set a limit to 1 using paging, data_counts for storage pool show 1, indicating that only storage pool P1, for example, is returned. However, total_counts will show 2 if the DS8870 system actually has two storage pools configured.

- data

  The data field includes the detailed information for returned objects. You can find different object attributes such as volume capacity and name of the requested objects.

Data types

The response data can contain the following data types:

- Storage capacity is given as an integer in bytes for fixed block open systems such as Microsoft Windows and IBM AIX®. For count key data (CKD) system, capacity will be expressed in cylinders.
- Boolean attributes are returned as true or false.
- Time Interval is given in milliseconds. For instance, you can find the authenticated active session with maximum idle interval as 1,800,000 milliseconds (which is 30 minutes).
- Time is taken from the time of the DS8870 Master Console. By complying with ISO8601, it is expressed in the following format:

  ```
  yyyy-MM-dd’T’HH:mm:ssTZ by compliance to ISO8601
  ```

  TZ contains time zone information which is represented as offset of local time to UTC time. For instances, if the local time is US Mountain time, which is 7 hours later than UTC time, then TZ is represented as `-0700`.

For example, you can use the following RESTful API request to retrieve system attributes:

```
https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/systems
```

The system then returns with the data structure and information shown in Example 1:

- The `server` field shows that the request completed successfully (a status of `ok` is returned).
- The `data` field contains the detailed system information. It includes machine type and model, microcode bundle level, different types of storage capacities, machine name, code release, machine serial number and state also.

Example 1   Response to RESTful API GET request

```
{
    "counts": {
        "data_counts": 1,
        "total_counts": 1
    },
    "data": {
        "systems": [
            {
                "MTM": "2421-961",
                "bundle": "87.50.101.0",
                "cap": "169219563978752",
                "capalloc": "152822452584448",
                "capavail": "16394963910656",
                "capraw": "236357419008000",
                "name": "IBM_DS8870",
                "release": "7.5",
                "sn": "75LN031",
                "state": "online"
            }
        ]
    },
    "server": {
        "code": "",
        "message": "Operation done successfully.",
        "status": "ok"
    }
}
```

DS8870 RESTful API security

The DS8870 RESTful services implement a token-based authentication in combination with the role-based DS8870 ESSNI authorization. It means every RESTful client application must first authenticate through the ESSNI server with valid DS8870 user credentials (that is, a username and password). Upon successful authentication, a token is returned that must be used for further API requests, until the token expires.

Figure 2 depicts this high-level authentication and authorization process.

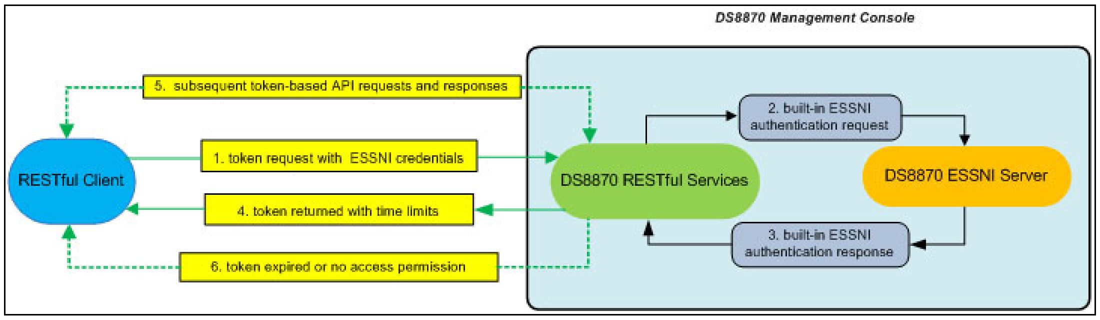

DS8870 Management Console
5. subsequent token-based API requests and responses
2. built-in ESSNI authentication request
1. token request with ESSNI credentials
RESTful Client
DS8870 RESTful Services
DS8870 ESSNI Server
4. token returned with time limits
3. built-in ESSNI authentication response
6. token expired or no access permission

Figure 2 DS8870 RESTful API token process flow

The process sequence is indicated by the numbered steps in the figure:

1. The RESTful client application must supply valid DS8870 credentials and request a token from the DS8870 RESTful services implemented at the Master Condole.
2. The RESTful service provides the supplied username and password to the ESSNI for validation.
3. The ESSNI returns the authentication status to the RESTful services.
4. If authentication is successful, a token and expiration time limits are returned to the client.
5. Any subsequent RESTful requests are passed along with the token.
6. If authentication is not successful or the token expired, the request is returned as unsuccessful.

