**Swordfish API – Developer Documentation**

**Contents:**

1. Requirements …………………… 02
2. Overview …………………… 02
   1. Dynamic Resources …………………… 02
   2. Template and API …………………… 02
   3. Resourcemanager.py …………………… 03
   4. Utils.py …………………… 04
3. Implementing Dynamic structure …………………… 04
   1. Writing template files …………………… 05
   2. Writing API files …………………… 06
4. Swordfish Functionality …………………… 08

1. **Requirements:**

**Python version:** 2.7, 3.1, 3.6

**Ubuntu:** 16.4v

1. **Overview:**

**2.1 Dynamic Resources:**

Dynamic resources can have following properties:

* It can create resources and sub-resources using template structure via POST operation.
* Can change the properties of a specific resource.
* GET POST PUT & DELETE operations are used.

**2.2 Mockups and templates:**

Mockups are just an example view or static view of possible configuration. Mockups will simply provide the overview of all swordfish models.

Template is a base structure for creating new element in a collection / Sub Collection. We can create members of collection using template structure. Template structure is designed based on latest SNIA Swordfish latest Schema V1.0.5

Template file divides in to two types of sections.one is “template declaration” and other is “ function”

**2.3 Resource\_manager.py file**:

* The redfish root structure is defined in Resourcemanager.py file. So now developer has to make some suggested changes to run swordfish emulator.
* Redfish emulator already contains static and dynamic resources in resourcemanager.py file. It already contains static resources from ./redfish/static directory which is similar/copy to mockup data.
* Resource Manager class can load static resources and dynamic resources.
* Swordfish developer has to attach API for dynamic resources.
* All the API files are stored in ./api\_emulator/swordfish directory.

Fig A: Imports all the api files

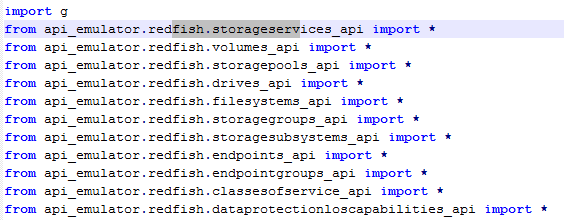
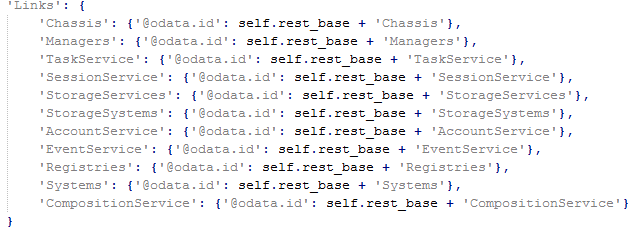


Fig B: Attaching API for dynamic resources

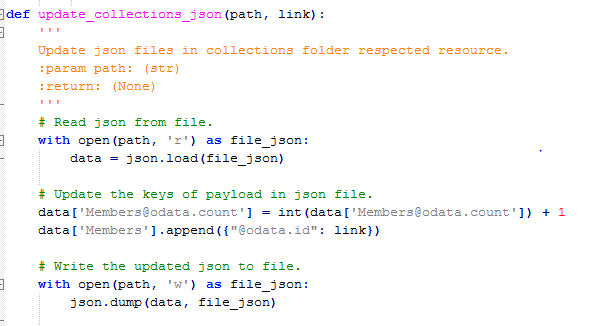


Fig C: Adding Collections to



**2.4 Utils.py file:**

* Added new service for members count.



