MDaaS UUT REST API

This is a conceptual description of each of the MDaaS UUT API endpoints. Please see the Documentation/schema.json OpenAPI definition for the exact methods, data structures, and value types used.

It is recommended to not call endpoints more frequently than necessary so as to not adversely affect test.

PUT /mdaas/environment

- Delivers information about the environment in which the MDaaS node is operating, from the caller to the MDaaS node, including:
  - The caller's identity, version and location
  - The caller's idea of the SKU being tested
  - The caller's identifier for this test run (for correlating test data and logs)
  - Rack Manager IP address
  - TOR service port IP address
- Applications which are using MDaaS should make a `PUT /mdaas/environment` call as soon as they determine a node is operational.
- This PUT call returns in the response body a copy of the values submitted, so the request and response JSON formats are the same.
- An example request, and its identical successful '202 Accepted' response (with added JavaScript-style comments. *Note that the sample values here may not look like real-world data*):

```
{
    // always use this library's MdaasUutApi.Version.ApiDllVersion constant for api_dll_version:
    "api_dll_version": "0.1.0 (mdaas_uut_api@180e136; 2020/7/9)",
    "caller": "BSL",
    "caller_version": "1.0.2",
    "caller_testrun": "9ec17ff7-b8fd-4b8f-aff2-70bd1098d2e0",
    "site": "Foxconn Juarez",
    // fields above here are required, below here are optional
    "rm_ip": "192.168.8.42",
    "tor_ip": "192.168.8.43",
    "blade_sku": "M1039570-002 C2010",
    "rack_sku": "Azure_Compute_Optimized_Gen6.1_104"
}
```

- A '400 Bad Request' response resulting from missing fields:

```
{
    "error": "Required arguments missing: caller_version"
}
```

  Note that all error responses by the API use this simple data format.

- A possible '500 Internal Server Error' response:

```
{
    "error": "Unable to find config directory"
}
```

GET /mdaas/status

- Returns information on the status of testing on the node, including:
    - whether the test is starting, running, finished, or failed
    - the current test item, total number of items, and a history of test items completed
    - when testing is finished, whether the node passed or failed
    - whether the upload of the blob file (system data and log file zip) succeeded or failed
    - additional information about the system
- Possible values of the `status` field are:
    - `not started` - the API is up, but the test system has not yet started
    - `starting` - gathering initial system data
    - `running` - test system is running within the test flow itself
    - `finishing` - gathering final system data and upload
    - `finished` - test run finished normally
    - `crashed` - test run failed to finish, should not happen
    - `exception` - system was not able to start, should not happen
- Typically the system should reach a `status` or `running` within five minutes of booting.
- The `azure_upload` field indicates whether the upload of the blob file (system data and log file zip) to Azure was successful. Its possible values are:
    - `pending` - while the test flow is still running
    - `succeeded` - the upload to Azure was successful
    - `failed` - the upload of the blob failed, and the caller should use the `/mdaas/blob` endpoint below to capture the blob file for offline delivery
- A successful '200 OK' response:

```
{
    "status": "running",  // see above for possible values and meanings
    "desc": "CPU/Memory Stress",  // description of the currently running test
    "items": 8,   // total number of test items
    "current": 4, // index of current test item, 0 is not yet started, 1 is first item
    "test_result": "pending",  // once 'status' is 'finished', this will be 'pass' or 'fail'
    "azure_upload": "pending", // see above for possible values and meanings
    "history": [  // a summary of completed test items
        {
            "status": "pass",         // 'pass' or 'fail'
            "desc": "Clear BMC SEL",  // test item description
            "err_code": "None",       // MDaaS error code if fail
            "err_msg": "None"         // failure description
        },
        {

            "status": "pass",
            "desc": "Disk S.M.A.R.T Test",
            "err_code": "None"
```

```
            "err_code": "None",
            "err_msg": "None"
        },
        {
            "status": "fail",
            "desc": "Check BMC SEL with Whitelist",
            "err_code": "BMC_CHK_SEL_ERR",
            "err_msg": "Found SEL error:{ 3 | 06/23/2020 | 03:48:04 | Unknown BMC Health |  | Deasserte
        }
    ],
    "detail": {  // additional status values that may be of interest
        "mdaas_mode": "bsl",  // the kernel boot flag used to specify the test mode
        "test_start_time": "2020-07-10T00:22:54.879168Z",  // ISO8601 timestamp used to uniquely
                           // identify the test run
        "ntp": "synced",      // whether the NTP time sync 'synced', 'failed' or is 'pending'
        "azure": "up"         // if connectivity to Azure is 'up', 'down' or 'unknown'
    }
}
```

