Document Identifier: DSP2041

Date: 2015-02-25

Version: 1.0.0a

Cloud Infrastructure Management Interface (CIMI) Extensions

Information for Work-in-Progress version:

IMPORTANT: This document is not a standard. It does not necessarily reflect the views of the DMTF or its members. Because this document is a Work in Progress, this document may still change, perhaps profoundly and without notice. This document is available for public review and comment until superseded.

Provide any comments through the DMTF Feedback Portal: http://www.dmtf.org/standards/feedback

Supersedes: None

Document Type: Informational

Document Class: Informative

Document Status: Work in Progress

Document Language: en-US

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted.

Implementation of certain elements of this standard or proposed standard may be subject to third party patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third party patent right, owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third party patent rights, or for such party's reliance on the standard or incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

For information about patents held by third-parties which have notified the DMTF that, in their opinion, such patent may relate to or impact implementations of DMTF standards, visit http://www.dmtf.org/about/policies/disclosures.php.

CONTENTS

Introduction.......... 4
Typographical conventions.......... 4
1 Scope.......... 5
2 Normative references.......... 5
3 Terms and definitions.......... 5
4 Symbols and abbreviated terms.......... 6
5 CIMI Extensions.......... 7
5.1 Purpose.......... 7
5.2 Content.......... 7
5.3 Format.......... 7
6 POSIX compliant scheduling support.......... 7
6.1 Extension design and rationale.......... 7
6.1.1 Specification being extended.......... 7
6.1.2 Purpose.......... 7
6.1.3 General design and rationale.......... 8
6.1.4 Binding with existing CIMI resources and operations.......... 8
6.1.5 Examples.......... 9
6.2 Extending and profiling existing CIMI features.......... 10
6.2.1 Update to the capability URIs.......... 10
6.2.2 Update to the Cloud Entry Point:.......... 10
6.2.3 Updates to the definition of the Job Resource (section 5.17.1):.......... 11
6.2.4 Updates to the Job Attribute table (section 4.6.1):.......... 12
6.2.5 Updates on Operation Resources on Jobs.......... 16
6.3 Additional resources and features.......... 17
6.3.1 Addition of a JobTemplate Resource:.......... 17
6.3.2 Update on JobCollection Resource (4.6.3):.......... 21
6.3.3 Addition of a JobTemplate Collection Resource:.......... 21
6.4 Implementation considerations.......... 22
ANNEX A (normative) CIMI Extension Document Template.......... 24
ANNEX B (informative) Change log.......... 25
Bibliography.......... 26

Tables

Table 1 – Cloud Entry Point attributes.......... 11
Table 2 – Job attributes.......... 12
Table 3 – JobTemplate attributes.......... 18

Introduction

Cloud Infrastructure Management Interface (CIMI) Extensions (DSP2041) was prepared by the Cloud Management Working Group and approved by the Process and Incubation Committee. This document defines the process governing the development and publication of extensions and profiles of the Cloud Infrastructure Management Interface (CIMI) Specification. It is targeted to all DMTF members and external authors of these publications as a framework to facilitate the evolution of CIMI.

The defined process outlined in this document includes:

- Draft a CIMI Extension or profile
- Submit the draft to the Cloud Management Working group
- Obtain an Information DSP document identifier
- Review and modify the document, conforming to appropriate format
- Vote to approve the publication of the document
- Follow DMTF process for publication of informational documents

Typographical conventions

When the extension reproduces text from the referenced CIMI version while defining changes to the existing text, these changes and their scope are indicated as follows:

For an addition:

The new text is introduced by: " **[EXT-ADD**: " and terminated by " **]** ' . The additional text is also color-coded in contrast with surrounding (black) existing text.

For a deletion:

The deleted text is introduced by: " **[EXT-DEL**: " and terminated by " **]** ' . The text to be deleted is also color-coded in contrast with surrounding (black) existing text.

For a substitution:

For a short text: the new text is introduced by: " **[EXT-SUB**: " <new text> " **/** " <old text> and terminated by " **]** ' . The new text is also color-coded in contrast with surrounding (black) existing text, while the old text is not.

For a long text: the new text is introduced by: " **[EXT-SUB**: " <new text> and terminated by " **].** The extent of the deleted old text is indicated otherwise, next to the "EXT-SUB" keyword, e.g. " **[EXT-SUB (replacing similar text)**: " when the extent of the replacement is not ambiguous.

For an update of lesser importance the exact wording of which remains to be decided, or may be optional:

The text to be considered for update is introduced by: " **[EXT-UPDATE**: " < text-to-be-considered-for-update> and terminated by " **]** ' .

