Add Hot Spare via WSMAN Feature Design Documentation

Background

This story is aimed to replace the RACAMD dependencies in feature of adding hot spare.

Investigation

Existing features in RackHD for adding hot spare

- Redfish API (*/Systems/{identifier}/Storage/{index}/Drives/{driveIndex}*)

  This API leverages RACADM graph to add hot spare. It requires the following parameters:

  ```
  identifier
  index
  driveIndex
  payload: {
      "username": "string",
      "password": "string",
      "hotspareType": "string",
      "volumeId": "string"
  }
  ```

  As a prerequisite, the target node should be discovered via WSMAN discovery graph. Besides, the *driveIndex* parameter is used to retrieve drive id from *hardware* catalog, which is collected via inventory graph.

- RACADM graph (*Graph.Add.Hotspare*)

  It accepts inputs as below:

  ```
  {
      options: {
          defaults: {
                  username: "",
                  password: "",
                  volumeId: "",
                  driveId: "",
                  hotspareType: "ghs",
                  ipAddress: ""
          }
      }
  }
  ```

SMI features related to RAID operation (SMI Microservice: Server Configuration Profile-WSMAN)

- Import component (*/api/1.0/server/configuration/import*)
  It is an xml file based feature, requires an xml file prepared in share folder. The core changes shall be included in this xml file.
  About how to modify the xml file, please refer to the following pdf guide:
  \\cnrdgps\gpsteamshare\CI_Team_Portal\03_Maglev Team\20170925_ReplaceRACADMwithWSMAN\Configuration XML Workflows.pdf

- Update components (*/api/1.0/server/configuration/updateComponents*)
  It accepts JSON object input, operates with xml file in backed to update components. As it was designed for updating components, it could not handle requirements of adding operation, such as adding volume, adding hot spare.

SMI features to get or export component configurations

- Export (*/api/1.0/server/configuration/export*)
  Export the components information into an xml file, prepared in share folder.

- Get components (*/api/1.0/server/configuration/getComponents*)
  Export the components information into an xml file, prepared in share folder, and also return the components information in JSON format via http response.

Requirements

Design a new graph to achieve adding global hot spare, or hot spare for specified volume via SMI Microservice. Replace the RACADM graph references with the new graph, keep the original Redfish API entry, including API URL, parameters and payload.

Architecture Design and Implementation

As last part of RAID operation features serial, add-hot spare is finally designed to keep the same architecture design as add-volume and delete-volume. About the new graph:

Graph name: *Graph.Dell.Wsman.Add.Hotspare*
Accept options in JSON format as below:

```
options: {
    defaults: {
            username: null,
            password: null,
            volumeId: null,
            hotspareType: 'ghs',
            ipAddress: null
    }
}
```

Four tasks are combined in sequence to implement the task graph.

- *Task.Dell.Wsman.Add.Hotspare.GetXml*
  Implemented by base task *Task.Base.Dell.Wsman.GetXml.* It leverages *getComponents* API in SMI SCP Microservice to export component configurations into xml file, which should be stored in share folder.

- *Task.Dell.Wsman.Add.Hotspare.UpdateXml*
  Modify component configurations based on inputs and exported xml file, then save the changes back into the xml file.

- *Task.Dell.Wsman.RAID*
  Call *import* API in SMI SCP Microservice, import the modified xml file into target node.

- *Task.Dell.Wsman.GetInventory*
  Retrieve inventory data, update corresponding catalog in Mongo DB.

The following shows the architecture design outline.

