CAPMC API Documentation

Release 1.0

Cray Inc.

June 26, 2015

CHAPTER

ONE

CAPMC CLI

The capmc utility provides remote monitoring and control capabilities to agents running externally to the Cray System Management Workstation (SMW). The capmc client utility accepts command line arguments from the caller and submits an appropriately formatted request via HTTPS to the monitoring and control service providers running on the SMW. Command results are supplied as JSON-formatted text to standard output. Remote access is restricted by means of public-key authorization established by the site administrator.

In order to use the capmc API calls, also referred to as *applets*, caller must first load the capmc module:

```
$ module load capmc
```

Alternatively, the caller can invoke capmc by specifying the absolute path (determined on installation). The default path is: **/opt/cray/capmc/default/bin/capmc**

1.1 Installation

The capmc utility is packaged in a noarch RPM format. Its only dependency is the Python Standard Library with SSL support enabled, version 2.6 or 2.7. On RPM based systems, the package may be installed with the native package manager.

```
$ rpm -ivh cray-capmc-1.0-1.0000.36550.40.1.noarch.rpm
Preparing...       ########################################## [100%]
   1:cray-capmc    ########################################## [100%]
```

Alternately, capmc may be installed on systems that do not support the RPM format using the rpm2cpio utility. The following example extracts the package contents into the current working directory.

```
$ rpm2cpio cray-capmc-1.0-1.0000.36550.40.1.noarch.rpm | cpio -idmv
./etc/opt/cray
./etc/opt/cray/capmc
./opt/cray
./opt/cray/capmc
./opt/cray/capmc/1.0-1.0000.36550.40.1
./opt/cray/capmc/1.0-1.0000.36550.40.1/bin
./opt/cray/capmc/1.0-1.0000.36550.40.1/bin/capmc
...
```

1.2 Return Values

The capmc utility returns 0 to the shell if the request was successfully submitted to the platform monitoring and control service. Individual command status is indicated in the JSON-encoded output data 'e' member. 'e' is always present in

the response payloads and will be non-zero to indicate a command specific error code if the requested operation failed. Callers of capmc should always evaluate result status returned in the message payload.

1.3 Credentials

Capmc utilizes an X.509 client certificate for authorization. The signed client certificate and private key will be supplied by the system administrator. Credentials may be supplied through environment variables or a configuration file.

1.3.1 Environment Variables

The required environment variables are listed below. Environment variables will override configuration file parameters. Environment variables which must be set to utilize client certificate authorization are as follows:

OS_KEY
: Specify the absolute path in the local filesystem where the X.509 client certificate key is installed. If a pass phrase is set on the key, the user will be prompted when required.

OS_CERT
: Specify the absolute path in the local filesystem where the X.509 client certificate is installed.

OS_CACERT
: Specify the absolute path in the local filesystem where the X.509 Certificate Authority (CA) certificate is located.

OS_SERVICE_URL
: Specify the URL where the application service is listening. This must include the fully qualified domain name (FQDN) of the SMW, protocol, and port number. The default port number is 8443.

1.3.2 Configuration File

When capmc is utilized for autonomous machine to machine communication, it is advisable to define the certificate paths and service url in a configuration file instead of the callers environment. Capmc reads its configuration file from the following location: **/etc/opt/cray/capmc/capmc.json**

X.509 client certificate configuration syntax:

```
{
   "os_key":         "/etc/opt/cray/capmc/capmc-client.key",
   "os_cert":        "/etc/opt/cray/capmc/capmc-client.pem",
   "os_cacert":      "/etc/opt/cray/capmc/capmc-cacert.pem",
   "os_service_url": "https://smw.example.com:8443"
}
```

Note: Capmc validates all X.509 certificates. It is assumed that all host certificates used within HTTPS API servers such as Xtremoted, or X.509 client authorization are signed by the same certificate authority.

Warning: The configuration and X.509 certificate files must have appropriate filesystem permissions. Users with read access to the global configuration file and the client certificate will be granted permission to utilize all capmc functionality.

CHAPTER

TWO

NODE STATUS & ON OFF CONTROL

CAPMC node power controls implement a simple interface for powering on or off compute nodes, querying node state information, and querying site-specific service usage rules. These controls enable external software to more intelligently manage system wide power consumption or configuration parameters. The simplest power management strategy may be to simply turn off compute nodes which may be idle for a significant time interval and turn them back on when demand increases. The following API calls are provided as a means for third party software to implement power management strategies as simple or complex as the site-level requirements demand.

2.1 node_rules

node_rules informs third party software about hardware (and perhaps site-specific) rules or timing constraints that allow for efficient and effective management of idle node resources. The data returned informs the caller of how long **node_on** and **node_off** operations are expected to take, the minimum amount of time nodes should be left off to save energy, and limits on the number of nodes that should be turned on or off at once. Default rules are supplied where appropriate.

Other values such as the maximum node counts for **node_on** or **node_off** and the maximum amount of time a node should remain off after a power down are left unset. The values are not strictly enforced by Cray system management software. They are meant to provide guidelines for authorized callers in their use of the CAPMC service.

2.1.1 CLI Interface

capmc node_rules

The command line interface has no arguments. Result text is returned via standard out.

```
$ capmc node_rules
{
  "e": 0,
  "err_msg": "",
  "latency_node_off": 60,
  ...
```

2.1.2 HTTP Interface

POST /capmc/get_node_rules

Example request:

```
POST /capmc/get_node_rules HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 2
Content-Type: application/json

{}
```

The request must **POST** an empty JSON object to the API server. This command takes no arguments.

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "",
  "latency_node_off": 60,
  "latency_node_on": 600,
  "max_off_req_count": -1,
  "max_off_time": -1,
  "max_on_req_count": -1,
  "min_off_time": 900
}
```

JSON Parameters

- **e** (*int*) – Request status code, zero on success
- **err_msg** (*string*) – Human readable error message
- **latency_node_off** (*int*) – Approximate time, in seconds, in which a node is expected to perform a clean shutdown and power off
- **latency_node_on** (*int*) – Approximate time, in seconds, in which a node is expected to perform a warm bounce and boot into a ready state
- **max_off_req_count** (*int*) – Maximum number of nodes which may be powered off at once. -1 indicates no limit
- **max_off_time** (*int*) – Maximum time, in seconds, in which a node may be in the powered off state. -1 indicates no limit
- **max_on_req_count** (*int*) – Maximum number of nodes which may be powered on and booted at once. -1 indicates no limit
- **min_off_time** (*int*) – Minimum time, in seconds, in which a node must remain in the powered off state after a shutdown and power off operation. -1 indicates no limit

Status Codes

- **200** – Network API call success

Note: Default rules are established automatically at system installation time. The administrator may choose to customize the rule set. Customization is performed by editing the respective parameter in a configuration file (**/opt/cray/hss/default/etc/xtremoted/rules.ini**) stored on the System Management Workstation.

2.2 node_status

A node's component state may be returned via the **node_status** function for the full system or a subset as specified by a nid range list or component filter. The status API call is intended, but not limited, to be used in conjunction with asynchronous operations which may modify node component state, such as **node_on** or **node_off**.

Third party utilities would issue an asynchronous operation, such as **node_on**, and if the operation was successfully enqueued poll for changes in component state after waiting for the expected boot time latency. If the targeted component state has switched from "off" to "ready" then the caller knows the operation was successful.

States reported through this API call mirror those defined in Cray HSS.

Node States:

- **disabled** - Component is physically installed, but ignored by Cray system management software
- **halt** - Operating system has shut down, hardware has not yet powered off
- **on** - Power is on and BIOS has initialized all hardware, node is waiting to be booted
- **off** - Power is off
- **ready** - Operating system is fully booted
- **standby** - Operating system is in process of booting

Note: The **node_status** API call does not report **empty** components.

2.2.1 CLI Interface

capmc node_status

-n, --nids <nid range list>
: Specify a rangelist of NIDs to query status. If omitted, the default is all NIDs.

-f, --filter <opts>
: Specify filters for status query. Filters may be pipe-separated (|), e.g. "optloptlopt". Valid filters are: **show_all**, **show_off**, **show_on**, **show_halt**, **show_standby**, **show_ready**, **show_diag**, and **show_disabled**. If omitted, the default is **show_all**.

```
$ capmc node_status --nids=1,40-43 --filter=show_on|show_ready
{
    "e": 0,
    "err_msg": "",
    "on": [
        40,
        42,
        43,
        41
    ],
    "ready": [
        1
    ]
...
```

2.2.2 HTTP Interface

POST /capmc/get_node_status

Example Request:

```
POST /capmc/get_node_status HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 96
Content-Type: application/json

{
  "filter": "show_on|show_ready",
  "nids": [
    1,
    40,
    41,
    42,
    43
  ]
}
```

The request must **POST** a properly formatted JSON object to the API server. The command takes two optional arguments which identify a NID list and component status filter.

JSON Parameters

- **filter** (*string*) – Pipe concatenated (|) list of filter strings
- **nids** (*int[]*) – User specified list, or empty array for all NIDs. This list must not contain invalid or duplicate NID numbers. If invalid NID numbers are specified then an error will be returned.

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "",
  "on": [
    40,
    42,
    43,
    41
  ],
  "ready": [
    1
  ]
}
```

JSON Parameters

- **e** (*int*) – Request status code, zero on success, nonzero on invalid input.
- **err_msg** (*string*) – Human readable error message
- **disabled** (*int[]*) – Optional member, list of disabled NIDs
- **halt** (*int[]*) – Optional, list of halted NIDs

- **on** (*int[]*) – Optional, list of powered on NIDs
- **off** (*int[]*) – Optional, list of powered off NIDs
- **ready** (*int[]*) – Optional, list of booted NIDs
- **standby** (*int[]*) – Optional, list of booting NIDs

Status Codes

- **200** – Network API call success
- **500** – Internal command failure

2.3 node_on

Power on and warm boot a selected list of compute node NIDs using the default boot image. This has no effect on the status of the high speed network (HSN). However, this command requires that the HSN ASIC attached to each node in the target list has previously been powered on and routed.