Cloud Infrastructure Management Interface (CIMI) Extensions

1 Scope

This document defines a DMTF process governing the creation and publication of documents that evolve the CIMI Specification by either extending its functionality or requiring implementations of specific subsets of optional functionality.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated or versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies. For references without a date or version, the latest published edition of the referenced document (including any corrigenda or DMTF update versions) applies.

DMTF DSP0263, *Cloud Infrastructure Management Interface (CIMI) Model and RESTful HTTP-based Protocol 1.1*, http://www.dmtf.org/sites/default/files/standards/documents/DSP0263_1.1.0.pdf.

3 Terms and definitions

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

The terms "shall" ("required"), "shall not", "should" ("recommended"), "should not" ("not recommended"), "may", "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Annex H. The terms in parentheses are alternatives for the preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that ISO/IEC Directives, Part 2, Annex H specifies additional alternatives. Occurrences of such additional alternatives shall be interpreted in their normal English meaning.

The terms "clause", "subclause", "paragraph", and "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 5.

The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 3. In this document, clauses, subclauses, or annexes labeled "(informative)" do not contain normative content. Notes and examples are always informative elements.

The terms defined in DSP0004, DSP0223, and DSP1001 apply to this document. The following additional terms are used in this document.

2.1

CIMI Extension

A document that adds functionality to a specific version of the CIMI Specification. CIMI Extensions may be created by DMTF members or external authors. The document is written as a series of delta text modifications to the CIMI Specification and may reference other CIMI Extensions for inclusion.

2.2

CIMI Profile

A document that profiles a specific version of the CIMI Specification, making a specific set of optional functionality required for purposes of increased interoperability.

4 Symbols and abbreviated terms

The abbreviations defined in DSP0004, DSP0223, and DSP1001 apply to this document. The following additional abbreviations are used in this document.

3.1

DMTF

Distributed Management Task Force

3.2

DSP

DMTF Specification

5 CIMI Extensions

5.1 Purpose

The purpose of publishing informative CIMI Extensions is to facilitate implementations containing vendor specific functionalities that could to be added at the CIMI Specification in an experimental manner. Oftentimes, a specific functionality will first be implemented by a single vendor, and then, after the business case is proven, imitated by other vendors. Other times, the functionality may not prove valuable to customers and thus not be imitated by other vendors. If this functionality were to be added to a version of the CIMI Specification directly, there would be a number of one-off features in the standard, increasing complexity without adding any interoperability.

5.2 Content

The content of the CIMI Extension document is primarily a set of deltas to a specific version of the CIMI specification. An implementer would use the extension in conjunction with the CIMI Specification to implement specified functionalities. CIMI Extension documents are not meant to be used standalone.

In addition, the CIMI Extension document is expected to provide a justification for implementers to incorporate the extension functionalities into their CIMI implementation. This is documented in the Scope, Introduction and Use Cases sections of the document.

5.3 Format

The document format for CIMI Extension documents shall follow the template as documented by ANNEX A.

6 POSIX compliant scheduling support

POSIX.1-2008 defines a standard operating system interface and environment, including a command interpreter (or “shell”), and common utility programs to support applications portability at the source code level.

6.1 Extension design and rationale

6.1.1 Specification being extended

These modifications are carried against version 0.125 of the core CIMI Specification.

6.1.2 Purpose

The objective of this extension is to leverage scheduling features already widely and natively available in POSIX-compliant operating systems.

It is expected that a POSIX-compliant operating system (OS) will be available at many Provider sites, i.e., as an OS that the Provider is using for its own administration needs. Even without making this assumption, it is expected that compliant virtual machines running Linux will be available to the Consumer (e.g., a MachineTemplate provided by the Provider) and configurable so that they can process POSIX commands issued in the context of this CIMI Extension.

This extension defines a binding between a CIMI Job resource and scheduling utilities “at” and “crontab” already supported natively by POSIX.1-2008 compliant operating systems.

6.1.3 General design and rationale

This extension uses the Job CIMI Resource as a container for POSIX-compliant commands, focusing in this case on the utilities supporting scheduling functions “at” and “crontab”. The Job as a container represents a run-time gateway to and from the external scheduling engine. A two-way binding mechanism is described that enables:

1) the invocation of a POSIX-compliant shell and scheduling utilities from the CIMI Provider implementation.

2) the call-back from the “scheduler” engine (here the POSIX-compliant operating system and its scheduling utilities) to the CIMI Provider implementation, for execution of the scheduled operation.