Note: DS8870 ESSNI implements a role-based user authorization. To complete successfully, any RESTful API request against the system must be authorized for that role. For instance, a user with Monitor role will be denied any request to modify the D8870 logical configuration.

Using RESTful API through RESTclient and cURL

Several options are available so you can experiment with the DS8870 RESTful interfaces before you eventually build your own DS8870 RESTful applications. In this section, we illustrate the options of a browser-based graphical user interface (GUI) REST client and a command-line tool. These two options are both easy to install and use.

Web browser RESTful API debugger client - RESTclient

The RESTclient is a web browser extension, available for several web browsers, such as Mozilla Firefox and Google Chrome. To download and install the RESTclient with your browser, see the following website:

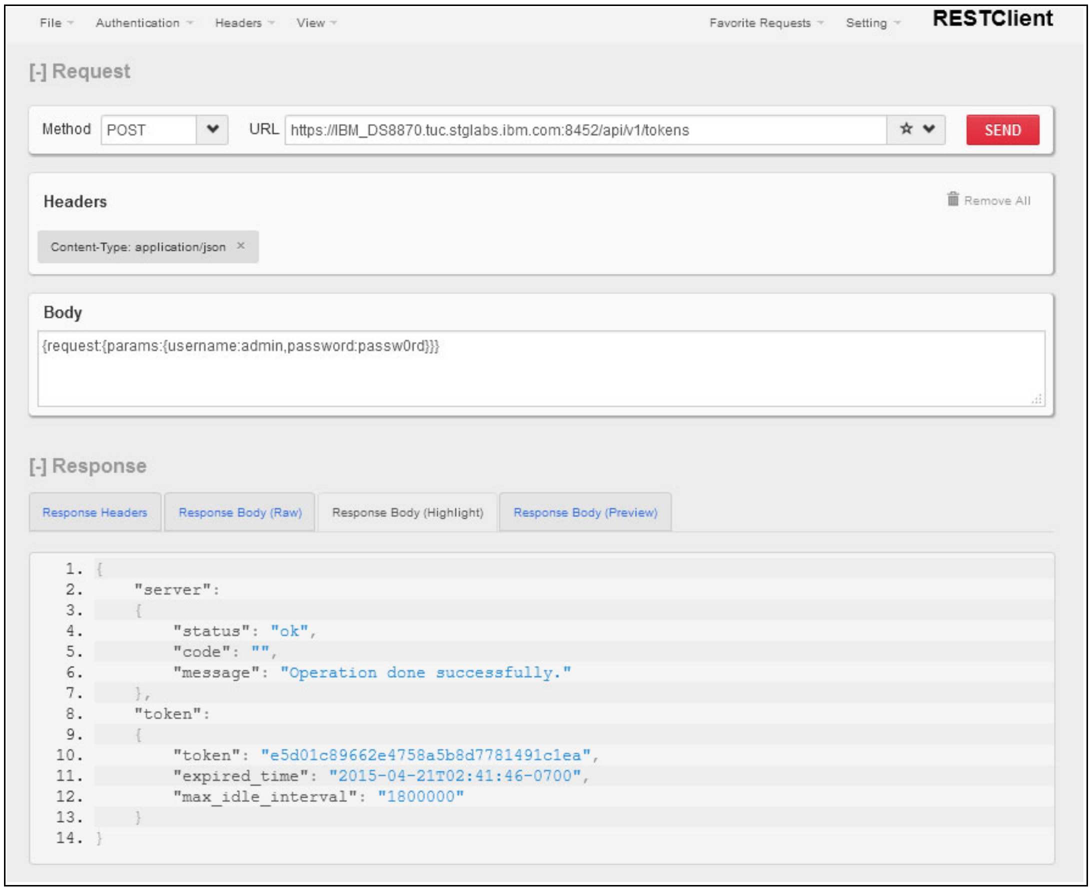
`http://restclient.net`

After you install the RESTClient plug-in, your browser offers an option to access the RESTClient GUI as shown in Figure 3.