The **node_on** API call is **asynchronous**. It returns immediately containing a status result indicating an error with invalid input parameters, or success indicating the operation has been enqueued into an asynchronous command processing queue. The caller must determine overall command status by polling for **node_status** after the expected power on and warm boot period has elapsed.

2.3.1 CLI Interface

capmc node_on

-n, --nids <nid range list>
: Specify a list of NIDs to power on and boot. Only compute node NIDs may be specified. This is a required option.

-r, --reason <log message>
: Specify an arbitrary text message which is given as the reason for performing the node_on operation. This argument is optional and used for informational purposes when reviewing system log messages. The value is not interpreted in any way by the Cray system management software.

```
$ capmc node_on --nids=40-43 --reason="need more capacity"
{
    "e": 0,
    "err_msg": ""
}
```

2.3.2 HTTP Interface

POST /capmc/node_on
: **Example Request**:

```
POST /capmc/node_on HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 85
Content-Type: application/json

{
```

```
  "reason": "need more capacity",
  "nids": [
    40,
    41,
    42,
    43
  ]
}
```

The request must **POST** an array of selected NIDs along with an optional human readable reason string.

JSON Parameters

- **reason** (*string*) – Arbitrary, free-form text
- **nids** (*int[]*) – User specified list of compute node NIDs to warm bounce and boot. An empty array is invalid. If invalid NID numbers are specified then an error will be returned.

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": ""
}
```

JSON Parameters

- **e** (*int*) – Request status code, zero on success, nonzero on invalid input.
- **err_msg** (*string*) – Human readable error message

Status Codes

- **200** – Network API call success
- **500** – Internal command failure

2.4 node_off

Shutdown and power off a selected list of compute node NIDs. This has no effect on the status of the high speed network (HSN). The HSN ASIC attached to each node will remain powered on and routed. After the **node_off** operation has completed, the selected nodes will be in a state suitable for warm booting back into the system at a later date.

The **node_off** API call is **asynchronous**. It returns immediately containing a status result indicating an error with invalid input parameters, or success indicating the operation has been enqueued into an asynchronous command processing queue. The caller must determine overall command status by polling for **node_status** after the expected shutdown and power off period has elapsed.

2.4.1 CLI Interface

capmc node_off

-n, --nids <nid range list>
Specify a list of NIDs to shut down and power off. Only compute node NIDs may be specified. This is a required option.

-r, --reason <log message>
Specify an arbitrary text message which is given as the reason for performing the node_on operation. This argument is optional and is used in the same manner as with the **node_on** command.

```
$ capmc node_off --nids=40-43 --reason="powersave, need less capacity"
{
    "e": 0,
    "err_msg": ""
}
```

2.4.2 HTTP Interface

POST /capmc/node_off

Example Request:

```
POST /capmc/node_on HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 96
Content-Type: application/json

{
  "reason": "powersave, need less capacity",
  "nids": [
    40,
    41,
    42,
    43
  ]
}
```

The request must **POST** an array of selected NIDs along with an optional human readable reason string.

JSON Parameters

- **reason** (*string*) – Arbitrary, free-form text
- **nids** (*int[]*) – User specified list of compute node NIDs to shutdown and power off. An empty array is invalid. If invalid NID numbers are specified then an error will be returned.

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": ""
}
```

JSON Parameters

- **e** (*int*) – Request status code, zero on success, nonzero on invalid input.
- **err_msg** (*string*) – Human readable error message

Status Codes

- **200** – Network API call success
- **500** – Internal command failure

CHAPTER

THREE

NODE CAPABILITIES & POWER CONTROL

CAPMC power capping controls implement a simple interface for querying component capabilities and manipulation of node or sub-node (accelerator) power constraints. This functionality enables external software to establish an upper bound, or estimate a minimum bound, on the amount of power a system or a select subset of the system may consume. The following API calls are provided as a means for third party software to implement advanced power management strategies.

3.1 get_power_cap_capabilities

The **get_power_cap_capabilities** call informs third-party software about installed hardware and its associated properties. Information returned includes the specific hardware types, NID membership, and power capping controls along with their allowable ranges. Information may be returned for a targeted set of NIDs or the system as a whole.

3.1.1 CLI Interface

capmc get_power_cap_capabilities

-n, --nids <nid range list>
: Specify a rangelist of NIDs to query status. If omitted, the default is all NIDs. The specified NIDs may be in any state.

```
$ capmc get_power_cap_capabilities --nids=40-43,48-49
{
  "e": 0,
  "err_msg": "",
  "groups": [
    {
      "name": "01:000d:206d:0073:0008:0020:3a34:8100",
      "desc": "ComputeANC_SNB_115W_8c_32GB_14900_KeplerK20XAccel",
      "host_limit_max": 185,
 ...
```

3.1.2 HTTP Interface

POST /capmc/get_power_cap_capabilities
: **Example Request**:

```
POST /capmc/get_power_cap_capabilities HTTP/1.1
Host: smw.example.com
Accept: application/json
```

```
Content-Length: 71
Content-Type: application/json

{
  "nids": [
    40,
    41,
    42,
    43,
    48,
    49
  ]
}
```

The request must **POST** a properly formatted JSON object to the API server. The command takes a single argument which identifies a NID list.

JSON Parameters

- **nids** (*int[]*) – User specified list, or empty array for all NIDs. This list must not contain invalid or duplicate NID numbers. If invalid NID numbers are specified then an error will be returned.

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "",
  "groups": [
    {
      "name": "01:000d:206d:0073:0008:0020:3a34:8100",
      "desc": "ComputeANC_SNB_115W_8c_32GB_14900_KeplerK20XAccel",
      "host_limit_max": 185,
      "host_limit_min": 95,
      "static": 0,
      "supply": 425,
      "powerup": 120,
      "nids": [
        48,
        49
      ],
      "controls": [
        {
          "name": "accel",
          "desc": "Accelerator control",
          "max": 225,
          "min": 180
        },
        {
          "name": "node",
          "desc": "Node manager control",
          "max": 410,
          "min": 275
        }
      ]
    },
```

```
    {
      "name": "01:000d:206d:0104:0010:0040:3200:0000",
      "desc": "ComputeANC_SNB_260W_16c_64GB_12800_NoAccel",
      "host_limit_max": 350,
      "host_limit_min": 200,
      "static": 0,
      "supply": 425,
      "powerup": 150,
      "nids": [
        40,
        41,
        42,
        43
      ],
      "controls": [
        {
          "name": "node",
          "desc": "Node manager control",
          "max": 350,
          "min": 200
        }
      ]
    }
  ]
}
```


JSON Parameters

- **e** (*int*) – Request status code, zero on success
- **err_msg** (*string*) – Human readable error message
- **groups** (*object[]*) – Object array containing hardware specific information and NID membership, each element represents a unique hardware type
- **groups[].name** (*string*) – Opaque identifier which Cray system management software uses to uniquely identify a node type
- **groups[].desc** (*string*) – Text description of the opaque node type identifier
- **groups[].host_limit_max** (*int*) – Estimated maximum power, specified in watts, which host CPU(s) and memory may consume
- **groups[].host_limit_min** (*int*) – Estimated minimum power, specified in watts, which host CPU(s) and memory require to operate
- **groups[].static** (*int*) – Static per node power overhead, specified in watts, which is unreported
- **groups[].supply** (*int*) – Maximum capacity of each node level power supply for the given hardware type, specified in watts
- **groups[].powerup** (*int*) – Typical power consumption of each node during hardware initialization, specified in watts
- **groups[].nids** (*int[]*) – NID members belonging to the given hardware type
- **groups[].controls** (*object[]*) – Array of node level control objects which may be assigned or queried, one element per control
- **groups[].controls[].name** (*string*) – Unique control object identifier

- **groups[].controls[].desc** (*string*) – Human readable description of the control object
- **groups[].controls[].min** (*int*) – Minimum value which may be assigned to the control object, units are dependent upon control type
- **groups[].controls[].max** (*int*) – Maximum value which may be assigned to the control object, units are dependent upon control type

Status Codes

- **200** – Network API call success
- **500** – Internal command failure

3.2 get_power_cap

The **get_power_cap** call returns the power capping control(s) and currently applied settings for the requested rangelist of NIDs. Control values which are returned as zero have special meaning. In such case, a zero value indicates the respective control is unconstrained.

3.2.1 CLI Interface

capmc get_power_cap

-n, --nids <nid range list>
Specify a rangelist of NIDs to query node level, and if applicable, accelerator level power caps. If omitted, the default is all NIDs. The specified NIDs must be in the **ready** state per the **node_status** command.