The rationale for this design is as follows:

a) By using a scheduling capability residing on the Provider side instead of the Consumer side, the Consumer operations are not dependent on network reliability (i.e., the network between Consumer and Provider does not have to be up and running at the time a scheduled operation needs to be executed.) This means increased reliability for the Consumer.

b) The responsibility of the proper scheduling falls on the Provider side, which is appealing to the Consumer who does not have to support and maintain a scheduling capability.

c) By being aware of the scheduling requests that are registered by Consumers ahead of time, the Provider can optimize its operations.

The process of scheduling an operation is two-step and follows the typical template-based Resource creation in CIMI. Yet – as template-based creation – it can be collapsed in one step. A new Resource – the JobTemplate – is defined as part of this extension:

- Step 1: or **registration** step - creation of a JobTemplate Resource that contains the scheduled operation. This creation is expected to be done by the Consumer, although predefined JobTemplates may pre-exist in a CEP.
- Step 2: or actual **scheduling** step - creation of a Job Resource based on the above template. This operation initiated by the Consumer amounts to executing the scheduling command registered in the JobTemplate. This scheduling command may, for example, request that a Machine be deleted on the coming Friday night.

Like for any Resource, a JobTemplate can be passed by value at the time the scheduling Job is created, merging both steps in one.

6.1.4 Binding with existing CIMI resources and operations

The following relies on the new JobTemplate Resource, defined in clause 6.4.

The JobTemplate `command` attribute is profiled as follow, for a POSIX.1-2008 invocation:

– Value of the `cmdtype` key: “posix”

– Value of the `cmd` key: a POSIX.1-2008 shell command or scheduling utility invocation such as “at” or “crontab” (see below).

<u>Additional scripts</u>:

`cimiexec`: This script invokes a CIMI Job and shall be a POSIX.1-2008 shell script. It supports the binding from the external scheduling engine to the Provider CIMI implementation. It shall function as follows:

- The script takes as first argument the name (a string) of a JobTemplate from which a Job will be created and invoked. It may define additional arguments.
- The script sets the following environment variables, to be mapped in the `output` attribute of the Job invoking the script:
  - `response`: set with the standard output of the command.
  - `returnCode`: set with the status code of the command execution.

For example, the command attribute of a JobTemplate may contain:

```
“cmd”: “cimiexec createmymachine”
```

A Job created from such a JobTemplate will invoke the script `cimiexec` as done at a shell command prompt, assuming the 1st argument is a string (`>cimiexec “createmymachine”`). The argument is the name of another JobTemplate (`createmymachine`) that the `cimiexec` implementation shall resolve into the related JobTemplate URI. The script in turn acts as a proxy Consumer and creates a Job from the `createmymachine` JobTemplate. For example, the script could be implemented as a “`curl`” command generating the Job creation request.

6.1.5 Examples

These examples consists of:

- Scheduling of a CIMI operation at a given date, using the “at” POSIX utility.
- Executing a CIMI operation on a recurring schedule, using the “crontab” POSIX utility.

<u>Example 1</u>: scheduling at a particular date

A JobTemplate containing the following command:

```
{ “name”: “myjobtemplate”,
…
 “command”: {
“cmdtype” : “posix”,
“cmd” : “ (echo cimiexec \”machinecleanup\”) | at 2pm next week “
},
…
}
```

A Job created from the `myjobtemplate` JobTemplate, executes at 2:00 PM the week after the current week. The cimiexec script is in turn executing a `machinecleanup` CIMI Job (generated from JobTemplate of name `machinecleanup`).

Example 2: recurring execution of a command

A JobTemplate is defined as containing the following command:

```
{ “name”: “myjobtemplate”,
…
 “command”: {
“cmdtype” : “posix”,
“cmd” : “ (echo 0 2 * * 6 cimiexec \”machinecleanup \”) | crontab“
},
…
}
```

When a Job is created from the `myjobtemplate` JobTemplate , it will execute every Saturday at 2am the “`cimiexec machinecleanup`” command in `cmd` that is in turn executing the CIMI Job (from JobTemplate of name `machinecleanup`) periodically.

6.2 Extending and profiling existing CIMI features

These modifications are carried against version 0.125 of the core CIMI specification.