1. **Implementing Dynamic structure**

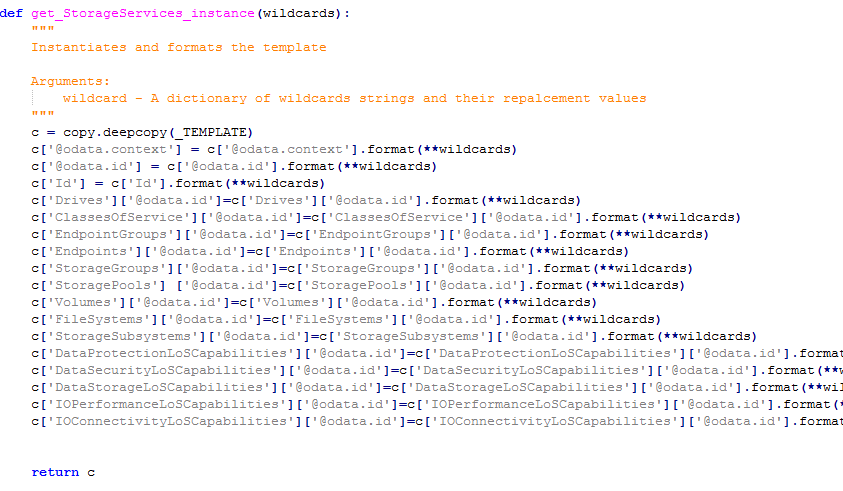
* The static resources are in /api\_emulator/redfish/static/.. Folder, these are just a simple copy.
* The dynamic resource is in /Resource/...Folder. We can specify any path to declare the dynamic resources.
* Developer has to create a template file and api files for dynamic structure.
* The file in template directory and api directory are to be paired appropriately.

**3.1 Writing Template Files:**

* Template file directory is /api\_emulator/swordfish/template
* Template declaration is as similar to json file (shown in below figure). All the properties for a particular resource are to be added by referring the updated schemas released by SNIA-Swordfish.
* Template declaration also contains substitution fields
* Each template contains similar fields {rb} & {id} which are declared in below function are replaced by root\_base(/redfish/v1) and ID .

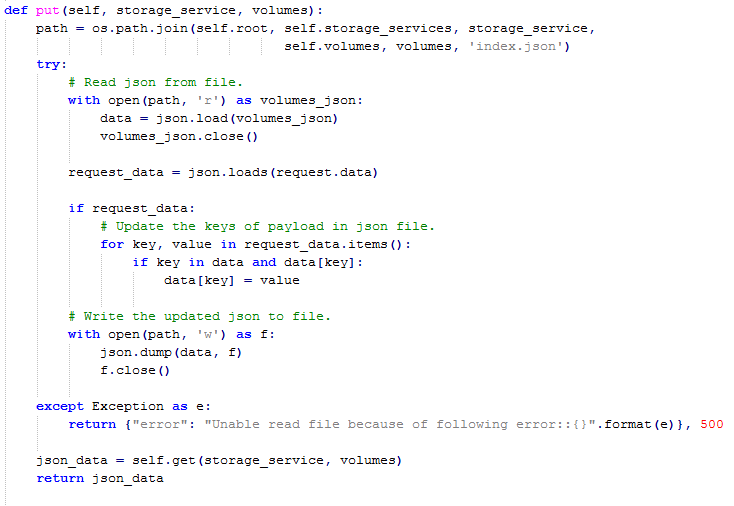


* Below figure contains function, which is used to call the instance of resource.
* We are using wildcards for replacing the values.
* Using deep-copy is to make copy of template declaration.
* {rb} –{rest-base} is default set to {redfish/v1}



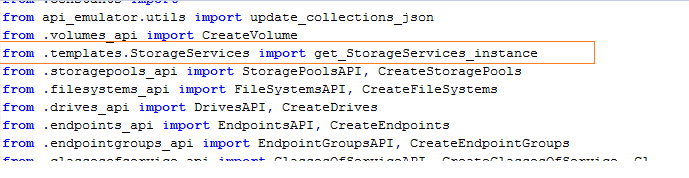
**3.2 Writing API Files:**

* We have to import the above template function to add a new element in to collection.
* The API file commonly used to retrieve data form remote systems. Using API , we can make request to web server and retrieve data that is needed. In emulator API code will interacting with all collections and member resources.
* For all Collection-API’s , GET and POST methods are defined, post command will create a sub resource API to the interface.
* For all API’S there are API classes and collection classes.
* The http functions GET, POST, PUT and DELETE are declared based on requirements.
* The Post function can create resources ,update members and members.id
* Post function can also attach API’s of sub resources and creates instance of sub-resources.
* PUT function is used to edit specific properties in a collection.