```
$ capmc get_power_cap --nids=40,48
{
  "e": 0,
  "err_msg": "",
  "nids": [
    {
      "nid": 40,
      "controls": [
...
```

3.2.2 HTTP Interface

POST /capmc/get_power_cap
Example Request:

```
POST /capmc/get_power_cap HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 35
Content-Type: application/json

{
  "nids": [
    40,
    48,
  ]
}
```

The request must **POST** a properly formatted JSON object to the API server. The command takes a single argument which identifies a NID list.

JSON Parameters

- **nids** (*int[]*) – User specified list, or empty array for all NIDs. This list must not contain invalid or duplicate NID numbers. If invalid NID numbers are specified then an error will be returned.

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "",
  "nids": [
    {
      "nid": 40,
      "controls": [
        {
          "name": "node",
          "val": 0
        }
      ]
    },
    {
      "nid": 48,
      "controls": [
        {
          "name": "node",
          "val": 0
        },
        {
          "name": "accel",
          "val": 0
        }
      ]
    }
  ]
}
```

Example Response: (Partial Success or Failure)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 52,
  "err_msg": "Invalid exchange",
  "nids": [
    {
      "nid": 40,
      "e": 52,
      "err_msg": "Invalid exchange"
    },
    {
      "nid": 48,
      "controls": [
```

```
            {
              "name": "node",
              "val": 0
            },
            {
              "name": "accel",
              "val": 0
            }
          ]
        }
      ]
    }
```


JSON Parameters

- **e** (*int*) – Overall request status code, zero on total success, non-zero if one or more node specific operations fail
- **err_msg** (*string*) – Human readable error message
- **nids** (*object[]*) – Object array containing NID specific result data, each element represents a single NID
- **nids[].nid** (*int*) – NID number owning the returned control objects
- **nids[].e** (*int*) – Optional, error status, non-zero indicates operation failed on this node
- **nids[].err_msg** (*string*) – Optional, human readable error message applicable to this node
- **nids[].controls** (*object[]*) – Optional, array of node level control objects which have been queried, one element per control
- **nids[].controls[].name** (*string*) – Unique control object identifier
- **nids[].controls[].val** (*int*) – Control object setting, or zero to indicate control is unconstrained, units are dependent upon control type

Status Codes

- **200** – Network API call success
- **500** – Internal command failure

3.3 set_power_cap

The **set_power_cap** call is used to establish an upper bound with respect to power consumption on a per-node, and if applicable, a sub-node basis. Established power cap parameters will revert to the default configuration on the next system boot.

If setting multiple different power caps is desired, then it is recommended that those be set programmatically via the **json** command with an input data structure conforming to the HTTP Interface request format for this command. The **json** command allows third party software to pass its own JSON formatted requests in a single transaction to the HTTP API service.

Note: Service nodes may not be power capped. If service node NIDs are specified then the request will fail with an invalid parameters error. When applying a power cap, unspecified controls are reset to their default value.

3.3.1 CLI Interface

capmc set_power_cap

-n, --nids <nid range list>
Specify a rangelist of NIDs to apply the specified power caps. This option may not be omitted. The specified NIDs must be in the **ready** state per the **node_status** command.

-N, --node <watts>
Specify the desired node level power cap. The value given must be within the range returned in the capabilities output. A value of zero may be supplied to explicitly clear an existing node level power cap.

Nodes with high powered accelerators and high TDP processors will be automatically power capped at the "supply" limit returned per the **get_power_cap_capabilities** command. If a node level power cap is specified that is within the node control range but exceeds the supply limit, the actual power cap assigned will be clamped at the supply limit.

-A, --accel <watts>
Specify the desired accelerator component power cap. The value given must be within the range returned in the capabilities output. A value of zero may be supplied to to explicitly clear an accelerator power cap.

The accelerator power cap value represents a subset of the total node level power cap. If a node level power cap of 400 watts is applied and an accelerator power cap of 180 watts is applied, then the total node power consumption is limited to 400 watts. If the accelerator is actively consuming its entire 180 watt power allocation, then the host processor, memory subsystem, and support logic for that node may consume a maximum of 220 watts.

3.3.2 HTTP Interface

POST /capmc/set_power_cap
Example Request:

```
POST /capmc/set_power_cap HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 472
Content-Type: application/json

{
  "nids": [
    {
      "nid": 20,
      "controls": [
        {
          "name": "node",
          "val": 400
        }
      ]
    },
    {
      "nid": 21,
      "controls": [
        {
          "name": "node",
          "val": 400
        }
      ]
    },
```

```
    {
      "nid": 60,
      "controls": [
        {
          "name": "node",
          "val": 410
        },
        {
          "name": "accel",
          "val": 220
        }
      ]
    }
  ]
}
```

The request must **POST** a properly formatted JSON object to the API server. The command takes an array of objects which identify target component NIDs, control names, and their associated set point values.

JSON Parameters

- **nids** (*object[]*) – Object array containing NID specific input data, each element represents a single NID
- **nids[].nid** (*int*) – NID number owning the specified input control objects
- **nids[].controls** (*object[]*) – Array of node level control objects to be adjusted, one element per control
- **nids[].controls[].name** (*string*) – Unique control object identifier
- **nids[].controls[].val** (*int*) – Control object setting, or zero to indicate control is unconstrained, units are dependent upon control type

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "e":0,
    "err_msg":""
}
```

Example Response: (Partial Success or Failure)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 52,
  "err_msg": "Invalid exchange",
  "nids": [
    {
      "nid": 60,
      "e": 52,
      "err_msg": "Invalid exchange"
    }
  ]
}
```

In the common case, the response payload is short and consists only of an integer status code and an optional message. However there may be instances, likely due to hardware errors, where a small number of nodes encounter a problem and are unable to comply with the command. If an error does occur, extra information pertaining to the specific component where the error occurred is included in the response payload.

JSON Parameters

- **e** (*int*) – Request status code, zero on success.
- **err_msg** (*string*) – Human readable error message
- **nids** (*object[]*) – Object array containing NID specific error data, NIDs which experienced success are omitted
- **nids[].nid** (*int*) – NID number owning the returned error data
- **nids[].e** (*int*) – Error status, non-zero indicates operation failed on this node
- **nids[].err_msg** (*string*) – Human readable error string applicable to this node

Status Codes

- **200** – Network API call success
- **500** – Internal command failure

CHAPTER

FOUR

NODE FREQUENCY & SLEEP STATE CONTROL

CAPMC node frequency and sleep state controls implement an interface such that external agents may modify, on the fly, CPU operating frequencies and sleep state limits. For example, an external agent may reallocate power amongst nodes by adjusting operating frequencies. A system administrator may disable package sleep states on idle nodes in an effort to keep system power consumption inside the power band negotiated with the utility provider. Or, batch schedulers may optimize power efficiency by running phases of an application at differing frequencies. The controls are provided as a means for third-party integrators to implement advanced power management policies.

Note: Runtime p-state & c-state controls are under development. The features described in this section are currently unreleased.

4.1 get_freq_capabilities

The **get_freq_capabilities** call informs third-party software about supported processor operating frequencies. Valid operating frequencies, specified in Hz, are returned in list form.

4.1.1 CLI Interface

capmc get_freq_capabilities

-n, --nids <nid range list>
Specify a rangelist of NIDs for which to query allowable frequency limits. This is a required option.

```
$ capmc get_freq_capabilities --nids=50
{
    "e": 0,
    "err_msg": "",
    "nids": [
      {
        "data": {
          "PWR_Attrs": [
            {
              "PWR_AttrName": "PWR_ATTR_FREQ",
...
```

4.1.2 HTTP Interface

POST /capmc/cnctl
Example request:

```
POST /capmc/cnctl HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 234
Content-Type: application/json

{
  "nids": [
    50
  ],
  "data": {
    "PWR_Function": "PWR_ObjAttrValueCapabilities",
    "PWR_Attrs": [
      {
        "PWR_AttrName": "PWR_ATTR_FREQ"
      }
    ],
    "PWR_MajorVersion": 0,
    "PWR_MinorVersion": 1
  }
}
```

The request must **POST** a JSON object to the API server containing a remote API call function name, a single element list containing the appropriate attribute name, and the protocol version.

JSON Parameters

- **nids** (*int[]*) – User specified NID list, must **not** be empty
- **data** (*object*) – Remote API call input object
- **data.PWR_Function** (*string*) – Remote API call function name, the caller must specify "PWR_ObjAttrValueCapabilities"
- **data.PWR_Attrs** (*object[]*) – Array of attribute name value pairs
- **data.PWR_Attrs[].PWR_AttrName** (*string*) – Attribute name, the caller must specify "PWR_ATTR_FREQ"
- **data.PWR_MajorVersion** (*int*) – Remote API call major version number, the caller must specify "0"
- **data.PWR_MinorVersion** (*int*) – Remote API call minor version number, the caller must specify "1"

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "Success",
  "nids": [
    {
      "nid": 50,
      "data": {
        "PWR_Attrs": [
          {
            "PWR_AttrName": "PWR_ATTR_FREQ",
            "PWR_AttrValueCapabilities": [
```

```
                2601000,
                2600000,
                2500000,
                2400000,
                2300000,
                2200000,
                2100000,
                2000000,
                1900000,
                1800000,
                1700000,
                1600000,
                1500000,
                1400000,
                1300000,
                1200000
              ],
              "PWR_ReturnCode": 0
            }
          ],
          "PWR_ReturnCode": 0,
          "PWR_ErrorMessages": null,
          "PWR_Messages": null,
          "PWR_MajorVersion": 0,
          "PWR_MinorVersion": 1
        }
      }
    ]
}
```


