Document Identifier: DSP2032

Date: 2015-02-19

Version: 2.0.0

CIM-RS White Paper

Supersedes: 1.0

Document Type: White Paper

Document Class: Informative

Document Status: Published

Document Language: en-US

CONTENTS

Tables

Abstract

This white paper provides background information for CIM-RS as defined in the DMTF specifications *CIM-RS Protocol* (DSP0210) and *CIM-RS Payload Representation in JSON* (DSP0211). This white paper will provide some explanation behind the decisions made in these specifications and give the reader insight into when the use of CIM-RS may be appropriate. There is also discussion of some of the considerations in choosing payload encodings such as JSON or XML.

This paper is targeted to potential users of CIM-RS who are considering developing a server-side interface to a CIM implementation that follows REST principles, or a client that consumes such an interface.

Foreword

The *CIM-RS White Paper* (DSP2032) was prepared by the DMTF CIM-RS Working Group, based on work of the DMTF CIM-RS Incubator.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. For information about the DMTF, see http://www.dmtf.org.

Acknowledgments

The DMTF acknowledges the following individuals for their contributions to this document:

- Cornelia Davis, EMC
- George Ericson, EMC
- Johannes Holzer, IBM
- Robert Kieninger, IBM
- Wojtek Kozaczynski, Microsoft
- Lawrence Lamers, VMware
- Andreas Maier, IBM (editor)
- Bob Tillman, EMC
- Marvin Waschke, CA Technologies (editor)

Document conventions

Typographical conventions

The following typographical conventions are used in this document:

- Document titles are marked in *italics*.

Deprecated and experimental material

A white paper has informative character. Therefore, material is not marked as experimental or deprecated as it would be in normative DMTF specifications.

Executive summary

The DMTF Common Information Model (CIM) is a conceptual information model for describing computing and business entities in Internet, enterprise, and service-provider environments. CIM uses object-oriented techniques to provide a consistent definition of such entities: A CIM model describes the state, relations, and behaviors of such managed objects. The CIM Schema published by DMTF is one such CIM model, establishing a common description of certain managed objects.

CIM and the CIM Schema provide a foundation for IT management software that can be written in one environment and easily converted to operate in a different environment. It also facilitates communication between software managing different aspects of the IT infrastructure. In this way, CIM and CIM Schema provide a basis for an integrated IT management environment that is more manageable and less complex than environments based on narrower and less consistent information.

CIM is built on object oriented principles and provides a consistent and cohesive programming model for IT management software. One of the developing trends in enterprise network software architecture in recent years has been Representational State Transfer (REST). REST represents a set of architectural constraints that have risen from the experience of the World Wide Web. Developers have discovered that the architecture of the web offers some of the same benefits in simplicity and reliability to enterprise software as it has provided over the Internet. IT management is an important application of enterprise software and there is growing interest in using CIM and CIM Schema based software in an architecture that follows REST constraints.

Fortunately, CIM follows basic architectural principles that largely fit well into RESTful architectures. As a result, the RESTful protocol defined by CIM-RS is tailored to the needs of CIM.

1 Terminology

In this document, some terms have a specific meaning beyond the normal English meaning. Those terms are defined in this clause.

Some of the terms and abbreviations defined in DSP0198 (such as "WBEM", "CIM", "URI", and others) are used in this document but are not repeated in this clause.

1.1

application state

the state that indicates where an application is in completing a task. In a RESTful system, the client is solely responsible for application or session state. The server is only responsible for resource state, the state of the resources managed by the service. An example of resource state is the account balance in a banking service, which would be maintained by the server. An example of application state is a specific client that has posted a deposit and is waiting for it to clear. Only the client would track the fact that it has posted a deposit request.

1.2

CIM-RS

CIM RESTful Services

the RESTful protocol for CIM covered by this white paper and related documents.