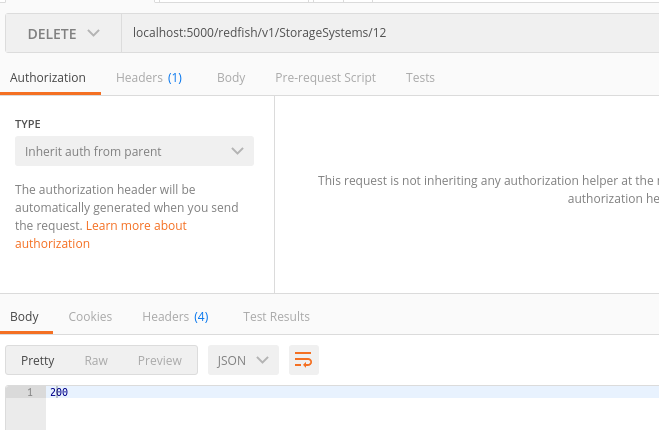


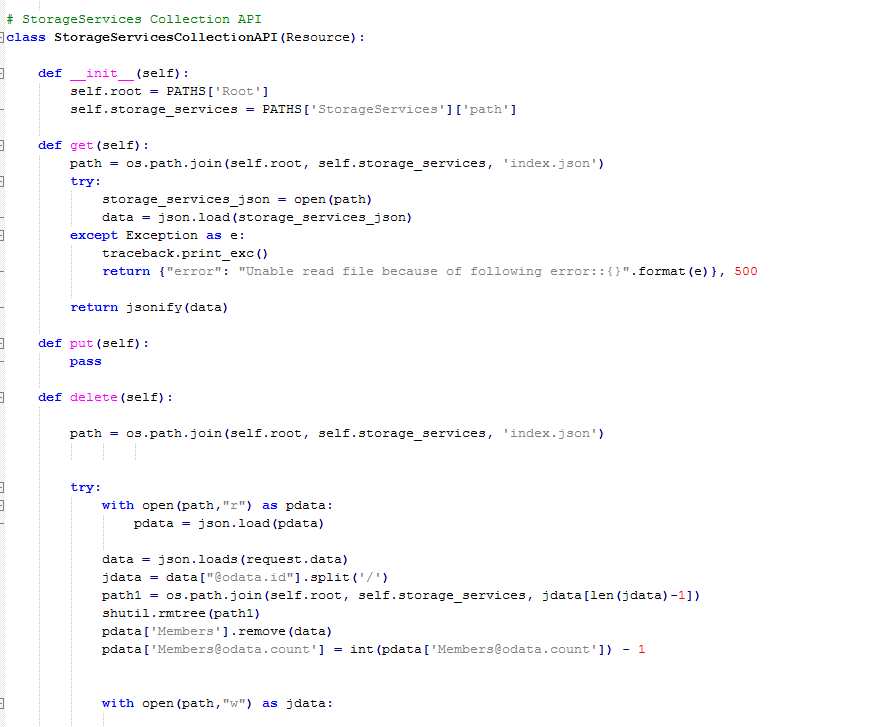
1. **Swordfish Functionality**:-

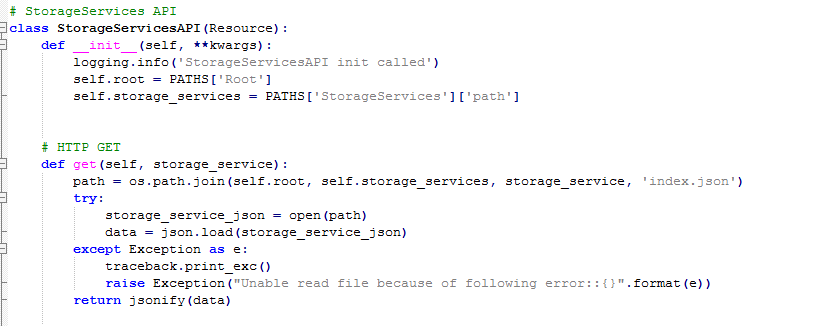
* For Example, If we want to create StorageServices as a dynamic resource using template and API files.
* StorageServices is a member of collection. we have to create template and API files with similar name.
  + Template file path is /api\_emulator/swordfish/template/StorageServices.py
  + API file path /api\_emulator/swordfish/storageservices\_api.py
* **Template** file contains two sections as shown in figure.
  + Template declaration
  + Function ("get\_StorageServices\_instance")



* Storageservices\_api.py file contains two different classes. Api class and collection class.
* StorageServices collectionAPI file
* By using url (/redfish/v1/StorageServices) we can “GET” members and collection data
* We can delete an item from collection using “Delete”.