As a first illustration, we show how to initially request the DS8870 security token. The token request must be issued using the HTTP **POST** method.

The Request section specifies the tokens RESTful API service request. It includes the HTTP Method (POST), URL of your DS8870 (with tokens requested), Headers (JSON format is specified), and Body (ESSNI user credentials). After completing these fields correctly, click **SEND** to request a token.

The DS8870 responds with the information visible in the Response section. The token ID is returned along with the expiration time and the `max_idle_interval`. Subsequent API requests must include that token.



Method: POST

URL: https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/tokens

Headers

Content-Type: application/json

Body

```
{request:{params:{username:admin,password:passw0rd}}}
```

```
{
    "server":
    {
        "status": "ok",
        "code": "",
        "message": "Operation done successfully."
    },
    "token":
    {
        "token": "e5d01c89662e4758a5b8d7781491c1ea",
        "expired_time": "2015-04-21T02:41:46-0700",
        "max_idle_interval": "1800000"
    }
}
```

Figure 3   RESTclient web-plugin quick view

Command-line RESTful API client: cURL

The cURL command-line software is no-cost and open-source tool for transferring data in URL syntax. It supports multiple operating systems including IBM AIX, IBM z/OS®, and most Linux distributions. Also, you can use cURL to experiment with DS8870 RESTful services.

Authentication with token apply

Again, we illustrate how to first request the security token. Example 2 shows the use of cURL for the `tokens` request.

Example 2 cURL token request and response

```
curl --tlsv1  -k -H "Content-Type: application/json" -X POST -d
'{request:{params:{username:admin,password:passw0rd}}}'
https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/tokens  |python -m json.tool
```

DS8870 supports Transport Layer Security (TLS) as the default communication protocol, which is specified by the **`--tlsv1`** option as follows:

- The `-k` option is used for DS8870-specific SSL certificate.
- The `-H` option specifies the content of header for HTTP communication. You must specify `"Content-Type: application/json"` because JSON is the only supported format for version-1 of the DS8870 RESTful API.
- The `-X` option specifies the HTTP method. POST method must be used for token request.
- The `-d` option specifies the data for the API request, to include the HTTP body. ESSNI username and password must be specified with the request.
- The URI (`https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/tokens`) addresses the DS88870 service (HMC IP address and port 8452), and token request.

Note: The built-in Python script `json.tool` is used for better formatting of response data.

Upon the successful completion, you get response data, similar to Example 3.

Example 3 Token request reply

```
{
    "server": {
        "code": "",
        "message": "Operation done successfully.",
        "status": "ok"
    },
    "token": {
        "expired_time": "2015-04-20T16:09:17-0700",
        "max_idle_interval": "1800000",
        "token": "f20ec4879d4e440192855e270bc37f6c"
    }
}
```

Create a Fixed Block volume

After you obtain a valid token, you can request other services, such as creating a Fixed Block volume as illustrated in Example 4. In this example, we create a 1 gibibyte (GiB) Fixed Block volume named as `REST_API_TEST` in storage pool P0 under logical subsystem (LLS) 0x00.

Example 4 How to create a DS8870 Fixed Block volume from cURL

```
[command_prompt]# curl --tlsv1 -k -H "Content-Type: application/json" -H
"X-Auth-Token:67ae680de0294e048aeefaae72b7a1e1" -X POST -d
'{request:{params:{name:REST_API_TEST, quantity:1, stgtype:fb, cap:1, captype:gib,
tp:none, pool:P0, lss:0}}}'
https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/volumes |python -m json.tool
{
    "data": {
        "volumes": [
            {
                "name": "REST_API_TEST"
            }
        ]
    },
    "link": {
        "href": "https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/volumes/0010",
        "rel": "self"
    },
    "server": {
        "code": "",
        "message": "Operation done successfully.",
        "status": "ok"
    }
}
```

The subfield `"href"` in the response data is the URL reference to the volume just created. It indicates volume ID is 0x0010 in hexadecimal format.

Update volume attributes

Example 5 demonstrates how to update the capacity of volume 0x0010 to 2 GiB.

