MX7000 REST-ful Interface APIs

Dell EMC Technical White Paper

Revisions

Date	Description
Jan 2019	Initial release

Acknowledgements

This paper was produced by the following members of the Dell EMC storage engineering team:

Author: Heidi Maeder, Andy Drysdale, Vijayasimha Naga and Christopher Poblete

Table of contents

Introduction

In the past years Intelligent Platform Management Interface (IPMI) unified server management by common commands to control servers from multiple manufacturers. With growing demands from cloud- and web-based data center infrastructure the RESTful and Redfish interface of the Management Module in the PowerEdge MX7000 is a modern standard. The infrastructure provides a Web based Hypermedia driven API using a simple folder structure, the client provided with the Service root URI can navigate through the entire resource tree through the links in the response payload on the individual URIs. The benefits are increased IT productivity, scalability, reduce CapEx, speed time to value, and enhanced security. Also, the Redfish and RESTful APIs enable web-scale automation.

Redfish is a scalable management API based on standards that uses RESTful interface. It provides a common interface across platforms and vendors very beneficial to DevOps automation. Currently the Redfish implementation is limited in the 1.0 release, but will be expanded in subsequent releases. Dell EMC and SPMF continue to drive Redfish development to meet customers' needs for management automation through open, industry standards. The later MX7000 REST releases will provide the overall benefit in Redfish development with the inherent ease of use.

MX7000 REST interface implementation uses the secure HTTPS for secure and reliable communication. The following documentation is an overview of the RESTful interface APIs available for the consumption of MX7000 users. Additional information can be retrieved from the REST API Reference Guide for OpenManage Enterprise Modular (OME-M), the MX7000 chassis management firmware.

Accessing the RESTful interface using Postman

The RESTful interface of OME-M can be accessed using your favorite REST client application. The examples in this document uses the Postman REST application.

The MX7000 RESTful interface can be navigated using the following steps. The root API doesn't need authentication, however the same is required to navigate further down the hierarchy. The interface supports Basic and Session based authentication, the examples in this document use Basic Authentication Scheme.

1. Install Postman via https://www.getpostman.com/apps
2. After installation -open postman
3. In Authorization tab fill the following information