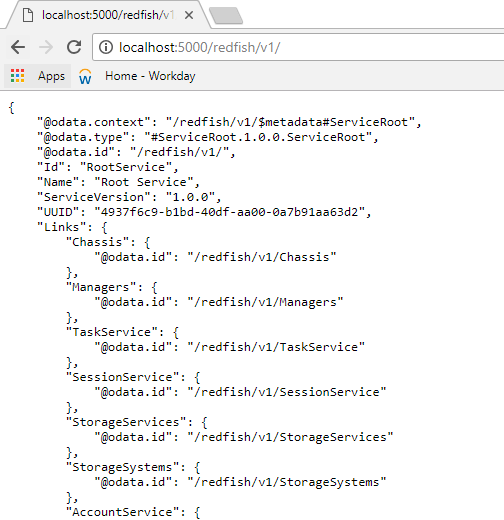


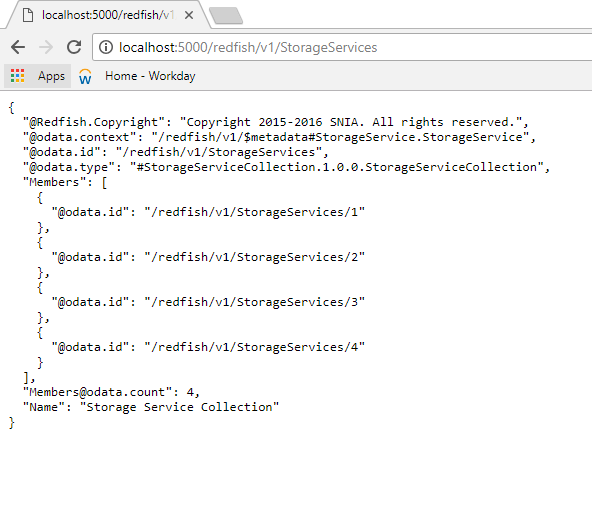


* StorageServicesAPI file
  + add new elements of a collection using “post”. By calling template file we can create a base structure for new element .

**Example**:

Go through Storageservices resource by following steps.



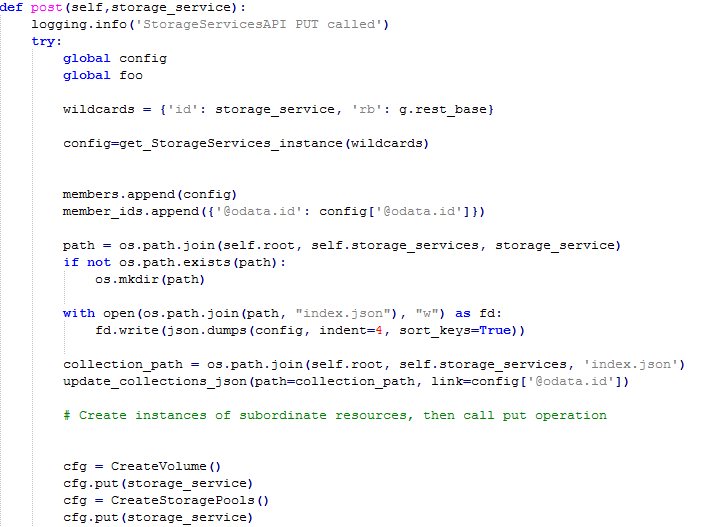


Now we are in **/redfish/v1/storageservices** , here the above figure shows that there are 4 members created in storageservices. If we ant to create more members we have to use “POST” function.

The subresources are created at the time of member creation by writing logic in “post”Function AS shown in the figure below.