JSON Parameters

- **e** (*int*) – Overall request status code, non-zero on partial success or failure
- **err_msg** (*string*) – Human readable error message
- **nids** (*object[]*) – Object array containing NID specific result data, each element represents a single NID
- **nids[].nid** (*int*) – NID number owning the returned attribute objects
- **nids[].data** (*object*) – Remote API call result object
- **nids[].data.PWR_Attrs** (*object[]*) – Array of attribute name value pairs
- **nids[].data.PWR_Attrs[].PWR_AttrName** (*string*) – Attribute name copied from original request
- **nids[].data.PWR_Attrs[].PWR_AttrValueCapabilities** (*int[]*) – List of supported processor operating frequencies, in Hz
- **nids[].data.PWR_Attrs[].PWR_ReturnCode** (*int*) – Attribute probe specific result code, non-zero on failure
- **nids[].data.PWR_ReturnCode** (*int*) – Per-node attribute probe result code, non-zero on failure
- **nids[].data.PWR_ErrorMessages** (*string*) – Per-node attribute error message, or null
- **nids[].data.PWR_Messages** (*string*) – Per-node attribute info message, or null
- **nids[].data.PWR_MajorVersion** (*int*) – Remote API call major version number

- **nids[].data.PWR_MinorVersion** (*int*) – Remote API call minor version number

Status Codes

- **200** – Network API call success
- **400** – Bad Request
- **500** – Internal command failure
- **504** – Gateway Timeout

4.2 get_freq_limits

The **get_freq_limits** call returns the minimum and maximum allowable operating frequencies on a per-node basis. The processor frequency operating window may be constrained from defaults using the **set_freq_limits** call.

4.2.1 CLI Interface

capmc get_freq_limits

-n, --nids <nid range list>
Specify a rangelist of NIDs for which to query currently active frequency limits. This is a required option.

```
$ capmc get_freq_limits --nids=50
{
    "e": 0,
    "err_msg": "",
...
```

4.2.2 HTTP Interface

POST /capmc/cnctl
Example request:

```
POST /capmc/cnctl HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 234
Content-Type: application/json

{
  "nids": [
    50
  ],
  "data": {
    "PWR_Function": "PWR_ObjAttrGetValues",
    "PWR_Attrs": [
      {
        "PWR_AttrName": "PWR_ATTR_FREQ"
      },
      {
        "PWR_AttrName": "PWR_ATTR_FREQ_LIMIT_MIN"
      },
      {
        "PWR_AttrName": "PWR_ATTR_FREQ_LIMIT_MAX"
```

```
        }
      ],
      "PWR_MajorVersion": 0,
      "PWR_MinorVersion": 1
    }
}
```

The request must **POST** a JSON object to the API server containing a remote API call function name, a three element list containing the appropriate attribute names, and the protocol version.

JSON Parameters

- **nids** (*int[]*) – User specified NID list, must **not** be empty
- **data** (*object*) – Remote API call input object
- **data.PWR_Function** (*string*) – Remote API call function name, the caller must specify "PWR_ObjAttrGetValues"
- **data.PWR_Attrs** (*object[]*) – Array of attribute name value pairs
- **data.PWR_Attrs[].PWR_AttrName** (*string*) – Attribute name, the caller must specify array elements for attributes named "PWR_ATTR_FREQ", "PWR_ATTR_FREQ_LIMIT_MIN", and "PWR_ATTR_FREQ_LIMIT_MAX"
- **data.PWR_MajorVersion** (*int*) – Remote API call major version number, the caller must specify "0"
- **data.PWR_MinorVersion** (*int*) – Remote API call minor version number, the caller must specify "1"

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "Success",
  "nids": [
    {
      "nid": 50,
      "data": {
        "PWR_Attrs": [
          {
            "PWR_AttrName": "PWR_ATTR_FREQ",
            "PWR_AttrValue": 2601000,
            "PWR_ReturnCode": 0,
            "PWR_TimeNanoseconds": 80864412,
            "PWR_TimeSeconds": 1433536488
          },
          {
            "PWR_AttrName": "PWR_ATTR_FREQ_LIMIT_MAX",
            "PWR_AttrValue": 2601000,
            "PWR_ReturnCode": 0,
            "PWR_TimeNanoseconds": 80876654,
            "PWR_TimeSeconds": 1433536488
          },
          {
            "PWR_AttrName": "PWR_ATTR_FREQ_LIMIT_MIN",
            "PWR_AttrValue": 1200000,
```

```
            "PWR_ReturnCode": 0,
            "PWR_TimeNanoseconds": 80883584,
            "PWR_TimeSeconds": 1433536488
          }
        ],
        "PWR_ErrorMessages": null,
        "PWR_MajorVersion": 0,
        "PWR_Messages": null,
        "PWR_MinorVersion": 1,
        "PWR_ReturnCode": 0
      }
    }
  ]
}
```


JSON Parameters

- **e** (*int*) – Overall request status code, non-zero on partial success or failure
- **err_msg** (*string*) – Human readable error message
- **nids** (*object[]*) – Object array containing NID specific result data, each element represents a single NID
- **nids[].nid** (*int*) – NID number owning the returned attribute objects
- **nids[].data** (*object*) – Remote API call result object
- **nids[].data.PWR_Attrs** (*object[]*) – Array of attribute name value pairs
- **nids[].data.PWR_Attrs[].PWR_AttrName** (*string*) – Attribute name copied from original request
- **nids[].data.PWR_Attrs[].PWR_AttrValue** (*int[]*) – Returned attribute value, in Hz
- **nids[].data.PWR_Attrs[].PWR_ReturnCode** (*int*) – Attribute probe specific result code, non-zero on failure
- **nids[].data.PWR_Attrs[].PWR_TimeSeconds** (*int*) – Elapsed seconds since the epoch, in UTC
- **nids[].data.PWR_Attrs[].PWR_TimeNanoseconds** (*int*) – Nanosecond timestamp component
- **nids[].data.PWR_ReturnCode** (*int*) – Per-node attribute probe result code, non-zero on failure
- **nids[].data.PWR_ErrorMessages** (*string*) – Per-node attribute error message, or null
- **nids[].data.PWR_Messages** (*string*) – Per-node attribute info message, or null
- **nids[].data.PWR_MajorVersion** (*int*) – Remote API call major version number
- **nids[].data.PWR_MinorVersion** (*int*) – Remote API call minor version number

Status Codes

- **200** – Network API call success
- **400** – Bad Request
- **500** – Internal command failure
- **504** – Gateway Timeout

4.3 set_freq_limits

The **set_freq_limits** call allows third-party software to constrain the range of frequencies a node's processor(s) may operate at. Valid values for minimum and maximum frequency are returned via the **get_freq_capabilities** call.

4.3.1 CLI Interface

capmc set_freq_limits

-n, --nids <nid range list>
Specify a rangelist of NIDs for which to set frequency limits. This is a required option.

-m, --min <min frequency>
Specify the minimum operating frequency. This is a required option.

-M, --max <min frequency>
Specify the maximum operating frequency. This is a required option.

