

November 15-18, 2010



Santa Clara Marriott
Santa Clara, CA

CIM-RS

CIM Operations over RESTful Services

DMTF CIM-RS Incubator

Presented by:
Andreas Maier, IBM
maiera@de.ibm.com

Last updated: 2010-09-28

Disclaimer

- The information in this presentation represents a snapshot of work in progress within the DMTF.
- This information is subject to change. The Standard Specifications remain the normative reference for all information.
- For additional information, see the Distributed Management Task Force (DMTF) Web site.

The DMTF was formed to lead the development, adoption and unification of management standards and initiatives for desktop, enterprise and internet environments



Agenda

- **What is "CIM-RS" & Status**
- **CIM-RS operations**
- **CIM-RS payload representation in JSON**
- **Summary**

What is "CIM-RS" ?

- **"CIM-RS"**
 - "CIM Operations over RESTful Services"
 - The name of a DMTF Incubator
 - The name of a WBEM protocol that uses RESTful Services
- **CIM-RS Incubator at DMTF**
 - Launched in 5/2010
 - Incubator leadership team: EMC, IBM, VMware, CA (since 9/2010)
 - Goals:
 - Produce informational specifications that provide the basis for producing standards describing the use of RESTful services between CIM infrastructure components
 - Produce white paper that provides insight into the use of RESTful services for systems management in the context of CIM, including to look at using RESTful approaches that are underway in other DMTF groups
 - After that: Facilitate an industry learning period with the goal of gaining confidence in the directions set by these informational specifications and white paper

Why a REST based protocol for CIM ?

- **Perceived to be low footprint, high performance**
- **Enjoys increasing support in the industry**
- **Client application code can talk REST more directly than current approaches**
 - Current approaches to access CIM servers use CIM client libraries
 - And sometimes "first class object" access layers on top of that
- **REST will be used for systems management**
 - REST APIs for systems management are coming up broadly
 - DMTF wants to step up to this and provide REST based access to CIM servers

Why JSON as a payload encoding ?

- **Perceived to be simple, easy to use and understand**
 - JavaScript standard (ECMA-262, 5th edition) defines JSON on only a few pages
- **Results in a compact payload representation**
- **JSON and RESTful services are often used together**
- **JSON allows avoiding both meta-modeling and schema-per-class**
 - CIM-XML needs only one XML Schema (independent of CIM classes), but suffers from meta-modeling complexities
 - WS-CIM avoids meta-modeling but requires one XML Schema and WSDL per CIM class

Current Status of CIM-RS

- DMTF CIM-RS Incubator produced Work in Progress versions of two informational specifications:
 - **DSP-IS0201 "CIM Operations over RESTful Services"**
 - Defines the use of HTTP methods for accessing CIM servers
 - Implements the operation semantics defined in DSP0223 (Generic Operations)
 - Does not mandate any particular payload representation
 - Payload representation decided at run time via HTTP content negotiation
 - **DSP-IS0202 "CIM-RS Binding to JSON"**
 - Defines a CIM-RS payload representation in JSON (Javascript Object Notation)
- Both have been released as Work in Progress in 10/2010

History of CIM-RS Incubator

- **IBM work before DMTF submission:**
 - Initial work in master thesis "RESTful Web Services and JSON for WBEM Operations" by J.Holzer, 7/2009
 - Verified the concept with an early prototype based on OpenPegasus, 7/2009
 - Technical submission to DMTF, 11/2009
- **DMTF CIM-RS Incubator**
 - Proposed 11/2009, Created 5/2010
 - Home page: <http://members.dmtf.org/apps/org/workgroup/cim-rs/>
 - Significantly reshaped the technical submission
 - Separation into CIM-RS operations and JSON payload representation
 - Incorporated more RESTful-ness into the CIM-RS operations
 - Started using JSON Schema for describing the JSON payload format
 - Work in Progress release in 10/2010

Agenda

- What is "CIM-RS" & Status
- **CIM-RS operations**
 - CIM-RS payload representation in JSON
 - Summary

REST in 5 minutes

- **REST = "REpresentational State Transfer"**
 - Introduced in 2000 in the dissertation of Roy Fielding
- **Inherently based on HTTP operations (e.g. GET, PUT, POST, DELETE)**
 - HTTP semantics provides for a number of benefits, e.g. transparent usage of HTTP caching
- **REST operations deal with "resources"**
 - A resource is addressed using a URI
 - Representation of resources can be determined by HTTP content negotiation (e.g. JSON, XML, ...)
- **Stateless protocol**
 - Server does not need to maintain state for client context or session between operations
 - Resources are still "stateful", i.e. make their state accessible
- **Differences between RESTful services and "stateful" Web Services:**
 - Addressing of resources (REST URI vs. WS-Addressing)
 - Abstraction from transport protocol (REST tied to HTTP, Web Services try to abstract from t.p.)
 - Usage of operation semantics (REST tied to HTTP, Web Services have the WS-* stack of specs)
 - Literature: "RESTful Web Services vs. "Big" Web Services: Making the Right Architectural Decision" by Pautasso, Zimmermann, Leymann (<http://www.jopera.org/files/www2008-restws-pautasso-zimmermann-leymann.pdf>)