   Type: Basic Auth

   Username: username of system

   Password: password of system

4. Within address bar enter the URI 'https://<ip>/redfish/v1' with HTTP GET
5. Press the blue Send button

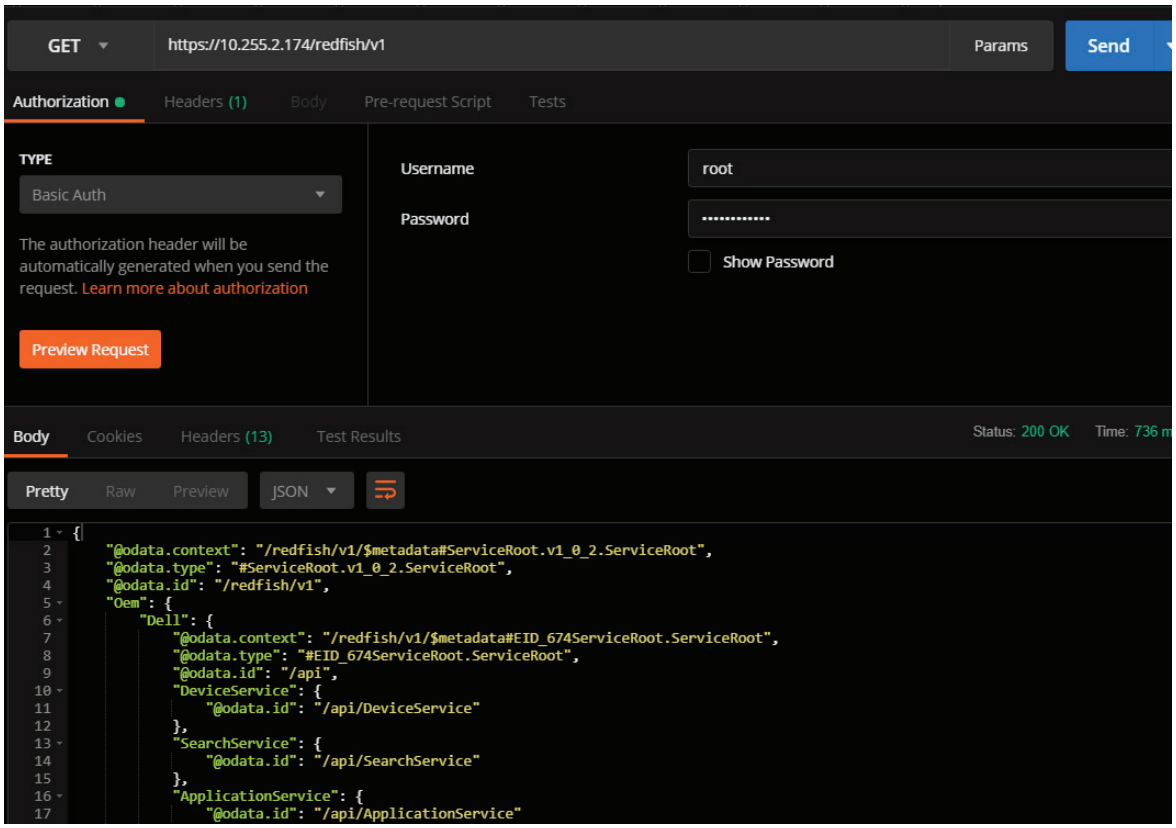

Figure 1 Accessing the Redfish service root in Postman

OME-M REST schema

The OME-M RESTful interface comprises of the Redfish schema compliant APIs under the resource URI “/redfish/v1” and OEM APIs under resource URI “/api”..

The following sections provide an introduction to the commonly used URI’s.

OEM resources URIs

/api/DeviceService

All the devices managed by the Chassis can be found using the /api/DeviceService/Devices URI. The different types of devices that can be found are Storage Sleds, Compute Sleds, IOMs, and Chassis. Each device will uniquely be identified by a device ID. The device specific inventory information can be retrieved using the URI /api/DeviceService/Devices(ID)/InventoryDetails. Traversing the links further will provide more detailed information of devices depending upon the device type like Fans, Power Supply etc., for a Chassis Device, Processors, Memory Information for a Compute Sled etc.,

The Id for each device can further be used when performing certain actions on devices such as power control operations, blink actions, firmware updates and more. Additional information on how the device ID is used to run an action or 'Job' can be found in /api/JobService section.

The other URI's within /api/DeviceService/ are useful to translate enumeration codes for certain attributes in the /api/DeviceService/Devices URI.

When the state of a device is not healthy, the associated Fault (or error messages that explain why the health state is either Critical or Warning) can be found under the SubSystemHealth link of the Device resource.

/api/GroupService

A group can be created containing different devices or subgroups. The group can be viewed, re-configured, deleted, and user privileges can be set.

/api/JobService

This api provides a path to execute certain Actions on the Devices or Group of Devices. Each Action shall have a Job ID allocated to it upon invocation and the status of the job can be monitored within /api/JobService/Jobs(ID).

The different type of jobs that can be created can be seen using /api/JobService/JobTypes. Each job creation requires a payload that includes the attributes Job Type, Target Type (Device or Group), Target ID (Device ID or Group ID) (refer to /api/DeviceService/Devices & /api/GroupService/Groups section) and other attributes specific to the Job Type.