- A possible '500 Internal Server Error' response:

```
{
    "error": "Could not read sequencer state: Could not run srv-diag: file not found"
}
```

GET /mdaas/log_list

- Returns a list of log files available, as URI paths for the `/mdaas/log` endpoint.
- A successful '200 OK' response:

```
{
    "log_options": [
        "/mdaas/log/diag",  // always-present alias for diag.log
        "/mdaas/log/api",   // always-present alias for api.log
        "/mdaas/log/abc.log",
        "/mdaas/log/api.log",  // log for the REST API itself
        "/mdaas/log/diag.log", // the main log for the test process, should always be available
                               // once testing starts
        "/mdaas/log/foo.log"
    ]
}
```

GET /mdaas/log/<log-filename-or-id>

- Returns the contents of the specified log. Use the `/mdaas/log_list` endpoint above to determine which logs are available.
- A successful '200 OK' response:

```
{
    "body": "2020-07-10 00:22:47,931 auto_start.<module>:68 INFO--\nauto_start.py started\n\n2020-07-10
}
```

- A '404 Not Found' response for a non-existent log request:

```
{
    "error": "Log `notfound.log` could not be found. Directory: /root/logs/notfound.log"
}
```

GET /mdaas/flow

- Returns a list of test items. This will only be available once the test is running.
- A successful '200 OK' response:

```
{
    "items": [
        {
            "class": "BMCClearSEL",
            "description": "Clear BMC SEL"
        },
        {
            "class": "MSTStress",
            "description": "MST Stress"
        },
        {
            "class": "BMCCheckSELWhiteList",
            "description": "Check BMC SEL with Whitelist"
        },
        {
            "class": "MCECheck",
            "description": "Check MCE Error"
        },
        // ...
    ]
}
```

- A '404 Not Found' response when the test is not yet running:

```
{
    "error": "No test flow file defined yet, please try again later"
}
```

GET /mdaas/version

- Returns MDaaS version information
- A successful '200 OK' response:

```
{
    "version": "MDaaS Release-89",  // the principal version identifier of the MDaaS image
    "detail": {  // additional version information, keys subject to change
        "image_version": "Mariner OS 1.0.MM4",
        "release_version": "Release-89",
        "build_date": "2020-06-05T00:06:35Z",
        "kernel_version": "3.1.4-340.x86_64",
        "kernel_flags": "initrd=linux/initrd-mdaas-2020-06-10-rel89-internal.cpio.gz rw loglevel=3 cons

        "uut_api_version": "20200610.2",
        "srvdiag_version": "20200603.7",
        "si": "internal"
```

```
        }
    }
```

GET /mdaas/blob_list

- Return a list of blob files available for download
- This returns a list of blob files (zip files containing all system data and logs from the test run) available for download, as URI paths for the `/mdaas/blob` endpoint. The list may be empty if no test was run or if the system is still starting up. If any blob exists, there will always be a special entry of `/mdaas/blob/latest` in the list, which will map to the most recently-created blob.
- A successful '200 OK' response:

```
{
    "blob_options": [
        "/mdaas/blob/12344556_2020-07-09T13.21.02_START.zip",
        "/mdaas/blob/12344556_2020-07-09T13.21.02_END.zip",
        "/mdaas/blob/latest"
    ]
}
```

- A '404 Not Found' response for a non-existent blob request:

```
{
    "error": "File does not exist. File: /root/logs/12344556_2004-01-01T00.00.00_START.zip"
}
```

GET /mdaas/blob/<blob-filename-or-id>

- Download (as a typical HTTPS file download) a blob file (zip file containing all system data and logs from the test run). The special endpoint `/mdaas/bloblist/latest` returns the most recent blob created.
- If MDiag is not able to upload its blob file to Azure, then the caller may download the blob file and convey it by other means (e.g. manual upload). Typically, if the caller sees in the response from the status endpoint, a `status` of `finished` or `crashed`, and an `azure_upload` of `failed`, they should use the `/mdaas/blob/latest` endpoint to download and save the most recent blob.
- A successful '200 OK' response is not a JSON object, but rather a standard HTTPS file download.