6.2.1 Update to the capability URIs

The following capability is added:

Resource Name	Capability Name	Description
Job	JobCommandTypes	If set, the value of this capability contains a list of command line interface types that are supported by the Provider **[EXT-ADD**: If set, the capability shall contain at least the “CIMI” value that identifies the CIMI-REST command line. If the properties are set, the Provider allows a Consumer to create Job resources.**]**

6.2.2 Update to the Cloud Entry Point:

The following collection is added:

Table 1 – Cloud Entry Point attributes

Name	CloudEntryPoint	
Type URI	http://www.dmf.org/cimi/CloudEntryPoint	
Attribute	**Type**	**Description**
jobTemplates	*collection [JobTemplate]*	A reference to the `JobTemplateCollection` of this Cloud Entry Point. **Constraints:** **Provider:** support optional; mutable **Consumer:** support optional; read-only

Serialization addition:

JSON:

```
"jobTemplates": { "href": string }, ?
```

XML:

```
<jobTemplates href="xs:anyURI"/> ?
```

6.2.3 Updates to the definition of the Job Resource (section 5.17.1):

The content of this section narrative is now as follows:

This Resource represents a process (i.e., a sequence of one or more operations directed to accomplish a specific goal) that is performed by the Provider.

If a Provider supports exposing `Job` Resources to Consumers, each request from a Consumer that would result in a change to the environment shall result in a `Job` Resource being created and an absolute URI reference to that `Job` Resource shall be made available to the requesting Consumer. Providers may create additional `Job` Resources for Provider initiated operations if the Provider chooses to expose these Jobs to Consumers.

[EXT-ADD:

A Provider may also allow Consumers to directly create `Jobs` as advertised by the JobCommandTypes capability.

As a consequence `Jobs` can be created by either Consumer or Producer and may provide different levels of control to the Consumer:

- `Jobs` directly created by Consumers involve a POST request to the “jobs” collection, and use a `JobTemplate` Resource. These can be deleted or updated (e.g., stopped) by Consumers.
- `Jobs` created on the initiative of the Provider i.e., without an explicit create `Job` request from the Consumer typically do not involve a `JobTemplate` and cannot be deleted or updated by Consumers. Such Jobs are intended to be exclusively managed by the Provider and have only an informative role to the Consumer. **]**

If a `Job` is not completed successfully (e.g., it is in the FAILED or STOPPED state), this specification does not place any requirements on the Provider to ensure that the affected Resources are left in certain states. Based on the environmental conditions at that time, the Provider might choose to "undo" any impact of the operation; simply halt processing; attempt some kind of "cleanup" action; or choose to do something else. **[EXT-UPDATE:** However, Providers shall list all Resources impacted by the Job in the "affectedResources" attribute, thus allowing Consumers an opportunity to examine the state of each Resource themselves. However, Providers shall list all Resources impacted by the Job in the "affectedResources" attribute, thus allowing Consumers an opportunity to examine the state of each Resource themselves.] In cases where a Resource has been deleted, references to that Resource shall not appear in the "affectedResources" attribute.

The `Job` Resource allows for nesting of `Jobs`. The determination of when a single operation is converted into multiple nested `Jobs` is out of scope of this specification. However, if there are nested `Jobs`, the top most Job Resource shall report the overall status of all `Jobs` and shall only be in a "SUCCESS" state if all nested `Jobs` are also in "SUCCESS" state. If nested `Jobs` are created, there is no requirement for the top-most `Job` Resource to reference all affected Resources in its "affectedResources" attribute. The Consumer needs to traverse the entire set of nested `Jobs` to determine the complete list of Resources impacted by the `Jobs`.

6.2.4 Updates to the Job Attribute table (section 4.6.1):

The content of Table 2 *is now as follows*:

Table 2 – Job attributes

Name	Job	
Type URI		
Attribute	**Type**	**Description**
state	*String*	The state of the process associated with this operation. Allowable values include: **QUEUED**: Indicates that the operation has not yet begun processing.**[EXT-ADD:** This is the default state at creation time. The Provider should start running a Job in that state as early as possible.**]** **RUNNING**: Indicates that the operation is still being executed. **FAILED**: Indicates that the operation failed to be completed successfully. **SUCCESS**: Indicates that the operation was successfully completed. **STOPPING**: Indicates that the operation is in the process of being stopped. **STOPPED**: Indicates that the operation was stopped before completion. The operations that result in transitions to the above defined states are defined in DSP0263 clause **Error! Reference source not found.** **Constraints:** **Provider:** support mandatory; mutable **Consumer:** support **.[EXT-SUB:** optional**/**mandatory**]**; read-only