Actions that can be executed on JobService are

1. Create
2. Stop
3. Delete
4. Enable
5. Disable

Example of Job Execution

Power Cycle Cold Boot – JobType DeviceAction_Task

The following information needs to be filled`

- Authorization type: Basic Auth
- Username and Password information
- JSON content with the link is in the picture below. Data should be raw with 'JSON(application/json) applied. Line 'Id": 25015 is the device Id on which the user intends to perform the operation.
- The Action next to the URI is 'POST'
- Click 'Send' blue button
- Return code of the https RESTful job can be found in 'Status'
- The HTTP Headers from the response shall contain an Id allocated for the Job. The Job can be monitored within /api/JobService/Jobs(Id)

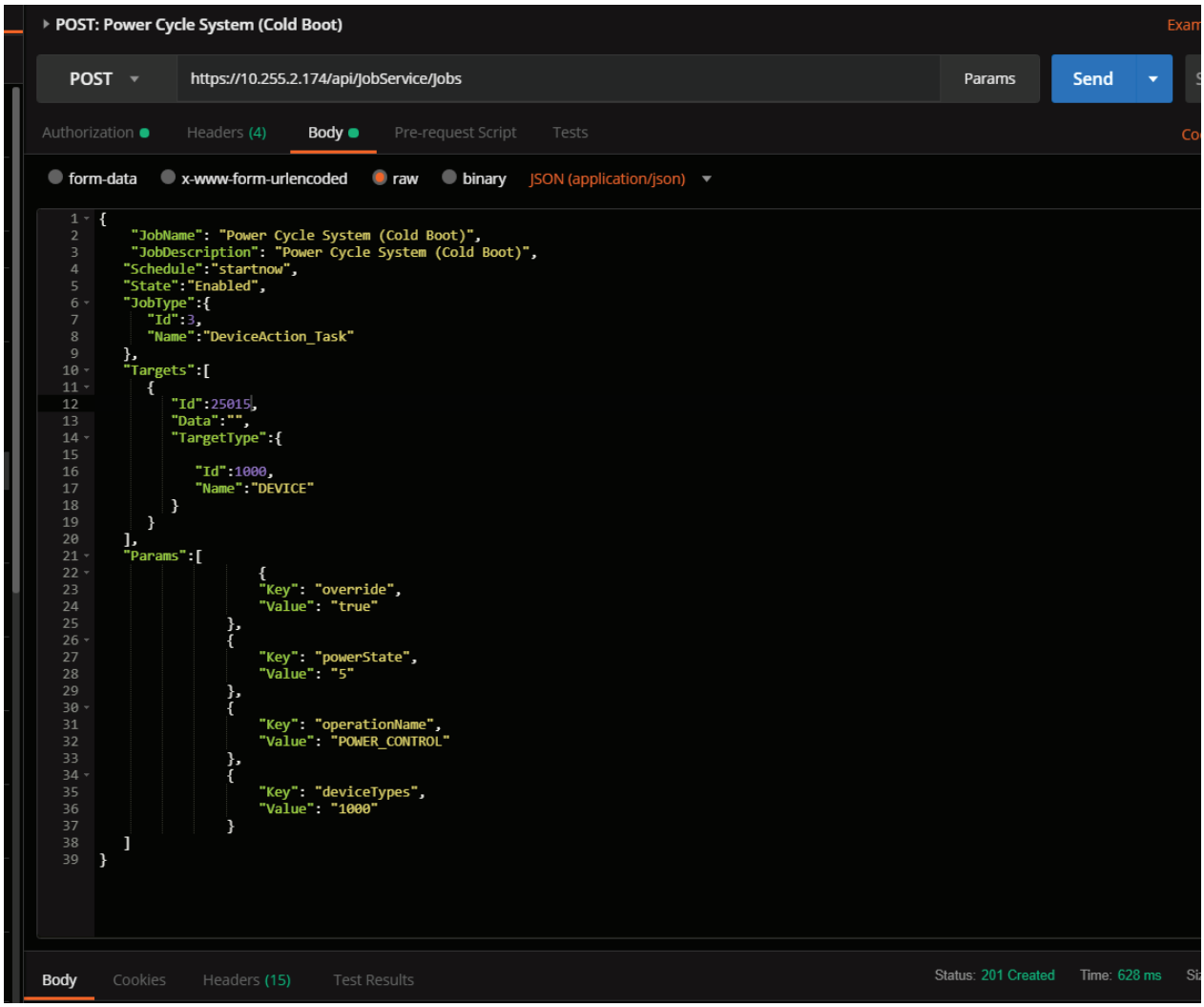

Figure 2 Example POST to create a Job using Postman

/api/ApplicationService

This URI provides a path for user to know the status of the Application, retrieve audit logs, configure and retrieve network & time configuration, generate, apply & retrieve Certificates etc.,.

