DISTRIBUTED MANAGEMENT TASK FORCE

Technical Note

August 2015

Redfish – Simple, Modern and Secure Management for Multi-Vendor Cloud and Web-Based Infrastructures

Introduction

In today's cloud- and web-based data center infrastructures, scalability is often achieved through large quantities of simple servers. This hyperscale usage model is drastically different than that of traditional enterprise platforms, and requires a new approach to management.

As an open industry standard that meets scalability requirements in multi-vendor deployments, Redfish integrates easily with commonly used tools by specifying a RESTful interface and utilizing JSON and OData.

Why a new interface?

A variety of influences have resulted in the need for a new standard management interface. First, the market is shifting from traditional data center environments to scale-out solutions. With massive quantities of simple servers brought together to perform a set of tasks in a distributed fashion (as opposed to centrally located), the usage model is different from traditional enterprise environments. This set of customers demands a standards-based management interface that is consistent in heterogeneous, multi-vendor environments.

However, functionality and homogeneous interfaces are lacking in scale-out management. IPMI, an older standard for out-of-band management, is limited to a "least common denominator" set of commands (e.g. Power On/Off/Reboot, temperature value, text console, etc.). As a result, customers have been forced to use a reduced set of functionality because vendor extensions are not common across all platforms. This set of new customers is increasingly developing their own tools for tight integration, often having to rely on in-band management software.

Seeking a modern interface that builds on widely-used tools to accelerate development, today's customers expect APIs to use the protocols, structures and security models that are common in emerging cloud interfaces. Specifically, these customers are asking for RESTful protocols that express data in JSON formats.

DMTF's Redfish API

Offering a secure, multi-node capable replacement for IPMI-over-LAN, Redfish uses REST, JSON and OData to address modern customer requirements. Usable by existing client applications and browser-based GUIs,

TABLE OF CONTENTS

1 Introduction
1 Why a new interface?
1 DMTF's Redfish API
2 Why REST, JSON and OData?
2 Why separate the data model from the protocol?
2 Basic Concepts
3 Collections
3 Operations
3 Conclusion
3 Recommended Reading

Redfish delivers powerful simplicity with its human-readable output.

Why REST, JSON and OData?

RESTful protocols are rapidly replacing SOAP. The cloud ecosystem is adopting REST and the web API community has followed suit.

RESTful protocols are much quicker to learn than SOAP. They have the simplicity of being a data pattern (as REST is not strictly a protocol) mapped to HTTP operations directly**.**

JSON is fast becoming the modern data format. It is inherently human readable, more concise than XML, has a plethora of modern language support and is the most rapidly growing data format for web service APIs.

JSON has one additional advantage for embedded manageability environments: most Baseboard Management Controllers (BMCs) already support a web server and managing a server through a browser is common (typically via a Java script-driven interface). By utilizing JSON in Redfish, the data from a Redfish service is viewed directly in the browser, ensuring the data and the programmatic interface is uniform in semantics and value.

But following RESTful practices and formatting results as JSON alone are not enough for interoperability. There are nearly as many RESTful interfaces as there are applications, and they all differ in the resources they expose, the headers and query options available, and how results are represented. Similarly, while JSON provides an easy-to read representation, semantics of common properties such as id, type, links, etc., are imposed through naming conventions that vary from service to service.

OData defines a set of common RESTful conventions, which provides for interoperability between APIs. Redfish adopts common OData conventions for describing schema, URL conventions, and naming, as well as the structure of common properties in a JSON payload. This not only encapsulates best practices for RESTful APIs which can be used in traditional and scalable environments, but further enables Redfish services to be consumed by a growing ecosystem of generic client libraries, applications, and tools.

Why separate the data model from the protocol?

The Redfish data model is extensible and is expected to cover additional properties in the future. The Redfish protocol, however, is expected to require fewer updates. Therefore, Redfish separates the protocol specification from the data model to avoid unnecessary versioning. Strict forward compatibility rules are included in the specification.

