For Developers: Developer API - Data Capsule API

This is a set of APIs exposed by the web services and back-end scripts. This is for advanced users that want to implement their own version of the data capsule, and will likely not concern HTRC Data Capsule users.

Overview

The system consists of three major components -- a web front end, a RESTful web service and scripts.

- Front-end Layer: The web front end is where the user can create VM, query VM status, switch VM mode, and kill VM.
- Web Service Layer: The RESTful web service takes user request and translates the requests to command lines to manage VM. The command lines are backed by the scripts. The web service also maintains VM status in a database. The web service can also be organized in a distributed manner, i.e., a request router dispatches user requests to web service in different machines.
- Hypervisor Layer: There are scripts used to create VM, configure network properly, switch VM mode, and kill VM. They are invoked by the web service with proper input parameters.

This document presents APIs for both web service layer and hypervisor layer. These APIs are for internal usage and there is no open access to them.

RESTful APIs and scripts APIs

Create VM

RESTful call	/createvm
Purpose	create a vm
Method	POST

Input Parameter	imageid=<imageid>&loginusername=<vncusername>&loginpassword=<vncpassword>&vcpu=<vcpu>&memory=<memory_in_MB>&type=<type (DEMO/RESEARCH)>&consent=<consent>&full_access=<access_for_full_data(true/false) >&title=<project_title>&desc_nature=<nature_of_project>&desc_requirement=<desc_requirement>&desc_links=<desc_links>&desc_outside_data =<desc_outside_data>&rr_data_files=<rr_data_files>&rr_result_usage=<rr_result_usage>
Response	200 - {vmid:<uuid>}, 400 - no Image is found, exceeds allocated resources per user 500 - internal error

Below is an example how the service will call the script.

```
bash create_vm_script.sh --image /var/image_file --vcpu 2 --mem 1024 --wdir /var/instance/uuid/ --vnc port --
ssh port --loginid vmloginid --loginpwd vmloginpwd
```

where

- /var/image_file is the original image which should be copied and pasted for each VM
- 2 is the number of cpu for the vm
- 1024 is the number of memory for the vm
- /var/instance/uuid/ is the working directory for vm with uuid
- ports for vnc and ssh are assigned permanently
- login username and password for this vm

Response

- succeeded (0) or failed (non-zero value). The script also prints out an error message. Use different return codes for different errors.

Update VM

RESTful call	/updatevm
Purpose	Update a vm / Accept or Reject full-access request
Method	POST
Input Parameter	type=<type>&consent=<consent>&full_access=<full_access>&title=<title>&desc_nature=<desc_nature>&desc_requirement=<desc_requirement>&desc_links=<desc_links>&desc_outside_data=<desc_outside_data>&rr_data_files=<rr_data_files>&rr_result_usage=<rr_result_usage>&desc_shared=<desc_shared>
Response	200 - {vmid:<uuid>}, 400 - no vm is found, permission denied, invalid conversion type, invalid capsule state 500 - internal error

Launch a VM

RESTful call	/launchvm
Purpose	launch a vm
Method	POST
Input Parameter	vmid=<vmid>

Response	200 - operation is in progress, 400 - no vm is found, permission denied, invalid capsule state 500 - internal error

Below is an example how the service will call the script, where policy file is the configuration file for firewall.

```
bash launch_vm_script.sh --wdir /var/instance/uuid/ --mode {maintain|secure} --policy /path/to/policyfile
```

Response

- succeeded (0) or failed (non-zero value). The script also prints out an error message. Use different return codes for different errors.

Query a VM's status

RESTful call	/show
Purpose	show vm status
Method	POST
Input Parameter	vmid=<uuid> (if no vmid is present, info of all vm relative to this username will be returned)