Core methods of HTTP 1.1 (RFC2616):

- **OPTIONS**
 - Retrieve options of a server
- **GET**
 - Retrieve representation of a resource
- **POST**
 - Create a resource from a resource representation
- **PUT**
 - Replace a resource from a resource representation
- **DELETE**
 - Delete a resource

Additional HTTP methods:

- **PATCH (RFC5789)**
 - Partial replacement or other modifications of a resource

- **Server resources:**
 - WBEM server
 - WBEM listener
- **CIM object resources:**
 - Namespace
 - Class
 - Instance
 - Qualifier type
- **CIM object collection resources:**
 - Class collection
 - Instance collection
 - Qualifier collection
 - Class associator / reference collection
 - Instance associator / reference collection
- **Special resources:**
 - Instance query
 - Method invocation

CIM-RS Operations

CIM-RS Operation := Combination of HTTP method and REST resource targeted

Target Resource	HTTP Methods	Functions
WBEM server	OPTIONS	Retrieve options of server
Namespace collection	GET	Retrieve set of namespaces
Namespace	GET	Retrieve namespace
Class collection	GET, POST	Retrieve set of classes, create class
Class	GET, PUT, DELETE	Retrieve, replace, delete a class
Class associator collection	GET	Traverse association at class level
Class reference collection	GET	Traverse association at class level
Class method invocation	POST	Invoke method on class
Instance collection	GET, POST	Retrieve set of instances, create instance
Instance	GET, PUT, PATCH, DELETE	Retrieve, replace, modify, delete an instance
Instance associator collection	GET	Traverse association at instance level
Instance reference collection	GET	Traverse association at instance level
Instance method invocation	POST	Invoke method on instance
Qualifier type collection	GET, POST	Retrieve set of qualifier types, create qualifier type
Qualifier type	GET, PUT, DELETE	Retrieve, replace, delete a qualifier type
Instance query	POST	Execute instance level query

CIM-RS Resource URIs

- Only the *namespace collection* resource has URI with well-known format:
 - `http://{host+port}/cimrs/namespaces`
 - `https://{host+port}/cimrs/namespaces`
 - Todo: WBEM server
- All other resources have server-defined URIs
 - URI format is opaque to client, except for:
 - Determining whether a URI is relative or absolute
 - URI normalization (as defined in RFC3986), including making relative URIs absolute
 - Query parameters in the URI
 - URIs for future operations come back to client via named links in payload element
 - Payload representation needs to support notion of named links
 - Names of links are defined for each resource type, for example:
 - "self" – link to the resource itself
 - "class" – link to the creation class of an instance
 - "instances" – link to instance collection of a class
 - This approach enables different resource URI formats, for example:
 - New URI format for CIM-RS
 - WBEM-URI (DSP0207) extended by CIM-RS specific resource types

Example for a new CIM-RS URI Format

This URI format is just an example; all URI formats are undefined, except for top-level resources (**bold**)

Target Resource	Example URI format
WBEM server	http://{host+port}
Namespace coll.	http://{host+port}/cimrs/namespaces
Namespace	http://{host+port}/cimrs/namespaces/{namespaceName}
Class coll.	http://{host+port}/cimrs/namespaces/{namespaceName}/classes
Class	http://{host+port}/cimrs/namespaces/{namespaceName}/classes/{className}
Class ass. coll.	http://{host+port}/cimrs/namespaces/{namespaceName}/classes/{className}/associators
Class ref. coll.	http://{host+port}/cimrs/namespaces/{namespaceName}/classes/{className}/references
Class method invoc.	http://{host+port}/cimrs/namespaces/{namespaceName}/classes/{className}/methodinvocation
Instance collection	http://{host+port}/cimrs/namespaces/{namespaceName}/classes/{className}/instances
Instance	http://{host+port}/cimrs/namespaces/{namespaceName}/classes/{className}/instances/{keylist}
Instance ass. coll.	http://{host+port}/cimrs/namespaces/{namespaceName}/classes/{className}/instances/{keylist}/associators
Instance ref. coll.	http://{host+port}/cimrs/namespaces/{namespaceName}/classes/{className}/instances/{keylist}/references
Instance meth. invoc.	http://{host+port}/cimrs/namespaces/{namespaceName}/classes/{className}/instances/{keylist}/methodinvocation
Qualifier type coll.	http://{host+port}/cimrs/namespaces/{namespaceName}/qualifiers
Qualifier type	http://{host+port}/cimrs/namespaces/{namespaceName}/qualifiers/{qualifierName}
Instance query	http://{host+port}/cimrs/namespaces/{namespaceName}/instancequery