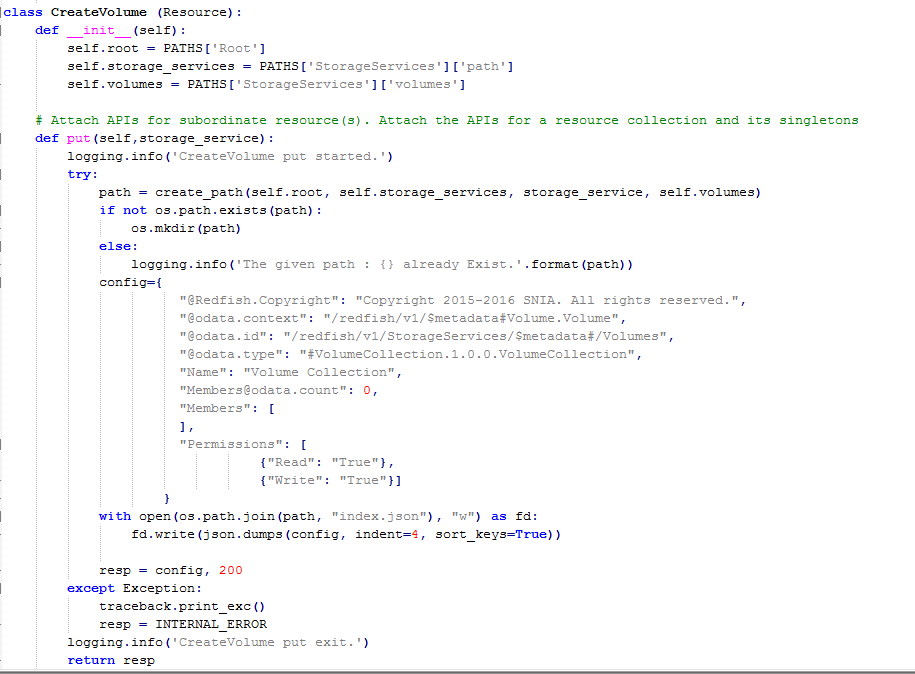
The logic of the above output is show in below figure.



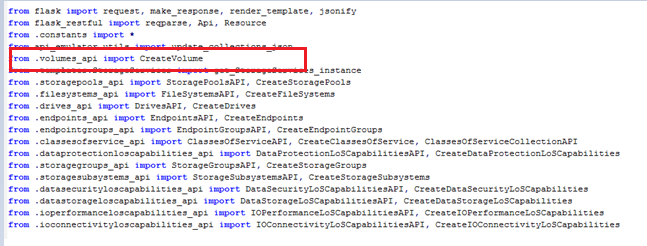
We are creating by passing “post” method. We are creating instance of subresources using post operation.

Example : we are creating volume as a sub-resource in storageServices.

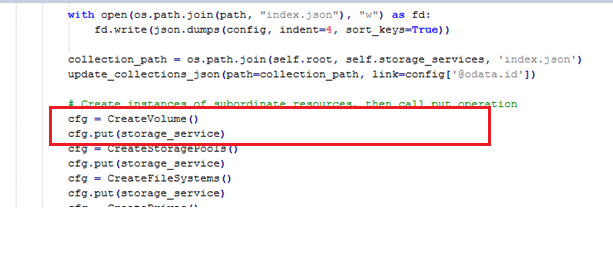
**Step1**: create a class” createvolume ()” in volume\_api.py file



**Step2**:- Importing volumes\_api file to storageservices\_api file



**Step3**:- create instance of volume and then call put operation.



So based on the above code StorageServices , a developer should create resources and sub-resources using template structure.

**Redfish**:

Swordfish Emulator have some features while comparing with redfish.

As swordfish also contains dynamic files and we are storing it in a directory.

Swordfish emulator used to run on top of redfish functionality.

In Redfish Emulator dynamic files are stored in memory. redfish emulator loads all the static and dynamic files .Emulator used to update and delete an item from collection by passing “ident”.

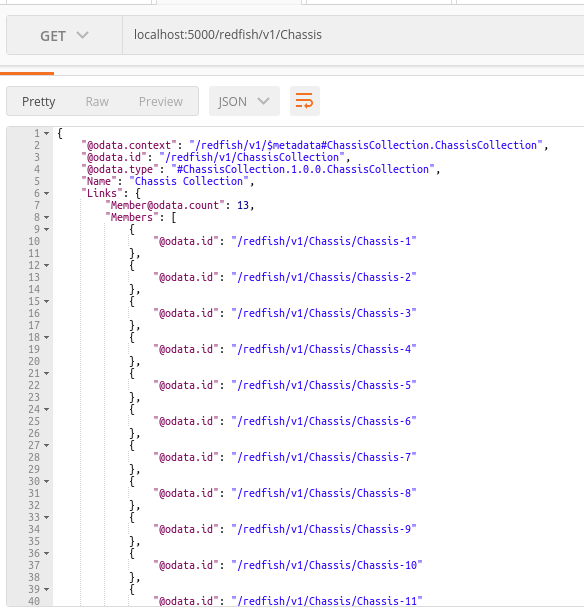
After making all changes if redfish used to restart its emulator ,the data in memory will be lost and it comes with a new version of redfish.