Response	200:
	```
	{
	        "status":[
	                        {
	                                "vncloginId":<vnc_userId>",
	                                "vcpus":<vcpu>,
	                                "vmInitialLoginPassword":<vm_pw>,
	                                "publicip":<ip>,
	                                "sshport":<port>,
	                                "vncport":<port>,
	                                "imageName":<imagename>,
	                                "vncloginPassword":<vnc_pw>,
	                                "memSize":<mem>,
	                                "volumeSize":<volumesize>,
	                                "vmInitialLoginId":<login_id>,
	                                "pubKeyExists":<public_key_exists>,
	                                "mode":<mode>,
	                                "vmid":<vmid>,
	                                "state":<state>,
	                    "desc_links": <desc_links>,
	                    "desc_nature": <desc_nature>,
	                    "desc_outside_data": <desc_outside_data>,
	                    "desc_requirement": <desc_requirement>,
	                    "desc_shared": <desc_shared>,
	                    "rr_data_files": <rr_data_files>,
	                    "rr_result_usage": <rr_result_usage>,
	                    "title": <title>,
	                    "tou": true/false,
	                    "type": <type>,
	                    "pubKeyExists": true/false,
	                                 "consent": true/false,
	                    "full_access": null/true/false,
	                "roles": [
	                        {
	                            "email": <user_email>,
	                            "full_access": <user's_full_access>,
	                            "guid": <user_guid>,
	                            "role": <"OWNER-CONTROLLER"/"OWNER"/"CONTROLLER"/"SHAREE">
	                            "tou": true
	                        }
	                    ]
	                }
	        ]
	}
	```
	 400 - no vm is found, permission denied
	
	500 - internal error

Below is an example how the service will call the script.

```
bash query_vm_status_script.sh --wdir /var/instance/uuid/
```

Response

- succeeded (0) or failed (non-zero value).
- If succeeded, prints status response with key/value separated by colon, and each value pair separated by a newline. For example,

```
mode: security status: running publicip: 192.168.1.1
```

- The script also prints out an error message. Use different return codes for different errors.

Switch a VM’s mode

RESTful call	/switchvm
Purpose	switch vm mode: modes are “maintenance” and “secure”.
Method	POST
Input Parameter	vmid=<uuid>&mode=<mode>
Response	200 - operation is in progress, 400 - no vm is found, permission denied 500 - internal error

Below is an example how the service will call the script.

```
bash switch_vm_script.sh --wdir /var/instance/uuid/ --mode secure --policy /path/to/policy/file
```

response

- succeeded or failed

Stop a VM

RESTful call	/stopvm
Purpose	stop vm : moves VM from running state to stopped state. This is the same effect as turning a "machine" off by pushing it's power button. A stopped VM can be restarted via /launchvm.
Method	POST
Input Parameter	vmid=<uuid>
Response	200 - operation is in progress, 400 - no vm is found, permission denied 500 - internal error

Below is an example how the service will call the script.

```
bash stop_vm_script.sh --wdir /var/instance/uuid/
```

response

- succeeded or failed

Delete a VM

RESTful call	/deletevm
Purpose	delete vm. All files under vm working directory (VM image, secure volume, etc) are permanently deleted.
Method	POST

Input Parameter	vmid=<uuid> More secure option: Add in a user's authentication token that is verified by the script.
Response	200 - operation is in progress, 400 - no vm is found, permission denied 500 - internal error

Below is an example how the service will call the script.

```
bash delete_vm_script.sh --wdir /var/instance/uuid/
```

response

- succeeded or failed

List VMs

RESTful call	/listvms
Purpose	Give key information of the VMs that are not deleted
Method	GET
Input Parameter	none
Response	200:

```
{
        "vmsInfo":[
                        {
                                "created_at":<created_at>,
                                "host":<ip>,
                                "memorySize":<memorySize>,
                                "numCPUs":<numCPUs>,
                                "numHostCPUCores":<numHostCPUCores>,
                                "numHostMemoryGB":<numHostMemoryGB>,
                                "vmmode":<mode>,
                                "vmid":<vmid>,
                                "vmState":<state>,
                    "type": <type>,
                "roles": [
                        {
                            "email": <user_email>,
                            "full_access": <user's_full_access>,
                            "guid": <user_guid>,
                            "role": <"OWNER-CONTROLLER"/"OWNER"/"CONTROLLER"/"SHAREE">
                            "tou": true
                        }
                    ],
                                "userEmail":null,
                                "username":null
                }
        ]
}
```

500 - internal error

Update User Key

RESTful call	/updateuserkey
Purpose	update/delete user's SSH key from database and associated VMs
Method	POST : Update SSH key of the user in the database and update all VMs(update the ssh key) that the user has access to PUT : If vmid != null : update VM with all sharees SSH keys, if vmid == null : for all VMs, update the SSH keys of all sharees DELETE : Remove user's SSH key from all the VMs that this user has access to, and Set the SSH key to null in the database
Input Parameter	POST : pubkey=<pubkey> PUT : vmid=<vmid> DELETE : none
Response	200 - successful completion 400 - no vm is found 500 - internal error