Example 5 How to update DS8870 volume attributes from cURL

```
[command_prompt]# curl --tlsv1 -k -H "Content-Type: application/json" -H
"X-Auth-Token:67ae680de0294e048aeefaae72b7a1e1" -X PUT -d
'{request:{params:{cap:2, captype:gib}}}'
https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/volumes/0010 |python -m
json.tool
{
    "server": {
        "code": "",
        "message": "Operation done successfully.",
        "status": "ok"
    }
}
```

The capacity (cap) is specified within HTTP request body. The `captype` parameter specifies that gibibyte (GiB) is used to express the volume capacity.

Read volume attributes

Volume attributes can be retrieved by using HTTP GET method. Example 6 illustrates how to read the updated attributes of volume 0x0010, and shows the information returned.

Example 6 How to read DS8870 volume attributes from cURL

```
[command_prompt]# curl --tlsv1 -k -H "Content-Type: application/json" -H
"X-Auth-Token:67ae680de0294e048aeefaae72b7a1e1" -X GET
https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/volumes/0010 |python -m
json.tool
{
    "counts": {
        "data_counts": 1,
        "total_counts": 1
    },
    "data": {
        "volumes": [
            {
                "MTM": "2107-900",
                "VOLSER": "",
                "allocmethod": "rotateexts",
                "cap": "2147483648",
                "capalloc": "",
                "datatype": "FB 512",
                "easytier": "none",
                "id": "0010",
                "link": {
                    "href":
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/volumes/0010",
                    "rel": "self"
                },
                "lss": {
                    "id": "00",
                    "link": {
                        "href":
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/lss/00",
                        "rel": "self"
                    }
                },
                "name": "REST_API_TEST",
                "pool": {
                    "id": "P0",
                    "link": {
                        "href":
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/pools/P0",
                        "rel": "self"
                    }
                },
                "state": "normal",
                "stgtype": "FB",
                "tieralloc": [
                    {
                        "allocated": "1073741824",
                        "tier": "ENT"
                    }
                ],
                "tp": "none"
            }
        ]
    },
    "server": {
        "code": "",
        "message": "Operation done successfully.",
        "status": "ok"
    }
}
```

In addition to the updated capacity, you can see other volume attributes in the response data, including LSS, name, related storage pool, and so on.

Delete a volume

HTTP method DELETE is then invoked to delete volume 0x100 that was just created. See Example 7.

Example 7 How to delete a DS8870 volume from cURL

```
[command_prompt]# curl --tlsv1 -k -H "Content-Type: application/json" -H
"X-Auth-Token:21b7f544c278430981612694 4232f782" -X DELETE
https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/volumes/0010 |python -m
json.tool
{
    "server": {
        "code": "",
        "message": "Operation done successfully.",
        "status": "ok"
    }
}
```

List of RESTful API v1 requests

This section describes the DS8870 version-1 RESTful API services and includes examples using cURL.

Note: All requests must be routed to HTTP port 8452, which is the management port for the DS8870. The URI to invoke the different requests is case-sensitive and must comply with the following format:

`https://IBM_DS8870_hostname:8452/api/v1/resource_name`

Token

As previously mentioned, requesting an authentication token must be the first RESTful API request. Table 1 shows that POST is the only supported HTTP method.

Table 1 Table 1 Tokens RESTful API request

HTTP method	URI path	Description
POST	/v1/tokens	Token request for authentication.

The syntax of the token request is shown in Example 8. It must be built into the HTTP body of a POST request.

Example 8 Token request format

```
{
   request: {
         params: {
               username: ESSNI username,
               password: ESSNI password,
               maxliveInterval: in milliseconds,
               maxInactiveInterval: in milliseconds
         }
   }
}
```

Systems

The `systems` request returns details about the configuration and storage attributes of a managed DS8870. See Table 2.

Table 2 Systems RESTful request

HTTP method	URI path	Description
GET	/v1/systems	Retrieve detail system properties for managed DS8870.

Volumes

The `volumes` request provides a rich set of volume operations, as shown in Table 3.