1.3

HATEOAS

Hypertext As The Engine Of Application State

the practice of using links embedded in resource representations to advertise further possible activities or related resources to the application. For example, an “order” link might be placed in the resource representation for an item offered in a catalog. The presence of the order link indicates that the item is orderable and represents a path to order the item. In a visual representation, the “order” link would appear as a button on the screen. Pushing the button, a POST or PUT HTTP method targeting the resource identifier provided in the link would be issued and would cause the item to be ordered. The returned resource represents the next application state, perhaps a form for entering quantity and shipping method. CIM-RS supports this concept by returning resource identifiers to related resources, for details see DSP0210.

1.4

HTTP content negotiation

negotiation between HTTP clients and HTTP servers to determine the format of the content transferred. When a client makes a request, they list acceptable response formats by specifying media types in an `Accept` header. Thus, the server is able to supply different representations of the same resource identified with the same resource identifier. A common example is GIF and PNG images. A browser that cannot display PNGs can be served GIFs based on the `Accept` header. In a RESTful system, the choice is more often between XML and JSON. For details, see RFC2616. Its use in CIM-RS is described in DSP0210.

1.5

JSON

JavaScript Object Notation, defined in RFC7159.

1.6

idempotent HTTP method

an HTTP method with the behavior that (aside from error or expiration issues) the side-effects of N consecutive identical requests are the same as for a single one of those requests. RFC2616 requires the

HTTP methods GET, HEAD, PUT and DELETE to be idempotent. HTTP methods that have no side effects (that is, safe methods) are inherently idempotent. For details, see RFC2616.

1.7

Internet media type

a string identification for representation formats in Internet protocols. Originally defined for email attachments and termed "MIME type". Because CIM-RS is based on HTTP, it uses the definition of media types from section 3.7 of RFC2616.

1.8

resource

in CIM-RS, an entity that can be referenced using a resource identifier and thus can be the target of an HTTP method. Example resources are systems, devices, or configurations.

1.9

resource identifier

in CIM-RS, a URI that is a reference to (or an address of) a resource. Generally, a resource may have more than one resource identifier; however in CIM-RS that is not the case.

1.10

resource representation

a representation of a resource or some aspect thereof, in some format. A particular resource may have any number of representations. The format of a resource representation is identified by a media type. In CIM-RS, the more general term "payload representation" is used, because not all protocol payload elements are resource representations.

1.11

resource state

the state of a resource managed by a RESTful service, in contrast to application state.

1.12

REST

Representational State Transfer

a style of software architecture for distributed systems that is based on addressable resources, a uniform constrained interface, representation orientation, stateless communication, and state transitions driven by data formats. Usually REST architectures use the HTTP protocol, although other protocols are possible. See *Architectural Styles and the Design of Network-based Software Architectures* for the original description of the REST architectural style.

1.13

RPC

Remote Procedure Call

an RPC is an implementation of a function in which a call to the function occurs in one process and the function is executed in a different process, often in a remote location linked by a network. RPC-based systems are often contrasted with RESTful systems. In a RESTful system, the interactions between client and server follow the REST constraints and the design focus is on the resources. In an RPC-based system, the design focus is on the functions invoked, and there is not necessarily even the notion of well-defined resources.

1.14

safe HTTP method

an HTTP method that has no side-effects. RFC2616 requires the HTTP methods GET and HEAD to be safe. By definition, an HTTP method that is safe is also idempotent.

1.15

SOAP

Simple Object Access Protocol, defined by the W3C.

2 Why build a RESTful interface for CIM

There has been a great deal of interest in constructing RESTful enterprise applications in the last few years and this interest has inspired the specification of CIM-RS. To understand the origins of this interest, the nature of REST and its relationship to IT management must be explored.

Enterprise applications are being built more and more frequently on architectures that involve remote network connections to some part of the implementation of the application. These connections are often via the Internet. This is especially true with the rise of cloud computing.