Update User Email

RESTful call	/updateuseremail
Purpose	Update user email in the database
Method	POST
Input Parameter	No input parameters, the email is extracted from the JWT token
Response	200 - successful 400 - no user is found 500 - internal error

Update User Key

RESTful call	/updateusertou
Purpose	Retrieve or update user's TOU parameter. This is the parameter that indicates whether the user has accepted general/per-capsule TOU agreement.
Method	POST : If vmid == null, update general tou parameter, if vmid != null, update per-capsule tou parameter GET : Retrieve whether you has accepted general TOU agreement
Input Parameter	POST : vmId=<vmId>,tou=<true/false> GET : username is extracted from the JWT token
Response	200 - successful completion, {tou: true/false} 400 - no user/vm is found 500 - internal error

Migrate Vm

RESTful call	/migratevm
Purpose	Migrate VM from one host to another host
Method	POST : Migrate VM to another host PUT : Alter only the database entries after VM is migrated manually
Input Parameter	POST : vmId=<vmId>,host=<host> PUT :vmId=<vmId>,host=<host>,sshport=<sshport>,vncport=<vncport>
Response	200 - successful completion 400 - no user/vm is found, invalid host name, no port resources found, invalid VM state 500 - internal error

Add VM Sharee

RESTful call	/addsharees
Purpose	Add a sharee to the VM by owner
Method	POST
Input Parameter	vmId=<vmId>,sharees=<[{"giud":<guid>,"email":<email>}]>,desc_shared=<reason_for_sharing_research-full_capsule>
Response	200 - successful completion 400 - no user/vm is found, permission denied, invalid sharees input, invalid VM state/mode, full-access request pending, exceeds no. of sharees limit 500 - internal error

Remove VM Sharee

RESTful call	/deletesharees
Purpose	Remove a sharee/(s) from the VM by owner
Method	POST
Input Parameter	vmId=<vmId>,sharees=<guid1,guid2>
Response	200 - successful completion 400 - no user/vm is found, permission denied, invalid sharees input, invalid VM state, invalid role(should be SHAREE) 500 - internal error

Manage Controller

RESTful call	/managecontroller
Purpose	Owner or Controller Delegate control of VM to another sharee/ Owner revoke control of VM from the controller
Method	POST

Input Parameter	vmId=<vmId>,controller=<guid_of_sharee/controller>,action=<DELEGATE/REVOKE>
Response	200 - successful completion 400 - no user/vm is found, permission denied, invalid VM state, invalid transition of roles 500 - internal error

Exit Sharee

RESTful call	/exitsharee
Purpose	Let sharee to exit from the VM when in the CONTRIBUTOR role
Method	POST
Input Parameter	vmId=<vmId>
Response	200 - successful completion 400 - no user/vm is found, permission denied, invalid VM state 500 - internal error

Add Support User

RESTful call	/addsupportuser
Purpose	Add support user to VM
Method	POST
Input Parameter	vmId=<vmId>
Response	200 - successful completion 400 - no user/vm is found, permission denied, invalid VM state 500 - internal error

Images

Share A Capsule Image

RESTful call	/shareimage
Purpose	Share an image of a capsule. Capsule should be in the SHUTDOWN state.
Method	POST

Input Parameter	vmId=<capsule_id>, imageName=<image_name_user_specifies> , imageDescription=<image_description_user_provides>, public=<image_visibility(true/false)>
Response	200 - successful completion Response body example: {"imageId":"b5be823e-409f-4d2c-8951-64c20b2c6452","updatedAt":"2021-05-06 12:04:30","imageDescription":"this is a new image"," loginUserName":null,"loginPassWord":null,"createdAt":"2021-05-06 12:04:30","public":false,"imageStatus":"SHARE_PENDING","sourceVM":" 694d1e94-4b3e-486a-9241-24fae8b76270","imageName":"ubuntu-694d1e94-4b3e-486a-9241-24fae8b76270","imagePath":"host.image. dirb5be823e-409f-4d2c-8951-64c20b2c6452.img","owner":"b2f7d37a-deb5-4f67-b6a1-abe47f8e2c65"} 400 - no user/vm is found, permission denied, invalid VM state, exceeded user's allocated image left quota. 500 - internal error