Table 3 Volumes requests and operations

HTTP method	URI path	Description
GET	/v1/volumes/{volume_id}	Retrieve detailed properties for a volume by volume ID. **Note:** The volume ID is hexadecimal without any prefix.
GET	/v1/lss/{lss_id}volumes	Retrieve detailed properties of all volumes under specified LSS ID. **Note:** The LSS ID is hexadecimal without any prefix.
GET	/v1/pools/{pool_id}/volumes	Retrieve detailed properties of all volumes under a specified Storage Pool ID.
GET	/v1/hosts/{host_name}/volumes	Retrieve detailed properties of all volumes under a specified host.
POST	/v1/volumes	Create one or multiple Fixed Block volumes or Count-Key-Data base volumes.
PUT	/v1/volumes/{volume_id}	Modify properties for a specified volume. **Note:** The volume ID is hexadecimal without any prefix.
Delete	/v1/volumes/{volume_id}	Delete a Fixed Block volume or a CKD base volume.

Storage pool

Table 4 and Table 5 show all supported `pools` request operations. Operations support only attributes retrieval.

Table 4 Storage pool requests

HTTP method	URI path	Description
GET	/v1/pools	Retrieve properties of all managed storage pools.
GET	/v1/pools/{pool_id}	Retrieve properties of a specified storage pool.

URI endpoints of the space-efficient repository pool include both track space-efficient and extent space-efficient repository volumes. The detail properties of repositories can be retrieved and modified. See Table 5 for details.

Table 5 Space-efficient repository operations

HTTP method	URI path	Description
GET	/v1/pools/{pool_id}/tserep	Retrieve properties of the track space-efficient repository under a specified storage pool.
GET	/v1/pools/{pool_id}/eserp	Retrieve properties of the extent space-efficient repositories under a specified storage pool.
PUT	/v1/pools/{pool_id}/tserep	Modify properties of the track space-efficient repositories under a specified storage pool.
PUT	/v1/pools/{pool_id}/eserp	Modify properties of the extent space-efficient repositories under a specified storage pool.
DELETE	/v1/pools/{pool_id}/tserep	Delete the track space-efficient repository under a specified storage pool.
DELETE	/v1/pools/{pool_id}/eserp	Delete the extent space-efficient repository under a specified storage pool.

Logical Subsystem

Detail configuration of logical subsystem or logical control unit (IBM z Systems™) can be retrieved through the `lss` request as shown in Table 6.

Table 6 Logical subsystem configuration requests

HTTP method	URI path	Description
GET	/v1/lss[?type=fb|ckd]	Retrieve detail information of specified lss. You can also specify type of LSS as FB or CKD.
GET	/v1/lss/{lss_id}	Retrieve data information of specified LSS with LSS ID. **Note:** LSS ID is in hexadecimal format without any prefix.

Example 9 shows how to retrieve all fixed block logical subsystems, by using the cURL command line.

Example 9 Retrieve configuration of a DS8870 Fixed Block LSS

```
curl --tlsv1 -k -H "Content-Type: application/json" -H "X-Auth-Token:
bfc79c2a810344c0a6ac62dea0a4c700"
https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/lss?type=fb |python -m json.tool
```

Hosts

The `hosts` request allows you to retrieve, create, modify, or delete host configurations, as shown in Table 7.

Table 7 Hosts request and operations

HTTP method	URI path	Description
GET	/v1/hosts	Retrieve all host configuration informations.
GET	/v1/hosts/{host_name}	Retrieve host configuration information with specified host name
POST	/v1/hosts	Create Open System host attachments to Fibre Channel port.
PUT	/v1/hosts/{host_name}	Modify the host port assignment for a SCSI host.
DELETE	/v1/hosts/{host_name}	Delete a SCSI host connection.

Example 10 illustrates how to create a host connection with a specified host name and the `"LinuxRHEL"` host type. Next, the hosts information is retrieved for review.

