# Task 1

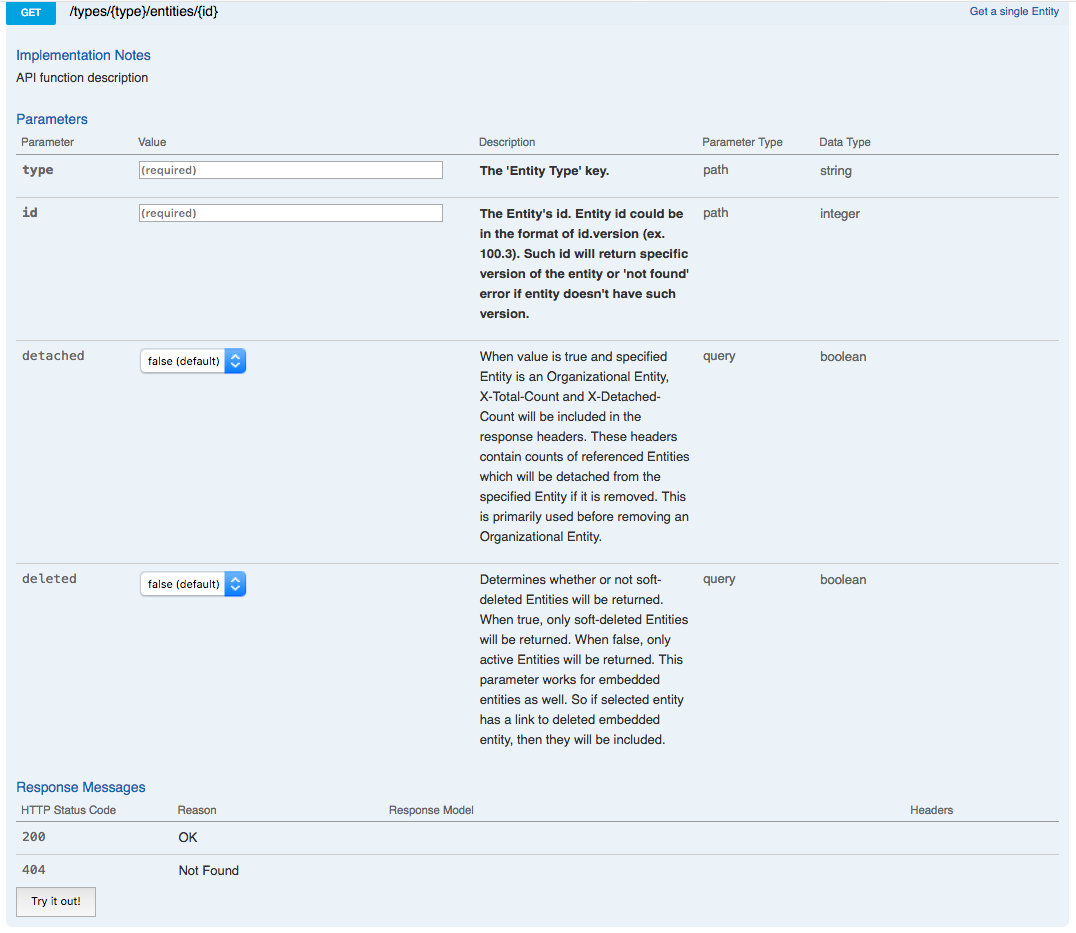
Please find below the part of Swagger documentation (<https://swagger.io/swagger-ui/>) for the API of Media Manager component. It describes GET API call that reads the entity and POST API call to create the data entity.