```
$ capmc set_freq_limits --nids=50 --min=1400000 --max=2500000
{
    "e": 0,
    "err_msg": "",
...
```

4.3.2 HTTP Interface

POST /capmc/cnctl
Example request:

```
POST /capmc/cnctl HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 378
Content-Type: application/json

{
  "nids": [
    50
  ],
  "data": {
    "PWR_Function": "PWR_ObjAttrSetValues",
    "PWR_Attrs": [
      {
        "PWR_AttrName": "PWR_ATTR_FREQ_LIMIT_MIN",
        "PWR_AttrValue": "1400000"
      },
      {
        "PWR_AttrName": "PWR_ATTR_FREQ_LIMIT_MAX",
        "PWR_AttrValue": "2500000"
      }
    ],
    "PWR_MajorVersion": 0,
    "PWR_MinorVersion": 1
  }
}
```

The request must **POST** a JSON object to the API server containing a remote API call function name, a two element list containing the appropriate attribute names, requested frequency limit values, and the protocol version.

JSON Parameters

- **nids** (*int[]*) – User specified NID list, must **not** be empty
- **data** (*object*) – Remote API call input object
- **data.PWR_Function** (*string*) – Remote API call function name, the caller must specify "PWR_ObjAttrValueCapabilities"
- **data.PWR_Attrs** (*object[]*) – Array of attribute name value pairs
- **data.PWR_Attrs[].PWR_AttrName** (*string*) – Attribute name, the caller must specify array elements for attributes named "PWR_ATTR_FREQ_LIMIT_MIN" and "PWR_ATTR_FREQ_LIMIT_MAX"
- **data.PWR_Attrs[].PWR_AttrValue** (*string*) – Attribute value
- **data.PWR_MajorVersion** (*int*) – Remote API call major version number, the caller must specify "0"
- **data.PWR_MinorVersion** (*int*) – Remote API call minor version number, the caller must specify "1"

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "Success",
  "nids": [
    {
      "nid": 50,
      "data": {
        "PWR_Attrs": [
          {
            "PWR_AttrName": "PWR_ATTR_FREQ_LIMIT_MAX",
            "PWR_ReturnCode": 0
          },
          {
            "PWR_AttrName": "PWR_ATTR_FREQ_LIMIT_MIN",
            "PWR_ReturnCode": 0
          }
        ],
        "PWR_ErrorMessages": null,
        "PWR_MajorVersion": 0,
        "PWR_Messages": null,
        "PWR_MinorVersion": 1,
        "PWR_ReturnCode": 0
      }
    }
  ]
}
```

JSON Parameters

- **e** (*int*) – Overall request status code, non-zero on partial success or failure

- **err_msg** (*string*) – Human readable error message
- **nids** (*object[]*) – Object array containing NID specific result data, each element represents a single NID
- **nids[].nid** (*int*) – NID number owning the returned attribute objects
- **nids[].data** (*object*) – Remote API call result object
- **nids[].data.PWR_Attrs** (*object[]*) – Array of attribute name value pairs
- **nids[].data.PWR_Attrs[].PWR_AttrName** (*string*) – Attribute name copied from original request
- **nids[].data.PWR_Attrs[].PWR_ReturnCode** (*int*) – Attribute set specific result code, non-zero on failure
- **nids[].data.PWR_ReturnCode** (*int*) – Per-node attribute set result code, non-zero on failure
- **nids[].data.PWR_ErrorMessages** (*string*) – Per-node attribute error message, or null
- **nids[].data.PWR_Messages** (*string*) – Per-node attribute info message, or null
- **nids[].data.PWR_MajorVersion** (*int*) – Remote API call major version number
- **nids[].data.PWR_MinorVersion** (*int*) – Remote API call minor version number

Status Codes

- **200** – Network API call success
- **400** – Bad Request
- **500** – Internal command failure
- **504** – Gateway Timeout

4.4 get_sleep_state_limit_capabilities

The **get_sleep_state_limit_capabilities** call informs third-party software about supported processor sleep states. Valid sleep states are returned in list form. Higher sleep state numbers correspond to deeper sleep states.

4.4.1 CLI Interface

capmc get_sleep_state_limit_capabilities

-n, --nids <nid range list>
Specify a rangelist of NIDs for which to query allowable sleep state limits. This is a required option.

```
$ capmc get_sleep_state_limit_capabilities --nids=50
{
    "e": 0,
    "err_msg": "",
...
```

4.4.2 HTTP Interface

POST /capmc/cnctl

Example request:

```
POST /capmc/cnctl HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 242
Content-Type: application/json

{
  "nids": [
    50
  ],
  "data": {
    "PWR_Function": "PWR_ObjAttrValueCapabilities",
    "PWR_Attrs": [
      {
        "PWR_AttrName": "PWR_ATTR_CSTATE_LIMIT"
      }
    ],
    "PWR_MajorVersion": 0,
    "PWR_MinorVersion": 1
  }
}
```

The request must **POST** a JSON object to the API server containing a remote API call function name, a single element list containing the appropriate attribute name, and the protocol version.

JSON Parameters

- **nids** (*int[]*) – User specified NID list, must **not** be empty
- **data** (*object*) – Remote API call input object
- **data.PWR_Function** (*string*) – Remote API call function name, the caller must specify "PWR_ObjAttrValueCapabilities"
- **data.PWR_Attrs** (*object[]*) – Array of attribute name value pairs
- **data.PWR_Attrs[].PWR_AttrName** (*string*) – Attribute name, the caller must specify "PWR_ATTR_CSTATE_LIMIT"
- **data.PWR_MajorVersion** (*int*) – Remote API call major version number, the caller must specify "0"
- **data.PWR_MinorVersion** (*int*) – Remote API call minor version number, the caller must specify "1"

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "Success",
  "nids": [
    {
      "nid": 50,
      "data": {
```

```
            "PWR_Attrs": [
              {
                "PWR_AttrName": "PWR_ATTR_CSTATE_LIMIT",
                "PWR_AttrValueCapabilities": [
                  0,
                  1,
                  2,
                  3,
                  4,
                  5
                ],
                "PWR_ReturnCode": 0
              }
            ],
            "PWR_ErrorMessages": null,
            "PWR_MajorVersion": 0,
            "PWR_Messages": null,
            "PWR_MinorVersion": 1,
            "PWR_ReturnCode": 0
          }
        }
      ]
    }
```


JSON Parameters

- **e** (*int*) – Overall request status code, non-zero on partial success or failure
- **err_msg** (*string*) – Human readable error message
- **nids** (*object[]*) – Object array containing NID specific result data, each element represents a single NID
- **nids[].nid** (*int*) – NID number owning the returned attribute objects
- **nids[].data** (*object*) – Remote API call result object
- **nids[].data.PWR_Attrs** (*object[]*) – Array of attribute name value pairs
- **nids[].data.PWR_Attrs[].PWR_AttrName** (*string*) – Attribute name copied from original request
- **nids[].data.PWR_Attrs[].PWR_AttrValueCapabilities** (*int[]*) – List of supported sleep states
- **nids[].data.PWR_Attrs[].PWR_ReturnCode** (*int*) – Attribute probe specific result code, non-zero on failure
- **nids[].data.PWR_ReturnCode** (*int*) – Per-node attribute probe result code, non-zero on failure
- **nids[].data.PWR_ErrorMessages** (*string*) – Per-node attribute error message, or null
- **nids[].data.PWR_Messages** (*string*) – Per-node attribute info message, or null
- **nids[].data.PWR_MajorVersion** (*int*) – Remote API call major version number
- **nids[].data.PWR_MinorVersion** (*int*) – Remote API call minor version number

Status Codes

- **200** – Network API call success
- **400** – Bad Request

- **500** – Internal command failure
- **504** – Gateway Timeout

4.5 get_sleep_state_limit

The **get_sleep_state_limit** call returns the state number identifying the deepest allowable sleep on a per-node basis. The deepest allowable sleep state limit may be constrained from defaults using the **set_sleep_state_limit** call.

4.5.1 CLI Interface

capmc get_sleep_state_limit

-n, --nids <nid range list>
Specify a rangelist of NIDs for which to query the currently active sleep state limit. This is a required option.

```
$ capmc get_sleep_state_limit --nids=50
{
    "e": 0,
    "err_msg": "",
...
```

4.5.2 HTTP Interface

POST /capmc/cnctl
Example request:

```
POST /capmc/cnctl HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 234
Content-Type: application/json

{
  "nids": [
    50
  ],
  "data": {
    "PWR_Function": "PWR_ObjAttrGetValues",
    "PWR_Attrs": [
      {
        "PWR_AttrName": "PWR_ATTR_CSTATE_LIMIT"
      }
    ],
    "PWR_MajorVersion": 0,
    "PWR_MinorVersion": 1
  }
}
```

The request must **POST** a JSON object to the API server containing a remote API call function name, a single element list containing the appropriate attribute name, and the protocol version.

JSON Parameters

- **nids** (*int[]*) – User specified NID list, must **not** be empty

- **data** (*object*) – Remote API call input object
- **data.PWR_Function** (*string*) – Remote API call function name, the caller must specify "PWR_ObjAttrGetValues"
- **data.PWR_Attrs** (*object[]*) – Array of attribute name value pairs
- **data.PWR_Attrs[].PWR_AttrName** (*string*) – Attribute name, the caller must specify "PWR_ATTR_CSTATE_LIMIT"
- **data.PWR_MajorVersion** (*int*) – Remote API call major version number, the caller must specify "0"
- **data.PWR_MinorVersion** (*int*) – Remote API call minor version number, the caller must specify "1"

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "Success",
  "nids": [
    {
      "nid": 50,
      "data": {
        "PWR_Attrs": [
          {
            "PWR_AttrName": "PWR_ATTR_CSTATE_LIMIT",
            "PWR_AttrValue": 5,
            "PWR_ReturnCode": 0,
            "PWR_TimeNanoseconds": 777098381,
            "PWR_TimeSeconds": 1433536488
          }
        ],
        "PWR_ErrorMessages": null,
        "PWR_MajorVersion": 0,
        "PWR_Messages": null,
        "PWR_MinorVersion": 1,
        "PWR_ReturnCode": 0
      }
    }
  ]
}
```

JSON Parameters

- **e** (*int*) – Overall request status code, non-zero on partial success or failure
- **err_msg** (*string*) – Human readable error message
- **nids** (*object[]*) – Object array containing NID specific result data, each element represents a single NID
- **nids[].nid** (*int*) – NID number owning the returned attribute objects
- **nids[].data** (*object*) – Remote API call result object
- **nids[].data.PWR_Attrs** (*object[]*) – Array of attribute name value pairs