Name	Job	
Type URI		
Attribute	**Type**	**Description**
targetResource	*Ref*	A reference to the top-level Resource upon which the operation is being performed. Typically, this Resource would be the Resource on which the operation **[EXT-ADD:** in the command attribute**]** was invoked. Note that if an "add" Job is executed against a "Collection" Resource (e.g., MachineCollection), the targetResource attribute shall reference the Collection Resource as that is the Resource on which the operation was performed. **[EXT-DEL:** Additionally, the newly created Resource shall appear in the "affectedResources" attribute.**]** **[EXT-ADD:** The attribute can be empty. This is the case when the Job represents more than one operation that concern different Resources. This may happen either when the Job is a grouping construct for other jobs (through the nestedJobs attribute) or has a sequence of several commands associated with it.**]** **Constraints:** **Provider:** support mandatory; immutable **Consumer:** support optional; read-only
affectedResources	*ref[]*	A list of references to Resources that have been impacted by this Job. Note that this list shall always contain the "targetResource" reference. Array item name: affectedResource **Constraints**: **Provider**: support mandatory; mutable **Consumer**: support mandatory; read-only
action	*URI*	URI that indicates the type of action being performed. **Constraints:** **Provider:** support mandatory; immutable **Consumer:** support mandatory; read-only
[EXT-ADD (*this row*)]: command	*Map*	A command to be executed by this Job. See the JobTemplate definition for the format of CIMI operations and of external commands. When the Job has been created from a JobTemplate, this command or an equivalent form of it is copied from the same command attribute in the JobTemplate. **Constraints:** **Provider:** support mandatory; mutable **Consumer:** support optional; read-write

Name	Job	
Type URI		
Attribute	**Type**	**Description**
[EXT-ADD (*this row*)]: output	*Map*	A command outcome matching the command in the command attribute. For a CIMI command, the following item names are defined that should be present in the command output: • **response** (string): If applicable, contains the response of the matching command, i.e., the HTTP body that has been sent back if the Job was created to track a Consumer request, or would have been sent back to the Consumer, in case the commands of the Job come from a JobTemplate and not from the Consumer. • **returnCode** (integer): The operation return code. Shall be present whenever the state is either FAILED or SUCCESS. The value is specific to the implementation. Values in the range of 0 to 9999 are reserved for use by this specification. This code is the HTTP return code of the HTTP response that has been sent back if the Job was created to track a Consumer request, • **location** (URI): If the command creates a new resource, returns its URI. The content is same as the location header in the HTTP response if the Job was created to track a Consumer request. • **affectedResource** (ref): A reference to the main resource if any. Several items with this name may be present. The targetResource shall be duplicated here as an affectedResource. For commands defined by a CIMI Extension, the map items to be present if any are defined in the extension. **Constraints:** **Provider:** support mandatory; mutable **Consumer:** support optional; read-only
returnCode	*Integer*	The operation return code. The specific value is specific to the implementation. Values in the range of 0 to 9999 are reserved for use by this specification. **Constraints:** **Provider:** support mandatory; mutable **Consumer:** support mandatory; read-only
progress	*Integer*	An integer value in the range 0 … 100 that indicates the progress of this Job. This value shall be 100 if the Job is no longer executing, regardless of the outcome. **Constraints:** **Provider:** support mandatory; mutable **Consumer:** support mandatory; read-only

Name	Job	
Type URI		
Attribute	**Type**	**Description**
statusMessage	*String*	A human-readable string that provides information about the operation. It is used to further qualify or provide additional information about the current status of the operation. For example, this attribute may indicate the reason why the operation failed, or whether the operation was cancelled by the Consumer or the Provider. **Constraints:** **Provider:** support mandatory; mutable **Consumer:** support mandatory; read-only
timeOfStatusChange	*dateTime*	A timestamp indicating the last time that the status of the operation changed. **Constraints:** **Provider:** support mandatory; mutable **Consumer:** support mandatory; read-only
parentJob	*Ref*	A reference to the Job of which this Resource is a subordinate. **Constraints:** **Provider:** support mandatory; immutable **Consumer:** support mandatory; read-only
nestedJobs	*ref[]*	An array of references to a set of subordinate Job Resources. Array item name: nestedJob **Constraints:** **Provider:** support mandatory; mutable **Consumer:** support mandatory; **[EXT-SUB:** read-write**/**read-only**]**

Serialization addition:

[EXT-SUB (to existing similar text):

JSON:

```
  "command":
    {
      ( (“POST”|”GET”|”DELETE”|”PUT”|”PATCH”) : string ,
        <httpHeadername> : string , +
        “body” : any ) |
      ( “cmdtype” : string ,
        “cmd” : string  )
    },

  "output":
    {  “response” : string , ?
      “returnCode” : number , ?
```

```
       “location” : string , ?
       “affectedResource” : string * },
```