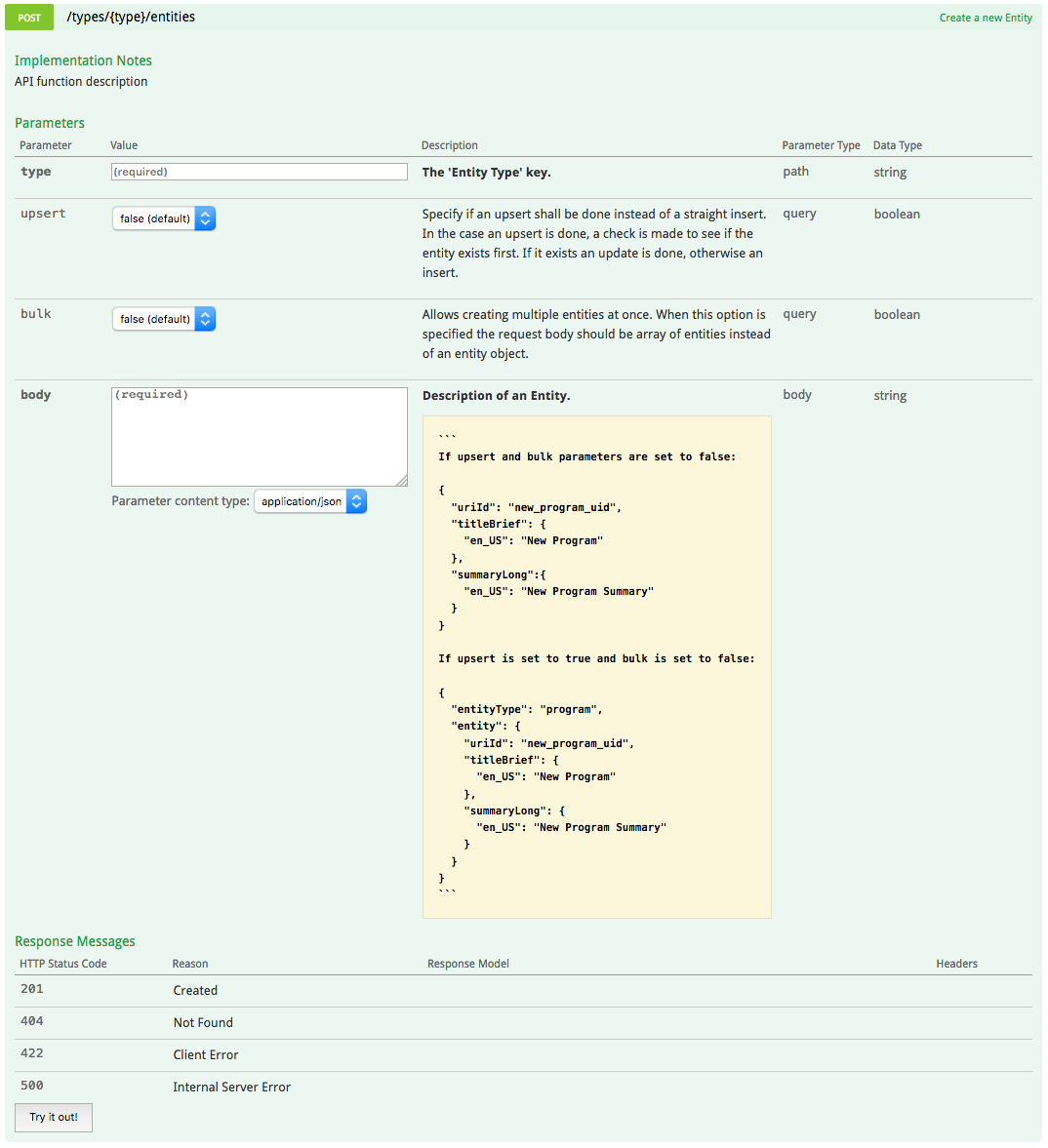
Describe the necessary HTTP requests to validate these APIs. Assume that products with IDs from 1 to 10 already exist in the component.

Note: no need to create performance or load tests.