REST is a set of architectural constraints that were designed around the features of the Internet. For example, REST constraints are designed to assure that applications that follow constraints will have maximum benefit from typical Internet features like caches, proxies, and load balancers.

In addition, REST constraints are closely tied to the design of HTTP, the primary application level protocol of the Internet. In fact, the prime formulator of REST, Roy Fielding, was also an author of the HTTP standard. Consequently, REST was designed to take full advantage of HTTP and HTTP meets the needs of REST.

Some of the specific benefits that have been experienced in RESTful applications are:

- **Simplicity.** REST limits itself to the methods implemented in HTTP and runs directly on the HTTP stack. Note, however, that this simplicity can be deceptive. The design effort to comply with REST may engender its own complexity.
- **Resilience in the face of network disturbance.** One of the hallmarks of a RESTful application is a stateless relationship between the server and the client. Each request from the client contains all the history the server needs to respond to the client. Therefore recovery when a server becomes inaccessible does not require unwinding a stack and complex recovery logic when requests are self-contained and independent.
- **Upgradability.** The operations available in RESTful application are discovered by the client as the processes occur. Consequently, in some cases, the server implementation often may be upgraded transparently to the client. In some cases, a well-designed client may be able to take advantage of new features automatically.

Although these are important benefits, it is important to note that REST is not a panacea. Not all activities are easily compatible with its constraints. Not every operation fits easily into the stateless paradigm. The discoverability of RESTful applications may breakdown as applications become more complex and transactions become more elaborate.

Nevertheless, as a result of these benefits and others, a substantial number of developers of IT management applications that use CIM and CIM Schema have turned to REST. Therefore, there is a need for a specification for a uniform protocol that will promote interoperability between RESTful CIM and CIM Schema based applications.

3 Characteristics of a RESTful protocol and CIM-RS

The characteristics of a RESTful protocol are not standardized or otherwise defined normatively. The principles and constraints of the REST architectural style have originally been described by Roy Fielding in chapter 5 of *Architectural Styles and the Design of Network-based Software Architectures*. The BLOG entry *REST APIs must be hypertext driven* authored by Roy Fielding provides further insight into REST principles. While that description of the REST architectural style is not limited to the use of HTTP, the HTTP protocol comes close to supporting that style and obviously has a very broad use.

The CIM-RS protocol is based on HTTP and supports the REST architectural style to a large degree. The following list describes to what extent the typical REST constraints are satisfied by the CIM-RS protocol:

- **Client-Server:** The participants in the CIM-RS protocol are WBEM client, WBEM server, and WBEM listener. WBEM stands for *Web Based Enterprise Management* and is a set of protocols for systems management defined by the DMTF. There is a client-server relationship between WBEM client and WBEM server, and one between WBEM server and WBEM listener, where the WBEM server acts as a client to the WBEM listener. Thus, the WBEM server has two roles: To act as a server in the interactions with the WBEM client, and to act as a client in the interactions with the WBEM listener.
  This REST constraint is fully satisfied in CIM-RS.

- **Stateless:** Interactions in CIM-RS are self-describing and stateless in that the servers (that is, the WBEM server in its server role, and the WBEM listener) do not maintain any application state or session state.
  This REST constraint is fully satisfied in CIM-RS.

- **Cache:** The HTTP methods used in CIM-RS are used as defined in RFC2616. As a result, they are cacheable as defined in RFC2616.
  This REST constraint is fully satisfied in CIM-RS.

  NOTE RFC2616 defines only the result of HTTP GET methods to be cacheable.

- **Uniform interface:** The main resources represented in CIM-RS are instances or collections thereof, representing modeled objects in the managed environment. CIM-RS defines a uniform interface for creating, deleting, retrieving, replacing, and modifying these resources and thus the represented objects, based on HTTP methods.
  This REST constraint is satisfied in CIM-RS, with the following deviation:

  CIM methods can be invoked in CIM-RS through the use of HTTP POST. This may be seen as a deviation from the REST architectural style, which suggests that any "method" be represented as a modification of a resource. However, DMTF experience with a REST like modeling style has shown that avoiding the use of methods is not always possible or convenient. For this reason, CIM-RS supports invocation of methods.