/api/AlertService

The alert log information of the entire system can be retrieved.

Template configuration on logs will notify a user base using alert types SNTP, SNMP, or syslog.

/api/UpdateService

This URI provides a path to upload firmware(DUP) for a device, the user may further schedule the Update Job via /api/JobService.

/api/AccountService

The user account can be retrieved and configured.

Accounts can be deleted, enabled, disabled, and the permissions can be retrieved. Active-Directory services can be configured and retrieved.

/api/ManagementDomainService

The open manage enterprise modular can be added or deleted to a domain group. A member can be disconnected or added to the domain group. Configuration types of the management domain service can be viewed and configured.

/api/SearchService

Will allow user to perform search operations for file or for strings in the REST-ful interface.

Redfish resource URIs

/redfish/v1/Chassis

This is similar to the /api/DeviceServices/Device tree. The API provides information of all devices in the Chassis including an instance of the Chassis itself represented via Chassis Model defined by DMTF.

/redfish/v1/Systems

This API provides information of all Compute Sleds in the Chassis represented via ComputerSystem Model defined by DMTF.

/redfish/v1/Registries

This API Provides the definition of event and error messages.

/redfish/v1/SessionService

The APIs in the hierarchy of this path enables the user to access the session service in open manage enterprise –modular. The session can be viewed, created, deleted, and more.

/redfish/v1/AccountService

This resource enables you to access the account configuration for OpenManage Enterprise—Modular. Using this resource, you can view the accounts and their roles, update the configuration of accounts, delete accounts, and so on

Odata Filter Support

MX7000 REST implementation allows filtering per Odata constructs on certain APIs that returns a collection of Entities. This feature is limited to only a selected set of APIs in the 1.0 release. The $filter option isolates a subset of the entries from a collection of entries associated with the resource path (/api/ApplicationService/AuditLogs for example). By selecting an appropriate predicate expression, data entries can effectively be filtered and isolated.

Data filtering:

The REST interface provides filtering options on certain URIs that return a collection of entities, this enables the clients to extract a selected set of records using comparison operators on certain attributes of the model entity behind the collection.

Ex: Filter Devices by System Id
/api/DeviceService/Devices?$filter=SystemId eq 123

Ex: Filter Devices by Model
/api/DeviceService/Devices?$filter=Model eq ‘PowerEdge MX740c’

Ex: Filter Devices by Device Type
/api/DeviceService/Devices?$filter=Type eq 3000 or Type eq 5000 ***Note**: For enumeration of Type, see/api/DeviceService/DeviceType*

Data Sorting:

The Console Software provides sorting options on certain URIs that return a collection of entities, this enables the clients to get sorted results using the Sort operators on certain attributes of the model entity behind the collection.

Ex: Get Devices sorted in ascending order by Status
/api/DeviceService/Devices?$orderby=Status asc ***Note**: For enumeration of Status, see /api/DeviceService/DeviceStatuses*

Data Pagination:

The Console Software provides Pagination options on certain URIs that return a collection of entities, this enables the clients to get paginated results.

Ex: Get 5th to 8th Devices from the Device Collection
/api/DeviceService/Devices?$skip=4&$top=4

Ex: Get first 4 Devices from the Device Collection
/api/DeviceService/Devices?$top=4

References

https://www.dmtf.org/sites/default/files/standards/documents/DSP2052_1.0.0.pdf

https://www.dmtf.org/standards/redfish

http://docs.oasis-open.org/odata/odata/v4.0/odata-v4.0-part1-protocol.html