Activate Image

RESTful call	/activateimage
Purpose	Transfer the image status to ACTIVE from SHARE_PENDING. This method will be used by the host servers to activate the image once the image file is synched with all the DC host nodes.
Method	PUT
Input Parameter	imageId=<image_id>
Response	200 - successful completion Response body example: {"imageId":"b5be823e-409f-4d2c-8951-64c20b2c6452","updatedAt":"2021-05-06 12:04:30","imageDescription":"this is a new image"," loginUserName":null,"loginPassWord":null,"createdAt":"2021-05-06 12:04:30","public":false,"imageStatus":"ACTIVE","sourceVM":"694d1e94-4b3e-486a-9241-24fae8b76270","imageName":"ubuntu-694d1e94-4b3e-486a-9241-24fae8b76270","imagePath":"host.image.dirb5be823e-409f-4d2c-8951-64c20b2c6452.img","owner":"b2f7d37a-deb5-4f67-b6a1-abe47f8e2c65"} 400 - no image is found, invalid image state 500 - internal error

Update Image

RESTful call	/shareimage
Purpose	Update image name, image description and image visibility. Image should be in ACTIVE status
Method	PUT

Input Parameter	imageId=<image_id> , imageName=<image_name_user_specifies> , imageDescription=<image_description_user_provides>, public=<image_visibility(true/false)>
Response	200 - successful completion Response body example: {"imageId":"b5be823e-409f-4d2c-8951-64c20b2c6452","updatedAt":"2021-05-06 12:04:30","imageDescription":"this is a new image"," loginUserName":null,"loginPassWord":null,"createdAt":"2021-05-06 12:04:30","public":false,"imageStatus":"ACTIVE","sourceVM":"694d1e94-4b3e-486a-9241-24fae8b76270","imageName":"ubuntu-694d1e94-4b3e-486a-9241-24fae8b76270","imagePath":"host.image.dirb5be823e-409f-4d2c-8951-64c20b2c6452.img","owner":"b2f7d37a-deb5-4f67-b6a1-abe47f8e2c65"} 400 - no image is found, permission denied, invalid image state , image name is not available 500 - internal error

Check Image Name

RESTful call	/checkimagename
Purpose	Check image name availability
Method	GET
Input Parameter	imageName=<image_name>
Response	200 - Image name available 400 - Image name not available 500 - internal error

List All Images

RESTful call	/listallimages
Purpose	list all images that user have access (Images in all status)
Method	GET

Response	200 - successful completion Response body example: {"imageInfo":[{"imageId":"fb398248-530f-4d26-963f-a1f0968fe09e","updatedAt":"2021-05-04 12:57:08","imageDescription":"this is a new image","loginUserName":null,"loginPassWord":null,"createdAt":"2021-05-04 12:57:08","public":false,"imageStatus":"DELETED","sourceVM":"54352a1a-e2b4-4a95-93b0-e747d2f9cb91","imageName":"ubuntu-123","imagePath":"host.image.dirfb398248-530f-4d26-963f-a1f0968fe09e.img","owner":"b2f7d37a-deb5-4f67-b6a1-abe47f8e2c65"},{"imageId":"fde25f21-da72-4752-8ca4-366fa94ccb5e","updatedAt":"2021-04-30 13:47:58","imageDescription":"This is the image name used to test image with no password.","loginUserName":"dcuser","loginPassWord":"NULL","createdAt":"2021-04-30 13:47:58","public":true,"imageStatus":"DELETED","sourceVM":null,"imageName":"ubuntu-16-04-no-password","imagePath":"/home/htrcprod/images/ubuntu-16-04-09122018.img","owner":"8bca447f-6692-46e7-8c04-802cd911fb36"},{"imageId":"fe9a028d-6682-4a13-92ef-e42f9cec13fd","updatedAt":"2021-04-30 13:47:58","imageDescription":"This is Ubuntu 16.04 image with no password. This was updated by 08-12-2018 release.","loginUserName":"dcuser","loginPassWord":"NULL","createdAt":"2021-04-30 13:47:58","public":true,"imageStatus":"ACTIVE","sourceVM":null,"imageName":"ubuntu-16-04-with-sample-volumes","imagePath":"/home/htrcprod/resources-dev/images/ubuntu-16-04-with-volumes-09122018.img","owner":"8bca447f-6692-46e7-8c04-802cd911fb36"}]} 500 - internal error