- **nids[].data.PWR_Attrs[].PWR_AttrName** (*string*) – Attribute name copied from original request
- **nids[].data.PWR_Attrs[].PWR_AttrValue** (*int[]*) – Returned attribute value
- **nids[].data.PWR_Attrs[].PWR_ReturnCode** (*int*) – Attribute probe specific result code, non-zero on failure
- **nids[].data.PWR_Attrs[].PWR_TimeSeconds** (*int*) – Elapsed seconds since the epoch, in UTC
- **nids[].data.PWR_Attrs[].PWR_TimeNanoseconds** (*int*) – Nanosecond timestamp component
- **nids[].data.PWR_ReturnCode** (*int*) – Per-node attribute probe result code, non-zero on failure
- **nids[].data.PWR_ErrorMessages** (*string*) – Per-node attribute error message, or null
- **nids[].data.PWR_Messages** (*string*) – Per-node attribute info message, or null
- **nids[].data.PWR_MajorVersion** (*int*) – Remote API call major version number
- **nids[].data.PWR_MinorVersion** (*int*) – Remote API call minor version number

Status Codes

- **200** – Network API call success
- **400** – Bad Request
- **500** – Internal command failure
- **504** – Gateway Timeout

4.6 set_sleep_state_limit

The **set_sleep_state_limit** call allows third-party software to constrain the deepest sleep state a node's processor(s) may enter. Valid values for sleep state limits are returned via the **get_sleep_state_limit_capabilities** call.

4.6.1 CLI Interface

capmc set_sleep_state_limit

-n, --nids <nid range list>
Specify a rangelist of NIDs for which to set the currently active sleep state limit. This is a required option.

-l, --limit <sleep state limit>
Specify the sleep state limit to put in place on the target NIDs. Valid limits are returned via the **get_sleep_state_limit_capabilities** call.

```
$ capmc set_sleep_state_limit --nids=50 --limit=4
{
    "e": 0,
    "err_msg": "",
...
```

4.6.2 HTTP Interface

POST /capmc/cnctl

Example request:

```
POST /capmc/cnctl HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 265
Content-Type: application/json

{
  "nids": [
    50
  ],
  "data": {
    "PWR_Function": "PWR_ObjAttrSetValues",
    "PWR_Attrs": [
      {
        "PWR_AttrName": "PWR_ATTR_CSTATE_LIMIT",
        "PWR_AttrValue": "4"
      }
    ],
    "PWR_MajorVersion": 0,
    "PWR_MinorVersion": 1
  }
}
```

The request must **POST** a JSON object to the API server containing a remote API call function name, a single element list containing the appropriate attribute name, requested sleep state limit value, and the protocol version.

JSON Parameters

- **nids** (*int[]*) – User specified NID list, must **not** be empty
- **data** (*object*) – Remote API call input object
- **data.PWR_Function** (*string*) – Remote API call function name, the caller must specify "PWR_ObjAttrValueCapabilities"
- **data.PWR_Attrs** (*object[]*) – Array of attribute name value pairs
- **data.PWR_Attrs[].PWR_AttrName** (*string*) – Attribute name, the caller must specify "PWR_ATTR_CSTATE_LIMIT"
- **data.PWR_Attrs[].PWR_AttrValue** (*string*) – Attribute value
- **data.PWR_MajorVersion** (*int*) – Remote API call major version number, the caller must specify "0"
- **data.PWR_MinorVersion** (*int*) – Remote API call minor version number, the caller must specify "1"

Example response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "Success",
  "nids": [
```

```
    {
      "nid": 50,
      "data": {
        "PWR_Attrs": [
          {
            "PWR_AttrName": "PWR_ATTR_CSTATE_LIMIT",
            "PWR_ReturnCode": 0
          }
        ],
        "PWR_ErrorMessages": null,
        "PWR_MajorVersion": 0,
        "PWR_Messages": null,
        "PWR_MinorVersion": 1,
        "PWR_ReturnCode": 0
      }
    }
  ]
}
```


JSON Parameters

- **e** (*int*) – Overall request status code, non-zero on partial success or failure
- **err_msg** (*string*) – Human readable error message
- **nids** (*object[]*) – Object array containing NID specific result data, each element represents a single NID
- **nids[].nid** (*int*) – NID number owning the returned attribute objects
- **nids[].data** (*object*) – Remote API call result object
- **nids[].data.PWR_Attrs** (*object[]*) – Array of attribute name value pairs
- **nids[].data.PWR_Attrs[].PWR_AttrName** (*string*) – Attribute name copied from original request
- **nids[].data.PWR_Attrs[].PWR_ReturnCode** (*int*) – Attribute set specific result code, non-zero on failure
- **nids[].data.PWR_ReturnCode** (*int*) – Per-node attribute set result code, non-zero on failure
- **nids[].data.PWR_ErrorMessages** (*string*) – Per-node attribute error message, or null
- **nids[].data.PWR_Messages** (*string*) – Per-node attribute info message, or null
- **nids[].data.PWR_MajorVersion** (*int*) – Remote API call major version number
- **nids[].data.PWR_MinorVersion** (*int*) – Remote API call minor version number

Status Codes

- **200** – Network API call success
- **400** – Bad Request
- **500** – Internal command failure
- **504** – Gateway Timeout

CHAPTER

FIVE

NODE ENERGY REPORTING

CAPMC API calls are provided such that external agents may query node energy consumption in various ways. The caller may query statistics by application id, job id, or simply use a list of NID numbers and a caller specified time window. Types of information returned may include aggregated energy usage on a set of nodes, energy usage per-node, or an energy accumulator point in time snapshot.

Warning: The **get_node_energy** and **get_node_energy_stats** API calls are resource intensive. Depending on system size and input parameters, those API calls may require several minutes to complete.

5.1 get_node_energy

Accumulated energy values for a set of nodes defined by a job (job_id), application (apid), list of NIDs (nids), or start and end time may be queried through the **get_node_energy** call. The input parameters are treated as conditions which are logically ANDed together. If an apid and start/end times are specified, then the values returned will be for the nodes involved in that apid during the interval specified by the start/end times.

Parameters returned include the following:

- Node count
- Duration of the interval, in seconds
- An array of (NID, energy) pairs

5.1.1 CLI Interface

capmc get_node_energy

-s, --start-time <YYYY-MM-DD HH:MM:SS>
Specify the starting time for the energy window calculation. This option must be used in conjunction with the **-e, –end-time** argument. If omitted, the starting time of the specified **apid** or **jobid** is used.

-e, --end-time <YYYY-MM-DD HH:MM:SS>
Specify the ending time for the energy window calculation. This option must be used in conjunction with the **-s, –start-time** argument. If omitted, the ending time of the specified **apid** or **jobid** is used.

-a, --apid <id>
Return statistics applicable to the NID list and application start & end times for the specified APLS id.

-j, --job <id>
Return statistics applicable to the NID list and application start & end times for the specified batch scheduler job id.

-n, --nids <nid range list>
Specify rangelist of NIDs to use for the energy window calculation. If omitted, the default is all NIDs matching other parameters.

5.1.2 HTTP Interface

POST /capmc/get_node_energy
Example Request:

```
POST /capmc/get_node_energy HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 122
Content-Type: application/json

{
  "start_time": "2015-06-03 14:07:32",
  "end_time": "2015-06-03 14:12:32",
  "nids": [
    23,
    24,
    25
  ]
}
```

Example Request: (By apid)

```
POST /capmc/get_node_energy HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 21
Content-Type: application/json

{
  "apid": 2977782
}
```

The request must **POST** a properly formatted JSON object to the API server. At a minimum, the request must contain a starting and ending time stamp with a NID list, an ALPS application ID, or a batch scheduler ID.

JSON Parameters

- **start_time** (*string*) – Optional, requested energy window sample start time
- **start_time** – Optional, requested energy window sample end time
- **nids** (*int[]*) – Optional, list of NIDs to use in energy counter query
- **apid** (*int*) – Optional, ALPS application ID
- **job_id** (*string*) – Optional, batch scheduler job id

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "",
```

```
  "nid_count": 3,
  "time": 300.0,
  "nodes": [
    {
      "nid": 23,
      "energy": 62623
    },
    {
      "nid": 24,
      "energy": 45454
    },
    {
      "nid": 25,
      "energy": 42870
    }
  ]
}
```

JSON Parameters

- **e** (*int*) – Error status, non-zero indicates statistics are unavailable
- **err_msg** (*string*) – Human readable error string indicating failure reason
- **nid_count** (*int*) – Number of nodes used in statistics query
- **time** (*double*) – Window width of energy calculation, in seconds
- **nodes** (*object[]*) – Object array containing node level energy info, each element represents a single node
- **nodes[].nid** (*int*) – NID number owning the returned energy accumulation
- **nodes[].energy** (*int*) – Accumulated energy computed over the requested time interval, specified in Joules

Status Codes

- **200** – Network API call success
- **504** – Gateway Timeout
- **500** – Internal command failure

5.2 get_node_energy_stats

Energy statistics for a set of nodes defined by a job (job_id), application (apid), list of NIDs (nodes), or start and end time may be queried through the **get_node_energy_stats** call. The input parameters are treated as conditions which are logically ANDed together. If an apid and start/end times are specified, then the statistics will be for the nodes involved in that apid during the interval specified by the start/end times. Both a temporal argument (apid, job_id, or start_time and end_time) and a component argument (apid, job_id, or NIDs) are required.