XML:

```
  <command>
( <httpMethod name=(“POST”|”GET”|”DELETE”|”PUT”|”PATCH”)> xs:anyURI
</httpMethod>
  <httpHeader name=”xs:string”> xs:string </httpHeader>
  <body> <xs:any>* </body> ) |
( <cmdtype name=”xs:string” />
  <cmd> xs:string </cmd> )
  </command>
  <output>
       <response> xs:string </response> ?
       <returnCode> xs:integer </returnCode> ?
       <location> xs:anyURI </location> ?
       <affectedResource> xs:string </affectedResource> *
</output>
```

]

6.2.5 Updates on Operation Resources on Jobs

The following updates are made on Section 5.17.1.1:

6.2.5.1 Operations Resource

This Resource supports the Read, Update, and Delete operations. Deleting a `Job` that is in the "RUNNING" state shall be the equivalent of first stopping the `Job` and then deleting it. A request to delete a running `Job` that does not support the "stop" action shall fail.

The following custom operations are also defined:

stop

/link@rel: `http://schemas.dmtf.org/cimi/1/action/stop`

This operation shall stop a `Job`.

Input parameters: None.

Output parameters: None.

During the processing of this operation, the `Job` shall be in the "STOPPING" state.

Upon successful completion of this operation, the `Job` shall be in the "STOPPED" state. **[EXT-ADD:** If the Job was initially in a state other than QUEUED or RUNNING state, the operation is ineffective and the Provider should respond with a 4xx HTTP error code. Every job involved in the execution - that is the jobs referenced in the attribute `nestedJobs` – shall also be moved to STOPPED or STOPPING only if they were QUEUED or RUNNING. **]**

HTTP protocol

To stop a `Job`, a POST is sent to the "http://schemas.dmtf.org/cimi/1/action/stop" URI of the `Job` where the HTTP request body shall be as described below.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/1/Action",
  "action": "http://schemas.dmtf.org/cimi/1/action/stop",
  "properties": { string: string, + } ?
  ...
}
```

XML media type: application/xml

XML serialization

```
<Action xmlns="http://schemas.dmtf.org/cimi/1">
  <action> http://schemas.dmtf.org/cimi/1/action/stop </action>
  <property key="xs:string"> xs:string </property> *
  <xs:any>*
</Action>
```

Upon successful processing of the request, the HTTP response body may be empty.

6.3 Additional resources and features

6.3.1 Addition of a JobTemplate Resource:

6.3.1.1 JobTemplate Resource

This Resource represents a Job definition that can be used and reused by a Consumer to create and execute `Jobs`.

For any CIMI operation that a Consumer can send (POST) directly to a Resource, the Consumer could instead create a `JobTemplate` that embeds this operation, and then create a `Job` from this template at the time of execution. The Consumer could also directly create such a `Job` by passing the `JobTemplate` data by value. Such an execution is always executed asynchronously, from the Consumer's viewpoint (i.e., a Consumer-driven `Job` creation always returns a `Job` handle, not a response to the embedded operation).

Here is an example of the body of a `Job` creation request (to be included in a POST to the “jobs” CEP collection) using a `JobTemplate` by value (in JSON). This `Job` is starting a Machine (`machine1`). A single request from the Consumer is sufficient to create and start such a `Job`:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/1/JobCreate",
  "name": "starterJob12",
  "description": "A Job that starts Machine 1 ",
  "JobTemplate": {
    “command”: {
      “POST”: “/machines/machine1“,
      “Content-Type”: “application/json” ,
       “body”:{ "resourceURI": "http://schemas.dmtf.org/cimi/1/Action",
                "action": http://schemas.dmtf.org/cimi/1/action/start }
      }
  }
}
```

A command in the `command` attribute may also be an external command – i.e., not a CIMI-defined operation. An external operation is not part of a CIMI implementation, but is delegated instead to a third-party processor or tool, for which a CIMI Extension must be defined (in a separate document).

A CIMI Extension is useful to leverage advanced functions not defined in CIMI, such as scheduling or scaling-out. For that purpose, this specification is only defining how to wrap such external commands in a `Job` so that a CIMI Provider implementation knows how to extract it and also how such commands may bind back to a CIMI operation or Resource. In such a case, the capability `JobCommandTypes` advertises the names of the supported command languages.