- **Layered system:** Layering is inherent to information models that represent the objects of a managed environment because clients only see the modeled representations and are not exposed to the actual objects. CIM-RS defines the protocol and payload representations such that it works with any model, and thus is well suited for implementations that implement a model of the managed environment independently of protocols, and one or more protocols independently of the model. CIM-RS supports the use of HTTP intermediaries (for example, caches and proxy servers).
  This REST constraint is fully satisfied in CIM-RS.

- **Code-On-Demand:** CIM-RS does not directly support exchanging program code between the protocol participants.
  This optional REST constraint is not satisfied.

Beyond that, CIM-RS has the following other characteristics:

- **Model independence:** CIM-RS does not define or prescribe the use of a particular CIM model. However, it does require the use of a CIM model defined using the CIM infrastructure/architecture. This allows reusing the traditional DMTF technology stack and its implementations, with only minimal impact to existing implementations. For details about CIM-RS resources, see clause 4.
- **Opaqueness of resource identifiers:** CIM-RS uses URIs as resource identifiers and defines all but a top-level URI to be opaque to clients. That allows reuse of the URIs supported by existing WBEM protocols without any remapping, as well as the use of new URI formats in the future. It encourages a client style of programming that is more RESTful than when clients parse resource URIs. For details about CIM-RS resource identifiers, see clause 5.
- **Consistency of operations:** Beyond following the REST constraints, the CIM-RS operations are consistent with the generic operations defined in DSP0223. This allows implementing CIM-RS as an additional protocol in existing WBEM infrastructures, causing impact only where it is necessary (that is, at the protocol level), leveraging existing investments. For details about CIM-RS operations, see clause 6.
- **Supports use of new RESTful frameworks:** Because CIM-RS is a RESTful protocol, it supports the use of new RESTful frameworks both on the client side and on the server side, without tying client application development to the use of traditional WBEM clients or CIM client APIs, and without tying server instrumentation development to the use of traditional WBEM servers, such as CIM object managers and providers.

4 Resources in CIM-RS

The REST architectural style allows for the representation of rather static entities such as disk drives, or entities with highly varying state such as a metric measuring the amount of available disk space at a specific point in time, or even entities that dynamically come into existence or cease to exist such as file system mounts.

In CIM-RS, CIM elements such as instances and classes are the resources that can be accessed. Because CIM instances represent managed objects in the managed environment, this provides direct access to these managed objects. For example, a disk drive in the managed environment is accessible as a resource in CIM-RS. CIM classes and CIM qualifier types (that is, the declaration of qualifiers) are also accessible in CIM-RS, but they are not needed for discovery or use of the managed resources. The reason they are accessible is for those clients that have a need to discover the structure of the CIM-RS resources that represent managed objects.

The way managed objects are defined to be represented as resources in CIM-RS, is by using a two-staged mapping approach:

- CIM models describe how managed objects in the managed environment are represented as CIM instances. This part deals with the model and is independent of any protocols.
- CIM-RS describes how CIM instances are represented as CIM-RS resources. This part deals with the protocol and is independent of any models.

This model independence allows CIM-RS to be implemented in an existing WBEM server as an additional protocol, or as a gateway in front of an existing unchanged WBEM server, leveraging the investment in that implementation. Specifically, in WBEM servers supporting a separation of CIMOM and providers, adding support for CIM-RS typically drives change only to the CIMOM but does not drive any change to the providers. On the client side, existing WBEM client infrastructures that provide client applications with a reasonably abstracted API can implement CIM-RS as an additional protocol, shielding existing client applications from the new protocol, should that be needed.