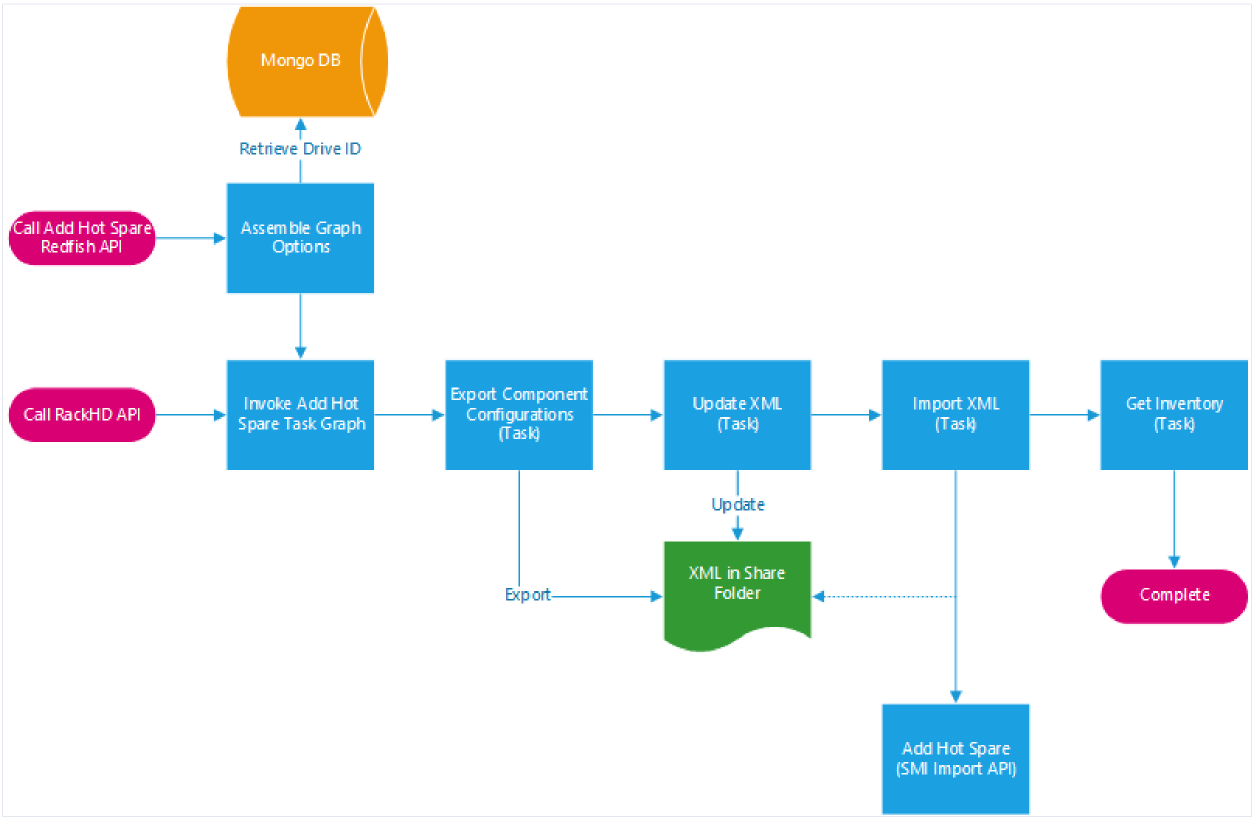

Figure-1 Add Hot Spare Architecture Design

Test Cases

API	Inputs	Result
POST /nodes/{identifier}/workflows	{ "options": { "defaults": { "username": "admin", "password": "admin", "volumeId": "Disk.Virtual.0:RAID.Slot.1-1", "driveId": "Disk.Bay.2:Enclosure.Internal.0-0:RAID.Slot.1-1", "hotspareType": "dhs", "ipAddress": "192.168.188.74", "shutdownType": 0 } } }	Succeed
POST /nodes/{identifier}/workflows	{ "options": { "defaults": { "username": "admin", "password": "admin", "volumeId": "", "driveId": "Disk.Bay.2:Enclosure.Internal.0-0:RAID.Slot.1-1", "hotspareType": "ghs", "ipAddress": "192.168.188.74", "shutdownType": 0 } } }	Succeed
POST /Systems/{identifier}/Storage /{index}/Drives/{driveIndex}	identifier: 59ca21d73a0bb58304df131d index: 0 driveIndex: 2 payload: { "options": { "defaults": { "username": "admin", "password": "admin", "volumeId": "Disk.Virtual.0:RAID.Slot.1-1", "hotspareType": "dhs",	Succeed

	"ipAddress": "192.168.188.74", "shutdownType": 0 } } }	

POST /Systems/{identifier}/Storage /{index}/Drives/{driveIndex}	identifier: 59ca21d73a0bb58304df131d
	index: 0
	driveIndex: 2
	payload: {
	"options": {
	"defaults": {
	"username": "admin",
	"password": "admin",
	"volumeId": "",
	"hotspareType": "ghs",
	"ipAddress": "192.168.188.74",
	"shutdownType": 0
	}
	}
	}
		Succeed

Issues Remaining

Basically, Graph task should be able to be executed independently with input options. So, it would be better to refactor the current architecture design in add-volume, delete-volume and add-hot spare features.