Example 10 Create a DS8870 host configuration

```
[command_prompt]# curl --tlsv1  -k -H "Content-Type: application/json"  -H
"X-Auth-Token:d371f0d435b344a0b31f0501fa4c5a4b" -X POST
https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/hosts -d '{request:{params:{
name:"test_host", hosttype:"LinuxRHEL"}}}' |python -m json.tool
{
    "link": {
        "href": "https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/hosts/test_host",
        "rel": "self"
    },
    "server": {
        "code": "",
        "message": "Operation done successfully.",
        "status": "ok"
    }
}
##### Retrieve the configured hosts information
[command_prompt]# curl --tlsv1  -k -H "Content-Type: application/json"  -H
"X-Auth-Token:d371f0d435b344a0b31f0501fa4c5a4b" -X GET
https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/hosts/test_host |python -m json.tool
{
    "counts": {
        "data_counts": 1,
        "total_counts": 1
    },
    "data": {
        "hosts": [
            {
                "addrdiscovery": "lunpolling",
                "addrmode": "SCSI map",
                "host_ports": {
                    "link": {
                        "href":
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/hosts/test_host/host_ports",
                        "rel": "self"
                    }
                },
                "hosttype": "LinuxRHEL",
                "ioports": {
                    "link": {
                        "href":
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/hosts/test_host/ioports",
                        "rel": "self"
                    }
                },
                "lbs": "512",
                "link": {
                    "href":
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/hosts/test_host",
                    "rel": "self"
                },
                "name": "test_host",
                "state": "online",
                "volumes": {
                    "link": {
                        "href":
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/hosts/test_host/volumes",
                        "rel": "self"
                    }
                }
            }
        ]
    },
    "server": {
        "code": "",
        "message": "Operation done successfully.",
        "status": "ok"
    }
}
```

Host port

Table 8 lists the supported services for host_ports.

Table 8 Host port request and operations

HTTP method	URI path	Description
GET	/v1/host_ports	Retrieve all configured host ports configuration for a DS8870.
GET	/v1/host_ports/{wwpn}	Retrieve configuration information of a host port with specified worldwide port name (WWPN).
GET	/v1/hostss/{host_name}/ host_ports	Retrieve configuration of all host ports under a specified host name.
POST	/v1/host_ports	Configure open systems host ports for a DS8870.
PUT	/v1/host_ports/{wwpn}	Modify the Volume Group (Hosts) assignment for a SCSI host port.
DELETE	/v1/host_ports/{wwpn}	Remove a SCSI host port connection

Host volume mappings

Host volume mappings represent a mapping relationship of all the configured volumes under a specified host. The `mappings` RESTful API operations are listed in Table 9.

Table 9 Host volume mappings operations

HTTP method	URI path	Description
GET	/v1/hosts/{host_name}/mappings	Retrieve host volume mappings for a specified host.
GET	/v1/hosts/{host_name}/mappings /{mapping_id}	Retrieve host volume mappings for a specified mapping id (that is, host LUN) under a specified host.
POST	/v1/hosts/{host_name}/mappings	Create volume member mappings for a specified host.
DELETE	/v1/hosts/{host_name}/mappings /{mapping_id}	Delete host volume mappings for a specified mapping id (that is, host LUN) under a specified host.

Example 11 shows how to retrieve volume mappings for a specified host, using cURL.

Example 11 Retrieve a DS8870 volume mappings from RESTful API

```
[command_prompt]# curl --tlsv1  -k -H "Content-Type: application/json"  -H
"X-Auth-Token:244418984b8341faa95beb2ed0668999" -X GET
https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/hosts/test_host/mappings/4001401F
|python -m json.tool
{
    "counts": {
        "data_counts": 1,
        "total_counts": 1
    },
    "data": {
        "mappings": [
            {
                "link": {
                    "href":
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/hosts/test_host/mappings/4001401F",
                    "rel": "self"
                },
                "lunid": "4001401F",
                "volume": {
                    "id": "011F",
                    "link": {
                        "href":
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/volumes/011F",
                        "rel": "self"
                    }
                }
            }
        ]
    },
    "server": {
        "code": "",
        "message": "Operation done successfully.",
        "status": "ok"
    }
}
```

The response data shows that host LUN ID 4001401F indicated in the "lunid" field maps to volume 011F indicated in the "volume" subfield.

IO ports

Use the ioports request to retrieve the DS8870 I/O ports configuration, as shown in Table 10.

Table 10 IO Port requests

HTTP method	URI path	Description
GET	/v1/ioports	Retrieve all I/O ports configured from a DS8870.
GET	/v1/ioports/{port_id}	Retrieve specified I/O port configuration.
GET	/v1/hosts/{host_name}/ioports	Retrieve all I/O ports under a specified host name.

System node

A system node represents a central processor complex in a DS8870. You can retrieve system node information, using the nodes request, as listed in Example 11.