Parameters returned include the following:

- Total energy for the set
- Average energy for nodes in the set
- Standard deviation of energy for nodes in the set
- An ordered pair (NID, energy) for the minimum and maximum energy consuming nodes

- Duration of the interval, in seconds
- Node count

5.2.1 CLI Interface

capmc get_node_energy_stats

-s, --start-time <YYYY-MM-DD HH:MM:SS>
Specify the starting time for the energy window calculation. This option must be used in conjunction with the **-e, –end-time** argument. If omitted, the starting time of the specified **apid** or **jobid** is used.

-e, --end-time <YYYY-MM-DD HH:MM:SS>
Specify the ending time for the energy window calculation. This option must be used in conjunction with the **-s, –start-time** argument. If omitted, the ending time of the specified **apid** or **jobid** is used.

-a, --apid <id>
Return statistics applicable to the NID list and application start & end times for the specified APLS id.

-j, --job <id>
Return statistics applicable to the NID list and application start & end times for the specified batch scheduler job id.

-n, --nids <nid range list>
Specify rangelist of NIDs to use for the energy window calculation. If omitted, the default is all NIDs matching other parameters.

5.2.2 HTTP Interface

POST /capmc/get_node_energy_stats
Example Request:

```
POST /capmc/get_node_energy_stats HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 122
Content-Type: application/json

{
  "start_time": "2015-06-03 14:07:32",
  "end_time": "2015-06-03 14:12:32",
  "nids": [
    23,
    24,
    25
  ]
}
```

The request must **POST** a properly formatted JSON object to the API server. At a minimum, the request must contain a starting and ending time stamp with a NID list, an ALPS application ID, or a batch scheduler ID.

JSON Parameters

- **start_time** (*string*) – Optional, requested energy window sample start time
- **start_time** – Optional, requested energy window sample end time
- **nids** (*int[]*) – Optional, list of NIDs to use in energy counter query
- **apid** (*int*) – Optional, ALPS application ID

- **job_id** (*string*) – Optional, batch scheduler job id

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "",
  "time": 300.0,
  "nid_count": 3,
  "energy_total": 150947,
  "energy_avg": 50315.666666666664,
  "energy_std": 8766.303072307936,
  "energy_max": [
    23,
    62623
  ],
  "energy_min": [
    25,
    42870
  ]
}
```

JSON Parameters

- **e** (*int*) – Error status, non-zero indicates statistics are unavailable
- **err_msg** (*string*) – Human readable error string indicating failure reason
- **time** (*double*) – Window width of energy statistics calculation, in seconds
- **energy_total** (*int*) – Sum of per node energy, in Joules
- **energy_avg** (*double*) – Per node average energy, in Joules
- **energy_std** (*double*) – Standard deviation, in Joules
- **energy_max** (*int[]*) – Ordered list identifying NID number followed by maximum observed node energy consumption
- **energy_min** (*int[]*) – Ordered list identifying NID number followed by minimum observed node energy consumption

Status Codes

- **200** – Network API call success
- **504** – Gateway Timeout
- **500** – Internal command failure

5.3 get_node_energy_counter

Energy counters for a set of nodes defined by a job (job_id), application (apid), or list of NIDs (nids) may be queried through the **get_node_energy_counter** call. The parameters apid, jobid, and nids are treated as selectors for the set of nodes to query. If an apid or jobid are supplied, the running counters for each node in that aprun or job will be returned. If a list of NIDs is supplied, then the counters for the nodes corresponding to the supplied NID list will be returned. One and only one of apid, jobid, or NIDs list must be specified. If a time value is specified, then the query

will retrieve the energy counters at or very near the specified time (if available, within one second). Otherwise, the most recent energy counter value will be returned.

Note: This API call returns a free running energy counter for each of the target NIDs. In order to be meaningful, such as when computing average power or energy consumed over a time interval, multiple calls must be made such that the caller can perform calculations based on the difference in returned energy counter values.

5.3.1 CLI Interface

capmc get_node_energy_counter

-t, --time <YYYY-MM-DD HH:MM:SS>
Specify the desired energy sample point in time. If omitted, the energy point in time will be taken as the most recent available sample in the last 30 seconds on a per node basis.

-a, --apid <id>
Return energy counters applicable to the NID list of the specified APLS id.

-j, --job <id>
Return energy counters applicable to the NID list of the specified batch scheduler job id.

-n, --nids <nid range list>
Specify rangelist of NIDs to retreive energy counters. If omitted, the default is all NIDs matching other parameters.

5.3.2 HTTP Interface

POST /capmc/get_node_energy_counter
Example Request:

```
POST /capmc/get_node_energy_counter HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 78
Content-Type: application/json

{
  "time": "2015-06-03 14:07:32",
  "nids": [
    23,
    24,
    25
  ]
}
```

The request must **POST** a properly formatted JSON object to the API server. At a minimum, the request must contain a time stamp with a NID list, an ALPS application ID, or a batch scheduler ID.

JSON Parameters

- **time** (*string*) – Optional, requested energy sample start time
- **nids** (*int[]*) – Optional, list of NIDs to use in energy counter query
- **apid** (*int*) – Optional, ALPS application ID
- **job_id** (*string*) – Optional, batch scheduler job id

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "",
  "nid_count": 3,
  "nodes": [
    {
      "nid": 24,
      "energy_ctr": 14802226,
      "time": "2015-06-03 14:07:32.886126-05"
    },
    {
      "nid": 23,
      "energy_ctr": 10196418,
      "time": "2015-06-03 14:07:32.022648-05"
    },
    {
      "nid": 25,
      "energy_ctr": 13649114,
      "time": "2015-06-03 14:07:32.886126-05"
    }
  ]
}
```

JSON Parameters

- **e** (*int*) – Error status, non-zero indicates statistics are unavailable
- **err_msg** (*string*) – Human readable error string indicating failure reason
- **nid_count** (*int*) – Number of nodes used in statistics query
- **nodes** (*object[]*) – Object array containing node level energy info, each element represents a single node
- **nodes[].nid** (*int*) – NID number owning the returned energy counter
- **nodes[].energy_cntr** (*int*) – Point in time energy accumulator value, specified in Joules
- **nodes[].time** (*string*) – Time stamp of returned energy value, includes fractional seconds and timezone offset

Status Codes

- **200** – Network API call success
- **504** – Gateway Timeout
- **500** – Internal command failure

CHAPTER

SIX

SYSTEM LEVEL MONITORING

CAPMC API calls are provided such that external agents may monitor near real time system level power consumption and energy usage. If a time window is specified, then historical records may be retrieved. Data may be returned in aggregate or constituent form containing information relating to total system or per-cabinet components, respectively. Additionally, a mechanism is provided for a system administrator to convey intent or other operational parameters, such as a maximum system power limit, or unreported static power overhead, to third-parties.

6.1 get_system_parameters

Read-only parameters such as expected worst case system power consumption, static power overhead, or an administratively defined system wide power limit may be returned via the **get_system_parameters** call. Returned values are used to convey intent between the system administrator and external agents with respect to target power limits and other operational parameters. The returned parameters are strictly informational.

6.1.1 CLI Interface

capmc get_system_parameters

The command line interface has no arguments. Result text is returned via standard out.

```
$ capmc get_system_parameters
{
  "e": 0,
  "err_msg": "",
  "power_cap_target": 0,
..
```

6.1.2 HTTP Interface

POST /capmc/get_system_parameters

Example Request:

```
POST /capmc/get_system_parameters HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 2
Content-Type: application/json

{}
```

The request must **POST** an empty JSON object to the API server. This command takes no arguments.

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "",
  "power_cap_target": 0,
  "power_threshold": 0,
  "static_power": 10700
}
```

JSON Parameters

- **e** (*int*) – Error status, non-zero indicates failure
- **err_msg** (*string*) – Human readable error string indicating failure reason
- **power_cap_target** (*int*) – Administratively defined upper limit on system power
- **power_threshold** (*int*) – System power level, which if crossed, will result in Cray management software emitting over power budget warnings
- **static_power** (*int*) – Additional static system wide power overhead which is unreported, specified in watts

Status Codes

- **200** – Network API call success
- **500** – Internal command failure

Note: The **power_threshold** parameter is a mechanism in which a system administrator may convey intent to an energy aware scheduler. It defines a target, or a desired worst case system power usage. It is the responsibility of the scheduler to enforce the nessecary energy aware scheduling policy in order to comply with the administrators intent.

6.2 get_system_power

The **get_system_power** call returns system level power information including the average, minimum, and maximum values observed over a user specified time interval. If no arguments are given, then information is returned for an interval consisting of the most recent 10 seconds.

6.2.1 CLI Interface

capmc get_system_power

-s, --start-time <YYYY-MM-DD HH:MM:SS>
Specify sampling window start time. If omitted, the default start time is 10 seconds into the past.

-w, --window-len <seconds>
Specify the sampling window length in seconds. The valid range is defined by the interval [2,3600]. If omitted, the default value of 10 seconds is used.