In order to fit well into WBEM infrastructures, CIM-RS supports the same operation semantics as the operations supported at client APIs, provider APIs, and existing WBEM protocols. The generic operations defined in DSP0223 are a common definition of operation semantics for such purposes. The operations of CIM-RS are described independently of DSP0223, but DSP0210 defines a mapping between generic operations and CIM-RS operations. For more details about the operations supported by CIM-RS, see clause 6.

Because CIM-RS is a RESTful protocol, it supports the use of new RESTful frameworks both on the client side and on the server side, without tying client application development to the use of traditional WBEM clients or CIM client APIs, and without tying server instrumentation development to the use of traditional WBEM servers, such as CIMOMs and providers.

This allows CIM-RS to be implemented using typical REST frameworks, without using CIMOM or WBEM infrastructure. In this case, the two-staged mapping approach still works well but requires the reading of more documents in order to understand what to implement, compared to an approach that describes both model and protocol in one document.

Of course, combinations of using new RESTful frameworks and traditional WBEM infrastructure are also possible: A typical scenario would be the use of a new RESTful framework in a client application, with a traditional WBEM server whose CIMOM portion got extended with CIM-RS protocol support.

It is important to understand that the model independence of CIM-RS and the resulting benefits are its main motivation and are a key differentiator to other approaches in DMTF of using REST. The model independence is what positions CIM-RS to be a first class member of the traditional DMTF technology stack, leveraging a large amount of standards defined by DMTF and others (most notably, the CIM architecture/infrastructure, the CIM Schema, and management profiles defined by DMTF and others).

On the downside, the model independence of CIM-RS causes a certain indirection in dealing with the managed objects: CIM-RS resources representing CIM instances of CIM classes can be understood only after understanding the CIM model they implement. The CIM model is defined by a CIM schema and typically also by a number of management profiles that scope and refine the use of the CIM schema to a particular management domain. So the number of documents that must be read before a client application can reasonably be developed against a CIM instrumentation supporting CIM-RS may be quite significant. On the other hand, this is no more complex than developing a client application against a CIM instrumentation supporting other existing WBEM protocols.

Following the REST architectural style, any entity targeted by an operation in the CIM-RS protocol is considered a resource, and the operations are simple operations such as the HTTP methods GET, POST, PUT, and DELETE.

The simplicity of these operations requires details to be "encoded" such as the difference between retrieving a single resource vs. a collection of resources, or retrieving a resource vs. navigating to a related resource, into the resource definitions. This leads to a number of variations of resources.

Note that the real-world entities are not called "resources" in this document. Rather, the standard DMTF terminology is used, where such real-world entities are termed "managed objects", and the real-world is termed the "managed environment". This terminology allows distinguishing resources as represented in the RESTful protocol from the managed objects they correspond to.

Table 1 lists the resource types of CIM-RS.

Table 1 – CIM-RS resource types and what they represent

Resource Type	Represents
Instance	a CIM instance, representing a modeled object in the managed environment
Instance collection	a collection of instances of a particular class
Instance associator collection	a collection of instances associated to a particular instance
Instance reference collection	a collection of association instances referencing a particular instance
Instance collection page	a page of a paged instance collection
Class	a CIM class, representing the type of a CIM instance
Class collection	a collection of classes (top-level classes in a namespace, or subclasses of a class)
Class associator collection	a collection of classes associated to a particular class
Class reference collection	a collection of association classes referencing a particular class
Qualifier type	a CIM qualifier type, representing the declaration of a metadata item
Qualifier type collection	a collection of qualifier types in a particular namespace
Listener indication delivery	a resource within a listener that is used to deliver indications to

Each of these resources can be addressed using a resource identifier; for details see clause 5.

Each of these resources has a defined set of operations; for details on that see clause 6.

Each of these resources has a defined resource representation in each of the supported representation formats; for details on that see clause 7.