Table 11 Table 11 System nodes request

HTTP method	URI path	Description
GET	/v1/nodes	Retrieve status and ID information of all system nodes within a DS8870.
GET	/v1/nodes/{node_id}	Retrieve status and ID information of a specified node ID within a DS8870.

Example 12 illustrates the system node information returned for a DS8870.

Example 12 How to retrieve DS8870 system node information from RESTful API

```
[command_prompt]# curl --tlsv1  -k -H "Content-Type: application/json"  -H
"X-Auth-Token:244418984b8341faa95beb2ed0668999" -X GET
{
    "counts": {
        "data_counts": 2,
        "total_counts": 2
    },
    "data": {
        "nodes": [
            {
                "link": {
                    "href":
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/nodes/00",
                    "rel": "self"
                },
                "server_id": "00",
                "state": "online"
            },
            {
                "link": {
                    "href":
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/nodes/01",
                    "rel": "self"
                },
                "server_id": "01",
                "state": "online"
            }
        ]
    },
    "server": {
        "code": "",
        "message": "Operation done successfully.",
        "status": "ok"
    }
}
```

IBM FlashCopy

The `flashcopy` API request is used to retrieve IBM FlashCopy® configuration information as shown in Table 12.

Table 12 FlashCopy requests

HTTP method	URI path	Description
GET	/v1/flashcopy	Retrieve FlashCopy configurations within a DS8870.
GET	/v1/volumes/{volume_id}/flashcopy	Retrieve FlashCopy configurations for a specified volume.

Remote Mirror and Copy

The `pprc` API request is used to retrieve Remote Mirror and Copy (formerly known as PPRC) configuration information, including Metro Mirror, Global Copy, and Global Mirror as listed in Table 13.

Table 13 Remote Mirror and Copy requests

HTTP method	URI path	Description
GET	/v1/pprc	Retrieve the configurations of Remote Mirror and Copy services of a DS8870.
GET	/v1/volumes/{volume_id}/pprc	Retrieve the configuration of Remote Mirror and Copy services for a specified volume. **Note:** The volume ID is in hexadecimal format without any prefix. It must also be the primary volume in a Remote Mirroring relationship.

Example 13 illustrates how to retrieve the configuration information for a Global Copy primary volume. The resulting data indicates that volume 0x113F, which is the primary volume in system IBM_DS8870, is mirrored to a remote volume 0x153F in a system with ID `"IBM.2107-1300891"` and that the relationship type is `globalcopy`.

Example 13 Using the pprc RESTful API request

```
[command_prompt]# curl --tlsv1  -k -H "Content-Type: application/json"  -H
"X-Auth-Token:2875ce068998408597840967d4bd9e4d" -X GET
https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/volumes/113F/pprc |python -m json.tool
{
    "counts": {
        "data_counts": 1,
        "total_counts": 1
    },
    "data": {
        "pprc": [
            {
                "sourcevolume": {
                    "id": "113F",
                    "link": {
                        "href":
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/volumes/113F",
                        "rel": "self"
                    }
                },
                "state": "copy_pending",
                "targetsystem": {
                    "id": "IBM.2107-1300891",
                    "link": {}
                },
                "targetvolume": {
                    "id": "153F",
                    "link": {}
                },
                "type": "globalcopy"
            }
        ]
    },
    "server": {
        "code": "",
        "message": "Operation done successfully.",
        "status": "ok"
    }
}
```

Events

The `events` API request can retrieve detailed event information, by severity and for a given time frame, as shown in Table 14.

Table 14 Events request

HTTP method	URI path	Description
GET	/v1/events/[?severity=info, warning, error&after=yyyy-MM-ddTHH:mm:ssZ&before=yyyy-MM-ddTHH:mm:ssTZ]	Retrieve event information. **Note:** You can specify the duration and severity of events for an application's requirement. TZ is the time zone information you can specify.

Example 14 shows how to list all warning events within specified time frame.