```
$ capmc get_system_power --start-time="2015-06-01 13:45:59" --window-len=30
{
  "start_time": "2015-06-01 13:45:59",
  "window_len": 30,
  "avg": 17488,
...
```

6.2.2 HTTP Interface

POST /capmc/get_system_power

Example Request:

```
POST /capmc/get_system_power HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 62
Content-Type: application/json

{
  "start_time": "2015-06-01 13:45:59",
  "window_len": 30
}
```

The request must **POST** a properly formatted JSON object to the API server. The command takes up to two optional arguments which identify a start time and window length.

JSON Parameters

- **start_time** (*string*) – Optional, sampling window start time
- **window_len** (*int*) – Optional, sampling window length

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "",
  "window_len": 30,
  "start_time": "2015-06-01 13:45:59",
  "avg": 17488,
  "max": 17661,
  "min": 17340
}
```

JSON Parameters

- **e** (*int*) – Error status, non-zero indicates statistics are unavailable
- **err_msg** (*string*) – Human readable error string indicating failure reason
- **window_len** (*int*) – Window length in seconds in which the statistics have been computed, may be different from the requested value
- **start_time** (*string*) – Window sample start time in **YYYY-MM-DD HH:MM:SS** format or symbolic constant **CURRENT_TIMESTAMP**, may be different from the requested value
- **avg** (*int*) – Average system power computed over the time window

- **max** (*int*) – Peak system power observed over the time window
- **min** (*int*) – Min system power observed over the time window

Status Codes

- **200** – Network API call success
- **500** – Internal command failure

6.3 get_system_power_details

The **get_system_power_details** call returns per cabinet power information including the average, minimum, and maximum values observed over a user specified time interval. If no arguments are given, then information is returned for an interval consisting of the most recent 10 seconds.

6.3.1 CLI Interface

capmc get_system_power_details

-s, --start-time <YYYY-MM-DD HH:MM:SS>
Specify sampling window start time. If omitted, the default start time is 10 seconds into the past.

-w, --window-len <seconds>
Specify the sampling window length in seconds. The valid range is defined by the interval [2,3600]. If omitted, the default value of 10 seconds is used.

```
$ capmc get_system_power_details --start-time="2015-06-01 13:45:59" --window-len=30
{
  "e": 0,
  "err_msg": "",
  "window_len": 30,
  "start_time": "2015-06-03 12:47:07",
    "cabinets": [
      {
        "avg": 17432.033333333333,
        "max": 17730,
 ...
```

6.3.2 HTTP Interface

POST /capmc/get_system_power_details
Example Request:

```
POST /capmc/get_system_power_details HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 62
Content-Type: application/json

{
  "start_time": "2015-06-03 12:47:07",
  "window_len": 30
}
```

The request must **POST** a properly formatted JSON object to the API server. The command takes up to two optional arguments which identify a start time and window length.

JSON Parameters

- **start_time** (*string*) – Optional, sampling window start time
- **window_len** (*int*) – Optional, sampling window length

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "",
  "window_len": 30,
  "start_time": "2015-06-03 12:47:07",
    "cabinets": [
      {
        "avg": 17432.033333333333,
        "max": 17730,
        "min": 17069,
        "x": 0,
        "y": 0
      },
      {
        "avg": 17456.033333333333,
        "max": 17738,
        "min": 17060,
        "x": 1,
        "y": 0
      }
    ]
}
```

JSON Parameters

- **e** (*int*) – Error status, non-zero indicates statistics are unavailable
- **err_msg** (*string*) – Human readable error string indicating failure reason
- **window_len** (*int*) – Window length in seconds in which the statistics have been computed, may be different from the requested value
- **start_time** (*string*) – Window sample start time in **YYYY-MM-DD HH:MM:SS** format or symbolic constant **CURRENT_TIMESTAMP**, may be different from the requested value
- **cabinets** (*object[]*) – Object array containing cabinet level power statistics, each element represents a single cabinet
- **cabinets[].avg** (*double*) – Average cabinet power computed over the time window
- **cabinets[].max** (*int*) – Peak cabinet power observed over the time window
- **cabinets[].min** (*int*) – Min cabinet power observed over the time window
- **cabinets[].x** (*int*) – Cabinet X coordinate, column address
- **cabinets[].y** (*int*) – Cabinet Y coordinate, row address

Status Codes

- **200** – Network API call success
- **500** – Internal command failure

CHAPTER

SEVEN

UTILITY FUNCTIONS

CAPMC contains several utility functions which may be used to query information such as node to component name mapping, system partition membership, or a node's role assignment. The client utility may also function in an HTTP pass-through mode, effectively allowing a caller to specify custom input payloads not possible using command line arguments alone. This mode of operation offers maximum flexibility while allowing the caller to utilize capmc's built in authorization mechanism.

7.1 get_nid_map

Some commands, specifically commands relating to power capping controls, require a NID list which does not contain service nodes. Other use cases may require associating a geographic component name to NID number. In such cases, calling applications may query a node's role or cname using the **get_nid_map** call. The call returns a list objects where each element represents a single node. Each object in the list contains a numeric id identifying the NID number, geographic component name, and the operational role.

7.1.1 CLI Interface

capmc get_nid_map

-n, --nids <nid range list>
: Specify rangelist of NIDs to retreive NID number to component name mapping and assigned role. If omitted, the default is all NIDs.

```
$ capmc get_nid_map --nids=1,140,141
{
  "e": 0,
  "err_msg": "",
  "nids": [
    {
      "cname": "c0-0c2s3n1",
      "nid": 141,
      "role": "compute"
    },
    {
 ...
```

7.1.2 HTTP Interface

POST /capmc/get_nid_map
: **Example Request**:

```
POST /capmc/get_nid_map HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 45
Content-Type: application/json

{
  "nids": [
    1,
    140,
    141
  ]
}
```

The request must **POST** a properly formatted JSON object to the API server. The command takes a single argument which identifies a target NID list.

JSON Parameters

- **nids** (*int[]*) – User specified list, or empty array for all NIDs.

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "",
  "nids": [
    {
      "cname": "c0-0c2s3n1",
      "nid": 141,
      "role": "compute"
    },
    {
      "cname": "c0-0c2s3n0",
      "nid": 140,
      "role": "compute"
    },
    {
      "cname": "c0-0c0s0n1",
      "nid": 1,
      "role": "service"
    }
  ]
}
```

JSON Parameters

- **e** (*int*) – Error status, non-zero indicates failure
- **err_msg** (*string*) – Human readable error string
- **nids** (*object[]*) – Object array containing node specific mapping information, each element represents a single NID
- **nids[].nid** (*int*) – NID number owning the returned geographical component name and service role
- **nids[].cname** (*string*) – Geographical component name

- **nids[].role** (*string*) – Currently assigned service role, may be one of "compute" or "service"

Status Codes

- **200** – Network API call success
- **500** – Internal command failure

7.2 get_partition_map

Some commands, specifically commands relating to frequency and sleep state controls, require a NID list which does not cross system partition boundaries. In such cases, calling applications may query partition membership using the **get_partition_map** call. The call returns a list objects where each element represents a single system partition. Each object in the list contains a numeric id identifying the partition number and the corresponding member NIDs.

7.2.1 CLI Interface

capmc get_partition_map

The command line interface has no arguments. Result text is returned via standard out.

```
$ capmc get_partition_map
{
  "e": 0,
  "err_msg": "",
  "partitions": [
    {
      "nids": [
        0,
        3,
..
```

7.2.2 HTTP Interface

POST /capmc/get_partition_map

Example Request:

```
POST /capmc/get_partition_map HTTP/1.1
Host: smw.example.com
Accept: application/json
Content-Length: 2
Content-Type: application/json

{}
```

The request must **POST** an empty JSON object to the API server. This command takes no arguments.

Example Response:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "e": 0,
  "err_msg": "",
  "partitions": [
```

```
        {
          "partition": 0,
          "nids": [
            0,
            3,
            166,
            167
          ]
        }
      ]
    }
```

JSON Parameters

- **e** (*int*) – Error status, non-zero indicates failure
- **err_msg** (*string*) – Human readable error string
- **partitions** (*object[]*) – Object array containing partition membership information, each element represents a single system partition
- **partitions[].partition** (*int*) – Partition number owning the returned NID membership list
- **partitions[].nids** (*int[]*) – Partition member NID list

Status Codes

- **200** – Network API call success
- **500** – Internal command failure

7.3 json

The client script, capmc, provides a function which allows a caller to construct and send a JSON formatted object to a user specified API handler. This command allows a caller to utilize the capmc authorization mechanism while constructing their own input parameter objects directly. This command is implemented purely within the client side script.

7.3.1 CLI Interface

capmc json

-r, --resource </capmc/api/path>
Post a JSON text data structure acquired from standard input to the specified resource on the server. JSON text input is limited to 10MB.

```
$ capmc --resource=/capmc/node_status < /path/to/input/data.json
```

/capmc

Symbols

E

G

J

N

S