Agenda

- What is "CIM-RS" & Status
- CIM-RS operations
- **CIM-RS payload representation in JSON**
- Summary

JSON in 2 minutes

- Empty Object:
`{ }`
- Empty Array:
`[]`
- Object members are properties (key-value pairs):
`{ "Caption": "Hello World" }`
- Value types are Array, Object, String, Number, Boolean:
`{ "AnArray": [],
 "AnObject": { },
 "AString": "Hello World",
 "ANumber": 1,
 "ABoolean": true }`
- Objects and Arrays can be nested:
`[{ "One": 1 }, { "Two": 2 }]`

- **IETF Draft RFC**
 - <http://tools.ietf.org/html/draft-zyp-json-schema-02>
 - Draft 03 about to be published by IETF
- **Uses JSON as a description language**
- **"Core schema"**
 - Specifies structure of JSON elements
 - Some ability to specify constraints between JSON elements
 - e.g. usage of one property requires usage of another property
 - **Used in CIM-RS JSON**
- **"Hyper-schema"**
 - Specifies HTTP methods for JSON payload elements
 - Not used in CIM-RS JSON since the HTTP methods need to be representation-independent

Example: CIM instance in JSON

```
{
  "links": {
    "self": { "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/
              classes/CIM_RegisteredProfile/instances/DMTF%3AFan%3A1.1.0" },
    "class": { "href": "../.." },
    "methodinvocation": { "href": "methodinvocation" },
    "associators": { "href": "associators" },
    "references": { "href": "references" }
  },
  "class": "CIM_RegisteredProfile",
  "properties": {
    "InstanceID": "DMTF:Fan:1.1.0",
    "RegisteredName": "Fan",
    "RegisteredOrganization": 2,
    "RegisteredVersion": "1.1.0",
    . . . # more properties
  }
}
```

- CIM instance properties are JSON, direct client access by property name
- Link names are JSON properties, direct client access by link name
 - URI structure follows the example described earlier
 - "self" link is absolute to make the result self-contained
 - Other links are relative to "self" link (and meaning of ".." segments is defined by RFC3986)

Example: CIM class collection in JSON

```
{
  "links": {
    "self": { "href":
"http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/classes" },
    "namespace": { "href": ".." }
  },
  "classes": {
    "CIM_SubClass1OfExample": {
      "links": {
        "self": { "href": "http://acme.com:5988/cimrs/namespaces/root%2Fcimv2/
classes/CIM_SubClass1OfExample" },
        "namespace": { "href": "../.." },
        "instances": { "href": "instances" },
        "methodinvocation": { "href": "methodinvocation" },
        "associators": { "href": "associators" },
        "references": { "href": "references" }
      },
      "superclass": "CIM_Example",
      "qualifiers": { . . . },
      "properties": { . . . }
      "methods": { . . . }
    },
    . . . # other classes
  }
}
```

- Class collection is a resource and has links
- Each class in the collection is a resource and has links
- Classes in the collection can be accessed directly by class name

Agenda

- What is "CIM-RS" & Status
 - CIM-RS operations
 - CIM-RS payload representation in JSON
- **Summary**

Summary: Work results and next steps

- **Work results:**

- Work in Progress versions of CIM-RS informational specifications released in 10/2010
- Note: As Work-in-Progress, these documents are subject to change

- **Next steps:**

- Informational status of CIM-RS specs in 12/2010
- White paper in 1Q2011
- Facilitate industry learning period
- Start standardization process

Some Take-Aways on CIM-RS ...

- **Schema and profile independence**
 - No further profile "mapping specs" are required beyond the CIM-RS specs
- **Multiple payload representations**
 - JSON representation is defined
 - XML representation can be added
- **No meta-modeling and no per-class JSON schema needed**
 - DMTF CIM Schema release process does NOT need to add a new JSON format
- **Operation semantics consistent with Generic Operations (DSP0223)**
 - Allows adding protocol handlers to existing CIM infrastructure components
 - Allows using existing providers unchanged
 - Allows development of protocol gateways
 - e.g. CIM-RS gateway sitting in front of a traditional CIMOM