CIM-RS supports retrieval of parts of resources. These parts are selected through query parameters in the resource identifier URI addressing the resource. That renders these parts to be separate resources, following the principles in the REST architectural style.

For more details about CIM-RS resources, see DSP0210.

5 Resource identifiers in CIM-RS

The REST architectural style recommends that all addressing information for a resource be in the resource identifier (and not, for example, in the HTTP header). In addition, it recommends that resource identifiers be opaque to clients and clients should not be required to understand the structure (or format) of resource identifiers or be required to assemble any resource identifiers.

CIM-RS generally follows these recommendations. In CIM-RS, resource identifiers are fully represented in URIs, without any need for additional information in HTTP headers or HTTP payload. The structure of URIs in CIM-RS is normatively defined and may be assembled or manipulated by clients. However, the values of key properties of CIM instances are often created by the server side implementation, and are undefined from a client perspective.

The URIs a client typically will need to assemble are those of instance collections to be retrieved. From that point on, the returned instances have their URIs attached and are used as the target resource in subsequent operations.

The main benefit of client-opaque URIs is that servers can use existing URI formats. However, the query parameters are defined by CIM-RS, and so the URI could already not be entirely opaque.

For more details about resource identifiers in CIM-RS, see DSP0210.

6 Operations in CIM-RS

The REST architectural style recommends that the operations on resources are simple and follow certain constraints. Although the use of HTTP is not a requirement for REST, the HTTP methods satisfy these constraints and are therefore a good choice for a RESTful system.

CIM-RS uses the HTTP methods GET, POST, PUT, and DELETE. An operation in CIM-RS is defined as the combination of HTTP method and target resource type (see Table 1).

GET is used to retrieve the targeted resource.

PUT is used for replacing the targeted resource partially or fully. Partial update is performed by issuing the PUT method against a resource identifier that uses query parameters to narrow the original resource to exactly the properties that are intended to be updated. Because the narrowed resource is fully replaced, this approach does not violate the idempotency constraint of the HTTP PUT method.

The alternative to use the HTTP PATCH method for partial update (see RFC5789) was originally chosen in the work of the CIM-RS Incubator but ultimately dismissed in the CIM-RS specifications, because support for the HTTP PATCH method is still limited in the industry at this point.

DELETE is used for removing the targeted resource.

POST is a non-idempotent operation in HTTP that can have many uses. The Request-URI in the header of a POST identifies the resource that will handle the entity enclosed in the message of the request, not necessarily the entity affected by the POST (see RFC2616, page 54). Following this pattern, POST is used in CIM-RS as follows:

- for invoking CIM methods, by targeting an instance or class resource.
- for creating resources, by targeting the collection resource for the type of resource to be created, which acts as a factory resource.
- for delivering indications to a listener.

For more details about operations in CIM-RS, see DSP0210.

7 Data representation in CIM-RS

The REST architectural style promotes late binding between the abstracted resource that is addressed through a resource identifier and the resource representation that is chosen in the interaction between client and server.

CIM-RS follows this by supporting multiple HTTP payload formats that are chosen through HTTP content negotiation.

The set of payload formats supported by CIM-RS is open for future extension, and currently consists of the following:

- JSON, as defined in DSP0211.

A payload format based on XML could be defined in the future.

JSON and XML are considered premier choices for a representation format of RESTful systems, dependent on the REST framework used, and the technical and business environment.

It is important to understand that the entities to be represented in the HTTP payload are not only the resource representations. For example, operations such as method invocation require the representation of input and output data entities (MethodRequest and MethodResponse payload elements) that are not resources (in the sense that they cannot be the target of CIM-RS operations).

Table 2 lists the payload elements defined in CIM-RS. These are the entities that need to be represented in any payload format of CIM-RS.