A `JobTemplate` may refer to nested `JobTemplates.` When a `Job` is created from such a `JobTemplate`, nested `Jobs` are also created from the nested `JobTemplates`. These `Jobs` are assumed to be independent of each other and are executed without any particular order, in a mode that remains at the discretion of the Provider – e.g., they could be executed in parallel.

Table 3 describes the `JobTemplate` attributes.

Table 3 – JobTemplate attributes

Name	JobTemplate	
Type URI	http://schemas.dmtf.org/cimi/1/JobTemplate	
Attribute	**Type**	**Description**

Command	*Map*	A command to be executed by Jobs created from this template.
		For a CIMI operation (default), the `command` shall be a map of the form:
		{ <httpMethod> : *URI* ,
		<httpHeaderName> : *string* , *
		“`body`” : *any* }
		Example (in JSON style) for a Machine creation operation:
		`{`
		`“POST” : “/machines” ,`
		`“Content-Type” : “application/json” ,`
		`“body” : {`
		`  "resourceURI":`
		`"http://schemas.dmtf.org/cimi/1/MachineCreate",`
		`  "name": "myMachine1",`
		`  "description": "My very first machine",`
		`  "machineTemplate": { "href":`
		`“http://example.com/machineTemplates/72000” }`
		`}`
		`}`
		Support of external commands

Name	JobTemplate	
Type URI	http://schemas.dmtf.org/cimi/1/JobTemplate	
Attribute	**Type**	**Description**
		A Provider may support creating Jobs executing external commands other than CIMI operations, e.g., a Linux command or other script invocation. In such cases, the command shall be a map of the form: { "cmdtype" : <type of the command line interface>, "cmd" : <native command> } cmdtype identifies the type of the command – or command line interface - used for this command, e.g., "posix". cmd contains the actual command to be executed, or a representation of it that is described in a CIMI Extension. The details of a standard binding to the command type identified in "cmdtype" (e.g., "posix") – are defined in a CIMI Extension outside the scope of this specification. **Constraints:** **Provider:** support mandatory; mutable **Consumer:** support optional; read-write
nestedJobTemplates	*ref[]*	An array of references to a set of subordinate JobTemplate Resources. Array item name: nestedJobTemplate **Constraints:** **Provider:** support mandatory; mutable **Consumer:** support mandatory; read-write

When implementing or using Job, Providers and Consumers shall adhere to the syntax and semantics of its attributes as described in Table 3 as well as in the tables describing referred Resources or related Collections. Both Consumer and Provider shall serialize this Resource as described below. The following pseudo-schemas describe the serialization of the Resource in both JSON and XML.

JSON media type: application/json

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/1/JobTemplate",
  "id": string,
  "name": string, ?
  "description": string, ?
  "created": string, ?
  "updated": string, ?
  "properties": { string: string, + }, ?
  "command":
    {
       ( (“POST”|”GET”|”DELETE”|”PUT”|”PATCH”) : string ,
         <httpHeaderName> : string , +
         “body” : any ) |
       ( “cmdtype” : string ,
         “cmd” : string  )
    },
  "nestedJobTemplates": [
    { "href": string }, +
  ], ?
  "operations": [
    { "rel": "edit", "href": string }, ?
    { "rel": "delete", "href": string }, ?
  ] ?
  ...
}
```

XML media type: application/xml

XML serialization:

```
<JobTemplate xmlns="http://schemas.dmtf.org/cimi/1">
  <id> xs:anyURI </id>
  <name> xs:string </name> ?
  <description> xs:string </description> ?
  <created> xs:dateTime </created> ?
  <updated> xs:dateIime </updated> ?
  <property key="xs:string"> xs:string </property> *
  <state> xs:string </state>
  <command>
       ( <httpMethod name=(“POST”|”GET”|”DELETE”|”PUT”|”PATCH”)> xs:anyURI
</httpMethod>
         <httpHeader name=”xs:string”> xs:string </httpHeader>
         <body> <xs:any>* </body> ) |
       ( <cmdtype name=”xs:string” />
         <cmd> xs:string </cmd> )
  </command>
  <nestedJobTemplate href="xs:anyURI"/> *
  <operation rel="edit" href="xs:anyURI"/> ?
  <operation rel="delete" href="xs:anyURI"/> ?
  <xs:any>*
```