Redfish automatically creates Template files and API files using “codegen”

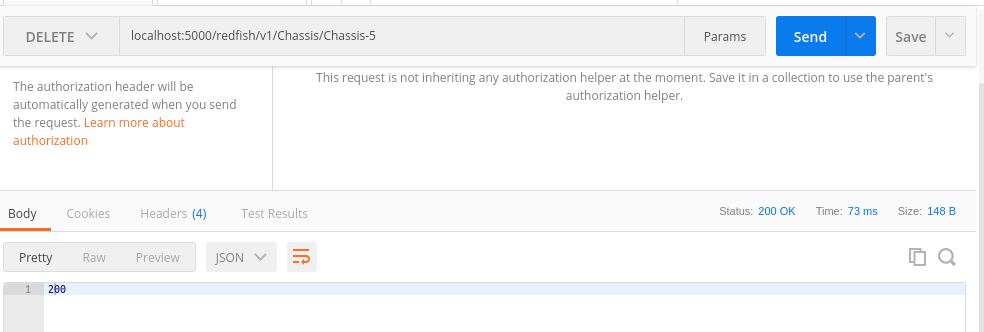
Whenever we start emulator, Redfish used to automate template and api files

Example:

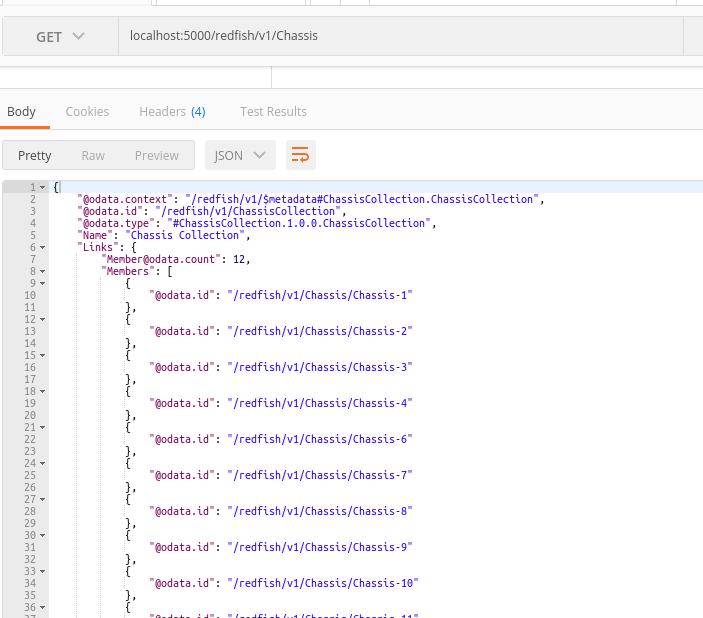
Below figure shows the main collection of chassis, which contains “13” items in a collection. We can go through each member of collection individually. We can update the collection and delete a collection .



The below figure shows deleting “Chassis-5” from collection



From below figure members and member count are updated . Chassis-5 is not in members list.



But if we try to restart the emulator , It automatically creates chassis member “chasiss-5 “

As its storing in memory it won’t save the previous updated data. It stores in cache and it automatically recollects every thing form latest changes.

**Swordfish**:

Swordfish emulator is dynamic .

* We can add a new item to collection
* We can update item in a collection
* We can delete an item in collection

If a user wants to add a new element to a collection, It will store in a “Resource Directory”

* If Swordfish emulator creates dynamic resources , it will take the template structure and it will create the basic structure.
* For update a particular resource it will first get the resource and then it will update the resource.
* For deleting a resource It will get the resource and then it will remove the @odata.id from members and update members count.