Table 2 – CIM-RS protocol payload elements

Payload element	Meaning
Instance	Representation of an instance resource; that is, a modeled object in the managed environment
InstanceCollection	A list of representations of instance resources
Class	Representation of a class resource; that is, a class declaration
ClassCollection	A list of representations of class resources
QualifierType	Representation of a qualifier type
QualifierTypeCollection	A list of representations of qualifier types
MethodRequest	The data describing a method invocation request, including input parameters
MethodResponse	The data describing a method invocation response, including its return value and output parameters
IndicationDeliveryRequest	The data describing a request to deliver an indication to a listener
ErrorResponse	The data describing an error response to any request

8 When would a site consider implementing CIM-RS

CIM-RS is implemented in two places: a centralized server and many clients (including event listeners). The server provides access to CIM-RS resources and the client accesses those resources. One of the goals of REST is enabling clients, such as generic HTTP browsers, to discover and access RESTful services without specialized documentation or programming. CIM-RS enables this kind of access, but realistically, such usage would be too granular and awkward for most tasks. More likely, CIM-RS will be used in the background as a web service that performs operations and collects data on IT infrastructure. The code that combines individual REST requests into task-oriented applications can be implemented either on the server side or on the client side.

On the server side, SOAP implementations respond to SOAP calls that are usually transported by HTTP as a layer under the SOAP stack. The RESTful stack is less elaborate because the layer corresponding to the SOAP is eliminated and calls are received directly from the HTTP server. Correspondingly, on the client, in SOAP implementation, calls are made via the SOAP stack and transported by HTTP. In REST, calls are made using native HTTP verbs. REST simplicity comes with a price. The SOAP stack, and the additional specifications that have been written over SOAP add rich functionality that may require extra effort to implement the equivalent in REST.

With the addition of CIM-RS, applications based on objects defined using CIM models can be surfaced via the CIM-RS RESTful protocol. The choice of protocol affects both the server implementation and the client implementation. In theory, the applications that result should be the same, but in practice there may be differences, based on factors such as the statelessness of RESTful and the ease of implementing some interaction patterns.

Many implementations are expected to involve using CIM-RS with existing implementations. The ease of these implementations will be largely dependent on the layering of the architecture of the CIM implementation. Ideally, the implementation of the CIM objects should be crisply separated from the transport mechanism. In that case, the CIM-RS implementation, using appropriate frameworks for interfacing underlying code with HTTP such as JAX-RS, should be straightforward and relatively quick to implement.

Every implementation decision is based on many factors, including:

- The experiences of the personnel involved. A group accustomed to RESTful applications will be better prepared to work with CIM-RS than a SOAP-based implementation. A group not familiar with REST may experience difficulty.
- The environment. For example, implementation behind a corporate firewall will not get as many advantages from a REST implementation as an implementation that spans widely separated architectures involving many firewalls.
- The purpose of the implementation. Some implementations will involve management of massive storms of events. Others will involve long lists of managed objects. Yet others will involve only light traffic, but complex control operations. Every implementation has its own footprint. REST architectures are designed to optimize the capacity, scalability, and upgradability of the server. The archetypical REST implementation is a server that serves an enormous number of clients, for example, a web storefront serving hundreds of thousands of clients simultaneously, but the data exchange with each client is intermittent, granular, and relatively small. This is far different from an enterprise IT management application that manages and correlates data from hundreds of thousands of objects, but only has a handful of clients. RESTful interfaces have proven themselves in the first example, but they have not yet acquired a long track record in the second example. This is not to say that REST, and CIM-RS in particular, is not appropriate for the second example, only that it may present new challenges.

CIM-RS provides an alternative to SOAP-based implementations and allows implementers to take advantages of the unique characteristics of REST. The decision to use CIM-RS should be made in the full context of the experience of the implementers, the environment, and purpose of the implementation.

9 Conclusion

CIM-RS is a set of specifications that describes a rigorous REST interface to resources modeled following the principles of the CIM metamodel. The immediate and obvious consequence of this goal is to provide REST access to management instrumentation based on the more than 1400 pre-existing classes in the DMTF CIM Schema (or in any other schema that follows the CIM metamodel) and in management profiles.