```
</JobTemplate>
```

6.3.2 Update on JobCollection Resource (4.6.3):

The following updates are made against 4.6.3:

6.3.2.1 JobCollection Resource

A `JobCollection` Resource represents the Collection of `Jobs` within a Provider and follows the Collection pattern defined in clause DSP0263 clause **Error! Reference source not found.**. This esource shall be serialized as follows:

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/1/JobCollection",
  "id": string,
  "count": integer,
  "jobs": [
    { "resourceURI": "http://schemas.dmtf.org/cimi/1/Job",
      "id": string,
      ... remaining Job attributes ...
    }, +
  ], ?
 [EXT-ADD:     "operations": [ { "rel": "add", "href": string } ? ] ]
  ...
}
```

XML serialization:

```
<Collection resourceURI="http://schemas.dmtf.org/cimi/1/JobCollection"
    xmlns="http://schemas.dmtf.org/cimi/1">
  <id> xs:anyURI </id>
  <count> xs:integer </count>
  <Job>
    <id> xs:anyURI </id>
    ... remaining Job attributes ...
  </Job> *
  [EXT-ADD:     <operation rel="add" href="xs:anyURI"/> ? ]
  <xs:any>*
</Collection>
```

6.3.3 Addition of a JobTemplate Collection Resource:

The following addition is made:

6.3.3.1 JobTemplateCollection Resource

A `JobTemplateCollection` Resource represents the Collection of `JobTemplates` within a Provider and follows the Collection pattern defined in DSP0263 clause **Error! Reference source not ound.**. This Resource shall be serialized as follows:

JSON serialization:

```
{ "resourceURI": "http://schemas.dmtf.org/cimi/1/JobTemplateCollection",
  "id": string,
  "count": integer,
  "jobTemplates": [
    { "resourceURI": "http://schemas.dmtf.org/cimi/1/JobTemplate",
      "id": string,
      ... remaining JobTemplate attributes ...
    }, +
  ], ?
  "operations": [ { "rel": "add", "href": string } ? ]
  ...
}
```

XML serialization:

```
<Collection resourceURI="http://schemas.dmtf.org/cimi/1/JobTemplateCollection"
    xmlns="http://schemas.dmtf.org/cimi/1">
  <id> xs:anyURI </id>
  <count> xs:integer </count>
  <JobTemplate>
    <id> xs:anyURI </id>
    ... remaining JobTemplate attributes ...
  </JobTemplate> *
  <operation rel="add" href="xs:anyURI"/> ?
  <xs:any>*
</Collection>
```

6.4 Implementation considerations

The registration of a scheduling command uses the JobTemplate CIMI resource as a gateway to the POSIX command line interface. The POSIX commands are executed on the Provider side. There are two implementation options, the details of which are outside the scope of this extension specification:

- The OS used for the extension is under control of the Consumer, and under its responsibility – i.e., a virtual Machine dedicated for this purpose. The VM used for the extension (e.g., running Linux) is subject to the same security procedures as the other resources of a CEP, e.g., access control, isolation from other CEPs.
- The OS used for the extension is under control of the Provider, and under its responsibility. Typically, the Provider would manage it as part of the container for a given CEP. The VM used

for the extension (e.g., running Linux) is not visible or accessible to the Consumer. It may still be specific to each particular CEP for security reasons.

The `JobTemplate` resource allows for defining an external command or a shell script invocation and to persist these on the Provider side. This definition can be reused. Every time a `Job` is created from such a `JobTemplate`, the Consumer is triggering an execution of this command invocation or shell script.

Such a script or command can in turn execute one or more CIMI operations and creates the corresponding jobs as nested jobs of the originator. A wrapper script is defined of name “cimiexec” that takes as first argument a `JobTemplate` name. Invocation of this script creates a `Job` from this `JobTemplate` - acting as a proxy Consumer.

NOTE The equivalent could be a `curl` command; however, the definition of a custom cimiexec script instead gives more flexibility to implementers, (`curl` is not part of the standard POSIX.1-2008 utility set), and is more concise than a full `curl` command (hides unnecessarily exposed URLs such as “jobs” collection JobTemplate URL for a given template name).

ANNEX A
(normative)

CIMI Extension Document Template

ANNEX B
(informative)

Change log

Version	Date	Description
1.0.0	2014-11-14	BLG scrub that adds POSIX-1 2008 compliant scheduling support
1.0.0	2014-12-08	wgv 0.4.0 posted update from BrightLeaf - comments need to be addressed
1.0.0a	2015-02-25	wgv 0.4.1 addressed comments and accepted changes work in progress release candidate

Bibliography

DMTF DSP4014, *DMTF Process for Working Bodies 2.0*, http://www.dmtf.org/sites/default/files/standards/documents/DSP4014_2.0.0.pdf