List My Images

RESTful call	/listmyimages
Purpose	list user's images which are in ACTIVE and SHARE_PENDING state
Method	GET
Response	200 - successful completion Response body example: {"imageInfo":[{"imageId":"9263e3aa-408d-42c3-9687-6a89dc988188","updatedAt":"2021-05-04 13:20:14","imageDescription":"this is a new image","loginUserName":null,"loginPassWord":null,"createdAt":"2021-05-04 13:20:14","public":false,"imageStatus":"ACTIVE","sourceVM":"54352a1a-e2b4-4a95-93b0-e747d2f9cb91","imageName":"ubuntu-112","imagePath":"host.image.dir9263e3aa-408d-42c3-9687-6a89dc988188.img","owner":"b2f7d37a-deb5-4f67-b6a1-abe47f8e2c65"},{"imageId":"aa0b7ed3-a028-4de7-aa0d-7b97f0f78193","updatedAt":"2021-05-05 15:02:03","imageDescription":"this is a new image","loginUserName":null,"loginPassWord":null,"createdAt":"2021-05-05 15:02:03","public":false,"imageStatus":"SHARE_PENDING","sourceVM":"54352a1a-e2b4-4a95-93b0-e747d2f9cb91","imageName":"ubuntu-444","imagePath":"host.image.diraa0b7ed3-a028-4de7-aa0d-7b97f0f78193.img","owner":"b2f7d37a-deb5-4f67-b6a1-abe47f8e2c65"},{"imageId":"b5be823e-409f-4d2c-8951-64c20b2c6452","updatedAt":"2021-05-06 12:04:30","imageDescription":"this is a new image","loginUserName":null,"loginPassWord":null,"createdAt":"2021-05-06 12:04:30","public":false,"imageStatus":"SHARE_PENDING","sourceVM":"694d1e94-4b3e-486a-9241-24fae8b76270","imageName":"ubuntu-694d1e94-4b3e-486a-9241-24fae8b76270","imagePath":"host.image.dirb5be823e-409f-4d2c-8951-64c20b2c6452.img","owner":"b2f7d37a-deb5-4f67-b6a1-abe47f8e2c65"},{"imageId":"c7a4c180-93e5-485a-bd0d-099b875bd6ec","updatedAt":"2021-05-04 12:52:17","imageDescription":"this is a new image","loginUserName":null,"loginPassWord":null,"createdAt":"2021-05-04 12:52:17","public":true,"imageStatus":"ACTIVE","sourceVM":"54352a1a-e2b4-4a95-93b0-e747d2f9cb91","imageName":"ubuntu-samitha","imagePath":"host.image.dirc7a4c180-93e5-485a-bd0d-099b875bd6ec.img","owner":"b2f7d37a-deb5-4f67-b6a1-abe47f8e2c65"}]} 500 - internal error