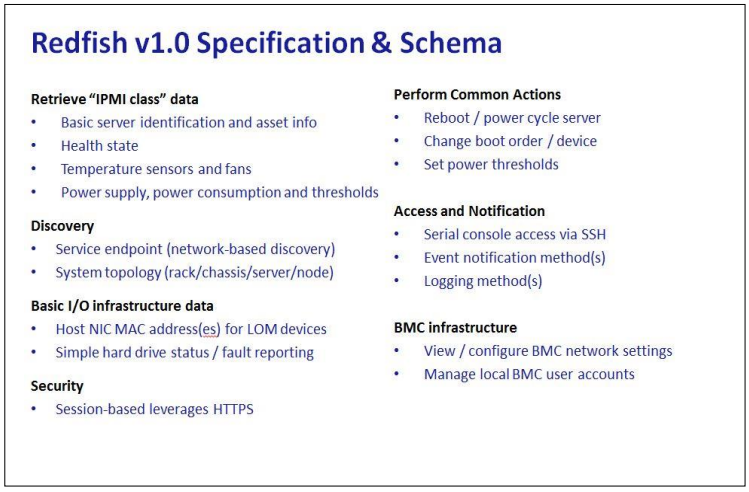

Fig 1 – Redfish Capabilities

Basic Concepts

In Redfish, every URL represents a resource, a service, or a collection of resources. In RESTful terms, these Uniform Resource Identifiers (URIs) point to resources and clients interact with resources.

The resource format is defined by the Redfish Schema, which the client can use to determine

the correct semantics, if needed (Redfish semantics are designed to be largely intuitive).

The Redfish Schema is defined in two formats. It is defined in the OData Common Schema Definition Language (CSDL), so generic OData tools and applications can interpret it. The schema is also defined in the JSON Schema format for other environments, such as Python scripts, JavaScript code and visualization.

Collections

In Redfish, a collection represents a group of similar resources. Examples include Systems, Managers and Chassis.

A System represents the logical view of a computer system. Any subsystem accessible from the host CPU is represented in a System resource. Each System instance will have CPUs, memory and other components. Computer systems are contained as members of the Systems collection.

The Managers collection contains BMCs, Enclosure Managers or any other component managing the infrastructure. Managers handle various management services and can also have their own components (such as NICs).

The Chassis collection contains resources that represent the physical aspects of the infrastructure. A single Chassis resource can house sensors, fans and the like. Racks, enclosures and blades are examples of Chassis resources included in the Chassis collection. In addition, Redfish provides a method to represent a Chassis contained within another Chassis.

Operations

Redfish uses HTTP operations including GET, PUT, PATCH, POST, DELETE and HEAD. GET retrieves data. POST is used for creating resources or to use actions. DELETE will delete a resource, but there are currently only a few resources that can be deleted. PATCH is used to change one or more properties on a resource, while PUT is used to replace a resource entirely (though only a few resources can be completely replaced). HEAD is similar to GET without the body data returned, and can be used for figuring out the URI structure by programs accessing a Redfish implementation.

Conclusion

Redfish represents a new style of standard that is capable of managing modern IT infrastructures – from hyperscale to blades to stand alone servers – in a consistent manner. As a result of broad industry collaboration, Redfish meets customer demands for simple, modern and secure management of scalable platform hardware, reducing vendor lock-in and increasing the productivity of system administrators.

Recommended Reading

- Redfish White Paper – dmtf.org/sites/default/files/standards/documents/DSP2044_1.0.0.pdf
- Redfish FAQ - dmtf.org/sites/default/files/standards/documents/DSP2045_1.0.0.pdf
- BrightTALK Webcast: Redfish Data Model Deep Dive – www.brighttalk.com/webcast/9077/163783
- BrightTALK Webcast: DMTF: Redfish Overview – https://www.brighttalk.com/webcast/9077/156709
- Redfish Home Page - www.dmtf.org/standards/redfish

Acknowledgements

Work on the Redfish standard takes place in the DMTF's Scalable Platforms Management Forum (SPMF) (http://dmtf.org/standards/spmf). SPMF members contributed to this Technical Note.