Example 14 Retrieving system events

```
[command_prompt]# curl --tlsv1  -k -H "Content-Type: application/json"  -H
"X-Auth-Token:a0b2f372b0704660824fbf7cbf68d247" -X GET
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/events?severity=warning&after=2013-04-2
0T12:04:05-0700" | python -m json.tool
{
    "counts": {
        "data_counts": 1,
        "total_counts": 1
    },
    "data": {
        "events": [
            {
                "formatted_parameter": [
                    "mb12",
                    "11",
                    "2"
                ],
                "id": "SE65c1",
                "resource_id": "IBM.2107-75KX511",
                "severity": "warning",
                "time": "2015-04-20T02:27:26-0700",
                "type": "StorageSystemStateChanged"
            }
        ]
    },
    "server": {
        "code": "",
        "message": "Operation done successfully.",
        "status": "ok"
    }
}
```

RESTful API services self-upgrade

The DS8870 RESTful API supports self-upgrade of RESTful API services. This API service can be used to update of RESTful services when available. Table 15 illustrates supported services for self-upgrade.

Table 15 RESTful API services self-upgrade

HTTP method	URI path	Description
GET	/v1/self_upgrade	Retrieve the self-upgrade information including time, status, and message.
POST	/v1/self_upgrade	Perform self-upgrade with a specified file image.

Example 15 shows how to perform self-upgrade from a specified file, in cURL.

Example 15 How to perform DS8870 self-upgrade from RESTful API

```
curl –k –H H X-Auth-Token:9d376ecd47bd4fb8a0264fb6626d291f -F
file=@E:\ESSNI\Restful\upload\R15g.7b150122a\apiSelfUpgradePack.tar.bz2 -F nick=go
https://mtc032h.tuc.stglabs.ibm.com:8452/api/v1/self_upgrade
```

System performance

System performance information can be retrieved by using the `performance` request as shown in Table 16.

Table 16 System performance URI endpoints

HTTP method	URI path	Description
GET	/v1/systems/{sn}/performance[?after =yyyy-MM-ddTHH:mm:ssZ&before= yyyy-MM-ddTHH:mm:ssTZ]	Retrieve system performance information.

Example 16 illustrates how to retrieve performance information within a specified time frame.

Example 16 How to retrieve DS8870 system performance from RESTful API

```
[command_prompt]# curl --tlsv1  -k -H "Content-Type: application/json"  -H
"X-Auth-Token:a0b2f372b0704660824fbf7cbf68d247" -X GET
"https://IBM_DS8870.tuc.stglabs.ibm.com:8452/api/v1/systems/75KX511/performance?af
ter=2015-04-15T14:42:25-0700" | python -m json.tool
{
    "counts": {
        "data_counts": 9809,
        "total_counts": 9809
    },
    "data": {
        "performance": [
            {
                "IOPS": {
                    "read": "40012.7",
                    "total": "81653.53",
                    "write": "41640.82"
                },
                "performancesampletime": "2015-04-22T14:44:06-0700",
                "responseTime": {
                    "average": "2.67",
                    "read": "1.89",
                    "write": "3.41"
                }
            },
..
```

Authors

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), San Jose Center.

Bert Dufrasne is an IBM Certified IT Specialist and Project Leader for IBM System Storage® disk products at the ITSO, San Jose Center. He has worked at IBM in various IT areas. He has written many IBM Redbooks® publications and has developed and taught technical workshops. Before joining the ITSO, he worked for IBM Global Services as an Application Architect. He holds a Master's degree in Electrical Engineering.

Adrian Orben joined IBM in 2003. He is a member of the Storage Support Team, since 2006, responsible for IBM High End Disk products. His focus is in support for IBM DS8000®, IBM XIV®, and IBM Flash Systems. During this time, Adrian achieved several storage-related certifications. Adrian holds a Bachelor's degree in Business Informatics from the University of Applied Science, Mainz, Germany.

Bob Xiao is a certified DS8000 Product Field Engineer in IBM China. Bob joined IBM in 2007 as a Software Engineer for development of BladeCenter firmware. He then moved to the IBM DS8000 Product Field Engineering team in 2009. His current major work includes DS8000 client support, DS8000 service, and quality improvements. He is also an experienced DS8000 client advocate for IBM DS8000 customers. He holds a Bachelor's degree in Automation and Master's degree in Electronic Engineering also.

Thanks to the following people for their contributions to this project:

Justin Cripps, Eddie Lin
IBM

Now you can become a published author, too!

Notices

This document REDP-5187-00 was created or updated on July 9, 2015.

Send us your comments in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbooks@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Trademarks

REDP-5187-00

ISBN 0738454281