This addresses an important issue in the industry: RESTful interfaces have become an interface of choice for application interaction over the Internet. With rising interest in cloud computing, which largely depends on Internet communications, the importance of REST interfaces is also rising. Consequently, a protocol that promises to give existing applications a RESTful interface with minimal investment is extremely attractive.

CIM-RS provides more than an additional interface to existing CIM-based implementations. The CIM metamodel is a general object oriented modeling approach and can be applied to many modeling challenges. Thus, for any applications built using models that conform to the CIM metamodel, CIM-RS specifies a standards-based RESTful interface that will increase interoperability. Developers can use the CIM-RS specifications as the basis for a design pattern and avoid reinventing a RESTful API for each application, saving time and effort and minimizing testing.

CIM-RS has the potential to become a basic pattern for application communication within the enterprise, between enterprises, and within the cloud. It applies to existing implementations of CIM objects, future CIM object implementations, and implementations of new objects modeled following the CIM metamodel.

ANNEX A

Change log

Version	Date	Description
1.0.0	2012-12-04	
2.0.0	2015-02-19	Published as DMTF Informational

Bibliography

Documents published by standards development organizations

DMTF DSP0004, CIM Infrastructure Specification 2.8,
http://www.dmtf.org/standards/published_documents/DSP0004_2.8.pdf

DMTF DSP0198, *WBEM Glossary 1.0,*
http://www.dmtf.org/standards/published_documents/DSP0198_1.0.pdf

DMTF DSP0210, *CIM-RS Protocol 2.0*,
http://www.dmtf.org/standards/published_documents/DSP0210_2.0.pdf

DMTF DSP0211, *CIM-RS Payload Representation in JSON 2.0*,
http://www.dmtf.org/standards/published_documents/DSP0211_2.0.pdf

DMTF DSP0223, *Generic Operations 2.0,*
http://www.dmtf.org/standards/published_documents/DSP0223_2.0.pdf

IETF RFC2616, *Hypertext Transfer Protocol – HTTP/1.1*, June 1999,
http://tools.ietf.org/html/rfc2616

IETF RFC3986, *Uniform Resource Identifier (URI): Generic Syntax*, January 2005,
http://tools.ietf.org/html/rfc3986

IETF RFC5789, *PATCH Method for HTTP*, March 2010,
http://tools.ietf.org/html/rfc5789

IETF RFC7159, *The JavaScript Object Notation (JSON) Data Interchange Format*, March 2014,
http://tools.ietf.org/html/rfc7159

ISO/IEC 10646:2003, *Information technology -- Universal Multiple-Octet Coded Character Set (UCS)*,
http://standards.iso.org/ittf/PubliclyAvailableStandards/c039921_ISO_IEC_10646_2003(E).zip

The Unicode Consortium, *The Unicode Standard, Version 5.2.0, Annex #15: Unicode Normalization Forms*,
http://www.unicode.org/reports/tr15/

Other documents

R. Fielding, *Architectural Styles and the Design of Network-based Software Architectures*, PhD thesis, University of California, Irvine, 2000,
http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

R. Fielding, *REST APIs must be hypertext driven*, October 2008,
http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven

J. Holzer, *RESTful Web Services and JSON for WBEM Operations*, Master thesis, University of Applied Sciences, Konstanz, Germany, June 2009,
http://mond.htwg-konstanz.de/Abschlussarbeiten/Details.aspx?id=1120

A. Manes, *Rest principle: Separation of representation and resource*, March 2009,
http://apsblog.burtongroup.com/2009/03/rest-principle-separation-of-representation-and-resource.html

L. Richardson and S. Ruby, *RESTful Web Services*, May 2007, O'Reilly, ISBN 978-0-596-52926-0,
http://www.oreilly.de/catalog/9780596529260/