Fill in the table below (see examples prefilled). Feel free to add comments/explanations.

|  |  |  |
| --- | --- | --- |
| **Request** | **Assertion(s)** | **Comments** |
| GET /types/program/entities/1 | Response header is correct  Response code is 200  Body contains valid JSON  JSON is according to the entity schema  Body data is correct | Positive Scenario |
| GET /types/program/entities/1?detached=false&deleted=false | Response header is correct  Response code is 200  Body contains valid JSON  JSON is according to the entity schema  Body data is correct and same as previous test in response header and body | Positive Scenario |
| GET /types/program/entities/1?detached=true | Response header is correct  Response code is 200  Body contains valid JSON  JSON is according to the entity schema  Body data is correct  If Organizational entity should return in response header X-Total-Count and X-Deatached-Count | Positive Scenario |
| GET /types/program/entities/1?deleted=true | Response header is correct  Response code is 200  Body contains valid JSON  JSON is according to the entity schema  Body data is correct  Only soft-deleted entities will be returned | Positive Scenario |
| GET /types/program/entities/1?detached=true&deleted=true | Response header is correct  Response code is 200  Body contains valid JSON  JSON is according to the entity schema  Body data is correct  Only soft-deleted entities will be returned  If Organizational entity should return in response header X-Total-Count and X-Deatached-Count | Positive Scenario |
| GET /types/program/entities/1?detached=false&deleted=true | Response header is correct  Response code is 200  Body contains valid JSON  JSON is according to the entity schema  Body data is correct | Positive Scenario |
| GET /types/program/entities/1?detached=true&deleted=false | Response header is correct  Response code is 200  Body contains valid JSON  JSON is according to the entity schema  Body data is correct | Positive Scenario |
| GET /types/program/entities/1.1 | Response header is correct  Response code is 200  Body contains valid JSON  JSON is according to the entity schema  Body data is correct version | Positive Scenario \*\* Assuming this entity has different versions 1 |
| GET /types/program/entities/2.2 | Response header is correct  Response code is 404  Body contains valid JSON “Error Message” | Negative Scenario \*\* Assuming this entity has not version 2 |
| GET /types/program/entities/11 | Response header is correct  Response code is 404  Body contains valid JSON “Error Message” | Negative Scenario |
| GET /types/program/entities/x | Response header is correct  Response code is 404  Body contains valid JSON “Error Message” | Negative Scenario |
| POST /types/program/entities Body: valid JSON according to schema  {  "uriId": "new\_program\_uid",  "titleBrief":{  "en\_US":"New Program"  },  "summaryLong":{  "en\_US":"New Program Summary"  }  } | Response header is correct  Response code is 201  Body is empty | Positive Scenario  \*\* Single record |
| POST /types/program/entities?upsert=false&bulk=false Body: valid JSON according to schema  {  "uriId": "new\_program\_uid1",  "titleBrief":{  "en\_US":"New Program1"  },  "summaryLong":{  "en\_US":"New Program Summary1"  }  } | Response header is correct  Response code is 201  Body is empty | Positive Scenario  \*\* Single record |
| POST /types/program/entities?upsert=true Body: valid JSON according to schema  {  “entityType”: “program”,  “entity”:{  "uriId": "new\_program\_uid",  "titleBrief":{  "en\_US":"New Program in English"  },  "summaryLong":{  "en\_US":"New Program Summary in English"  }  }  } | Response header is correct  Response code is 201  Body is empty | Positive Scenario  \*\* Single record and update existing record |
| POST /types/program/entities?upsert=true Body: valid JSON according to schema  {  “entityType”: “program”,  “entity”:{  "uriId": "new\_program\_uidX",  "titleBrief":{  "en\_US":"New Program in English"  },  "summaryLong":{  "en\_US":"New Program Summary in English"  }  }  } | Response header is correct  Response code is 201  Body is empty | Positive Scenario  \*\* Single record with new data |
| POST /types/ non-program/entities?upsert=true Body: valid JSON according to schema  {  “entityType”: “non-program”,  “entity”:{  "uriId": "new\_program\_uid",  "titleBrief":{  "en\_US":"New Program in English"  },  "summaryLong":{  "en\_US":"New Program Summary in English"  }  }  } | Response header is correct  Response code is 404  Body is not empty, showing error message | Negative Scenario  \*\* Single record  entityType is not exists |
| POST /types/program/entities?upsert=true Body: valid JSON according to schema  {  “entityType”: “non-program”,  “entity”:{  "uriId": "new\_program\_uid",  "titleBrief":{  "en\_US":"New Program in English"  },  "summaryLong":{  "en\_US":"New Program Summary in English"  }  }  } | Response header is correct  Response code is 422  Body is not empty, showing error message | Negative Scenario  \*\* Single record  entityType is wrong in body |
| POST /types/program/entities?bulk=true Body: valid JSON according to schema  [  {  "uriId": "new\_program\_uid2",  "titleBrief":{  "en\_US":"New Program2"  },  "summaryLong":{  "en\_US":"New Program Summary2"  }  },  {  "uriId": "new\_program\_uid3",  "titleBrief":{  "en\_US":"New Program3"  },  "summaryLong":{  "en\_US":"New Program Summary3"  }  }  ] | Response header is correct  Response code is 201  Body is empty | Positive Scenario  \*\* Multiple record |
| POST /types/program/entities?bulk=false Body: valid JSON according to schema  [  {  "uriId": "new\_program\_uid2",  "titleBrief":{  "en\_US":"New Program2"  },  "summaryLong":{  "en\_US":"New Program Summary2"  }  },  {  "uriId": "new\_program\_uid3",  "titleBrief":{  "en\_US":"New Program3"  },  "summaryLong":{  "en\_US":"New Program Summary3"  }  }  ] | Response header is correct  Response code is 500  Body is not empty, showing error message | Negative Scenario  \*\* Multiple record |
| POST /types/program/entities Body: empty JSON according to schema  {} | Response header is correct  Response code is 500  Body is not empty, showing error message | Negative Scenario |