List Active Images

RESTful call	/listactiveimages
Purpose	list all ACTIVE images that user have access
Method	GET
Response	200 - successful completion Response body example: {"imageInfo":[{"imageId":"9263e3aa-408d-42c3-9687-6a89dc988188","updatedAt":"2021-05-04 13:20:14","imageDescription":"this is a new image","loginUserName":null,"loginPassWord":null,"createdAt":"2021-05-04 13:20:14","public":false,"imageStatus":"ACTIVE","sourceVM":"54352a1a-e2b4-4a95-93b0-e747d2f9cb91","imageName":"ubuntu-112","imagePath":"host.image.dir9263e3aa-408d-42c3-9687-6a89dc988188.img","owner":"b2f7d37a-deb5-4f67-b6a1-abe47f8e2c65"},{"imageId":"b5be823e-409f-4d2c-8951-64c20b2c6452","updatedAt":"2021-05-06 12:04:30","imageDescription":"this is a new image","loginUserName":null,"loginPassWord":null,"createdAt":"2021-05-06 12:04:30","public":false,"imageStatus":"ACTIVE","sourceVM":"694d1e94-4b3e-486a-9241-24fae8b76270","imageName":"ubuntu-694d1e94-4b3e-486a-9241-24fae8b76270","imagePath":"host.image.dirb5be823e-409f-4d2c-8951-64c20b2c6452.img","owner":"b2f7d37a-deb5-4f67-b6a1-abe47f8e2c65"},{"imageId":"c7a4c180-93e5-485a-bd0d-099b875bd6ec","updatedAt":"2021-05-04 12:52:17","imageDescription":"this is a new image","loginUserName":null,"loginPassWord":null,"createdAt":"2021-05-04 12:52:17","public":true,"imageStatus":"ACTIVE","sourceVM":"54352a1a-e2b4-4a95-93b0-e747d2f9cb91","imageName":"ubuntu-samitha","imagePath":"host.image.dirc7a4c180-93e5-485a-bd0d-099b875bd6ec.img","owner":"b2f7d37a-deb5-4f67-b6a1-abe47f8e2c65"},{"imageId":"f3c2a554-5197-492f-a0fe-03e4330a5938","updatedAt":"2021-04-30 13:47:58","imageDescription":"This is Ubuntu 16.04 image with no password. This was updated by 08-12-2018 release.","loginUserName":"dcuser","loginPassWord":"NULL","createdAt":"2021-04-30 13:47:58","public":true,"imageStatus":"ACTIVE","sourceVM":null,"imageName":"ubuntu-16-04","imagePath":"/home/htrcprod/resources-dev/images/ubuntu-16-04-09122018.img","owner":"8bca447f-6692-46e7-8c04-802cd911fb36"},{"imageId":"fe9a028d-6682-4a13-92ef-e42f9cec13fd","updatedAt":"2021-04-30 13:47:58","imageDescription":"This is Ubuntu 16.04 image with no password. This was updated by 08-12-2018 release.","loginUserName":"dcuser","loginPassWord":"NULL","createdAt":"2021-04-30 13:47:58","public":true,"imageStatus":"ACTIVE","sourceVM":null,"imageName":"ubuntu-16-04-with-sample-volumes","imagePath":"/home/htrcprod/resources-dev/images/ubuntu-16-04-with-volumes-09122018.img","owner":"8bca447f-6692-46e7-8c04-802cd911fb36"}]} 500 - internal error

Get Image Information

RESTful call	/getimage
Purpose	Get information of the image that user have access
Method	GET
Input Parameter	imageId=<image_id>

Response	200 - successful completion Response body example: {"imageId":"7bea06e5-930a-4248-bc86-6ae231cfc3a0","updatedAt":"2021-05-04 13:11:12","imageDescription":"this is a new image"," loginUserName":null,"loginPassWord":null,"createdAt":"2021-05-04 13:11:12","public":false,"imageStatus":"DELETED","sourceVM":" 54352a1a-e2b4-4a95-93b0-e747d2f9cb91","imageName":"ubuntu-456","imagePath":"host.image.dir7bea06e5-930a-4248-bc86-6ae231cfc3a0.img","owner":"b2f7d37a-deb5-4f67-b6a1-abe47f8e2c65"} 400 - no image is found, permission denied 500 - internal error

Image Deletion Process

Image deletion process goes in two steps. First user can request to delete the image using ***/requestimagedelete*** API call. This will only change the image status to DELETE_PENDING state. That'll make the image not available for the capsule creation. Then using a nightly cron job actual image deletion happens using the ***/deleteimage*** API call. This two step implementation is to avoid errors in the capsule creation process if the same image was used for it. i.e. At the capsule creation the image file will be copied from the main images directory to the capsule's directory in the host server. This process takes few seconds. If the image which user tries to delete is used by another user(if the image is public) to create a capsule at the same time , that image file will be in the middle of coping from the images directory to the capsule's directory. If the actual deletion happens at that moment it'll disturb the capsule creation process and capsule will go into the ERROR state.

Request Image Deletion

RESTful call	/requestimagedelete
Purpose	Request to delete the image. Successful completion will update the image status to DELETE_PENDING and restore user's image left quota. Image should be in the ACTIVE state at the request time.
Method	PUT
Input Parameter	imageId=<image_id>
Response	200 - successful completion 400 - no image is found, permission denied, Image is not in the ACTIVE state 500 - internal error

Delete Image

RESTful call	/deleteimage
Purpose	Delete the image. Image should be in the DELETE_PENDING state at the request time. This method will be used internally(for nightly cronjob) to delete all the DELETE_PENDING images.
Method	DELETE
Input Parameter	imageId=<image_id>
Response	200 - successful completion 400 - no image is found, permission denied, Image is not in the DELETE_PENDING state 500 - internal error