# Task 2

Here is the scenario we’d like to test.

* All necessary components (mentioned below) are setup and running properly.
* There is a QTS system. It reads XML files which are put to the folder. As soon as file is read and processed (i.e. consumed) file gets deleted. File should be consumed within 60 seconds.
* When file is consumed QTS starts the process called ‘**consumer**’ in Workflow Engine. Process typically runs for 10-60 seconds. It should not take longer than 180 seconds.
* The result of the process is published to Media Manager component. If the data is published to Media Manager the scenario is assumed successful.

Component interfaces are the following:

* Folder that QTS is expecting new files to be

/qts\_watch\_folder/

* XML file describes the program and has the following format   
  <program><name>*program\_name*</name></program>

*program\_name* is a parameter. It is used to identify program.

* API call to Workflow Engine to check the processes status is  
  GET http://wfe/processes  
  The response is the list of currently running processes, associated program name and process status. Multiple processes can be run for various programs at the same time. When process is running the status is ‘running’, when process is completed the status is ‘completed’.  
  Response is JSON with the following structure:

|  |
| --- |
| [{  "name": "process\_name",  "program\_id": "program\_name",  "status": "running"  },  {  "name": "process\_name2",  "program\_id": "program\_name2",  "status": "running"  }, {  "name": " process\_name3",  "program\_id": "program\_name3",  "status": "completed"  }  ] |

* Media Manager content API is   
  GET http://mm/entities/program/{program\_name}  
  Responses are:  
  200 if the entity exists in the DB  
  404 if the entity does not exist

Please create the positive test case for this scenario.

Start with creating a file and putting it to the folder. Verify the process is running. When process is completed verify that data is created in Media Manager.

The result should be the either

* Robot Framework keyword(s) and supporting python helpers if needed
* Pytest code

Pay attention on how you organize the code to files/folders and modules/libraries. Python and Robot framework code should be valid (should compile) but we don’t expect it to actually run. Feel free to add comments/explanations to the code.