DSS9000 RackManager and RackManager Toolkit Docs

This is a ***Private Dell ESI Github repo*** containing published documentation for the DSS9000 RackManager.

- This doc is the top-level MAN page for the DSS9000 RackManager and RackManager Toolkit.
- the general RackManager ***Concept*** slide deck is at ./***RackManager-Overview.pdf***
- the ***Redfish-Users-Guide*** with APIs supported is at ./***Redfish-Users-Guide.md*** and ***Redfish-Users-Guide.pdf***
- Three sub-folders contain detail man pages referenced herein
  - ./man_pages -- contains all of the user utility command MAN pages
  - ./service_man_pages -- contains MAN pages for RMTK services managed by systemd--being updated and not published yet
  - ./dev_spec -- contains development specs with additional implementation detail--being updated and not published yet
- All documents are natively in Github Markdown format (except concept deck) -- and we are adding pdf bundles
  - docs in man_pages, service_man_pages, and dev_specs will soon be available ***also*** in a single indexed pdf
  - a pdf version of Redfish-Users-Guide is also available here
  - These are easy to browse on Github natively --- it rendors Markdown fast and good, and also renders pdf (but slow)
  - you can download a markdown reader from: `www.markdownpad.com`, * free for the basic reader.
  - you can download the entire repo to zip or with a git clone

---

About RackManager

The DSS9000 ***RackManager*** is an embedded CentOS server in the DSS9000 rack that provides enhanced rack-level management functions.

- The ***RackManager*** hardware platform can be implemented in two ways:
  - For the DSS9000, the *default* platform is an Atom Server that is embedded in the IM (Infrastructure Module) of the DSS9000 rack
    - this may also be referred to as the "Stark RackManager (SRM)" or the "SilverShadow" card
  - If more performance is needed for some features, the RackManager can also be implemented in the form of a 1U server installed in the rack. This is a 1.1 feature
    - this for example could be a PowerEdge R440
- ***RackManager*** interfaces with other controllers in the DSS9000 rack only via the internal rack "Management Network"
  - The DSS9000 with RackManager has an enhanced GbE internal Management Network that interconnects RackManager directly with all of the sled BMCs, as well as the other infrastructure controllers (e.g. the MCs, IMs, BCs)
  - RackManager uses the management network to:

    - communicate with internal rack infrastructure management controllers -- primarily the managed MC
    - communicate directly with node BMCs
      - This allows RackManager software to use any network-based API supported by the BMC (e.g., ipmitool, WSManagement, racadm, redfish, etc.) over the high-speed GbE internal management network

- ***RackManager*** has the following external interfaces

  - Two external RJ45 Ethernet Interfaces to connect to customer networks:
    - for the DSS9000 integrated Stark RM, these ports are labeled Mgmt1 and Mgmt2 on the IM module in the first PowerBay
    - for external 1U RMs, these two ports are most likely LOM2 and LOM3 of the external server
      - LOM1 will be used to connect to the DSS9000's IM module
  - A serial console to the RM is also supported:
    - for the DSS9000 integrated Stark RM, a USB-Serial interface on the IM module can be used to connect to either the Stark RM's serial port.
    - for external 1U RMs, the serial console connects to the BMC via the serial MUX -- see the server hardware guide.

- ***RackManager*** runs the off-the-shelf CentOS 7.1 Minimal Operating System:

  - will pre-installed at the factory on the Stark RackManager along with the RackManager Toolkit
  - can be re-installed or updated using normal CentOS yum update facilities

- ***RackManager*** does not replace the DSS9000's MC, but rather provides a more flexible and higher function management infrastructure for rack-level management

  - The MC is still present in each PowerBay in order to manage the PowerBay.
  - The Managed-MC in PowerBay-1 still consolidates rack-level power and the cooling status, and provides an internal network API that RackManager uses to get the rack-level infrastructure status, or to power-on/off/reset sleds, etc.

About RackManager Boot Firmware

The embedded RackManager in the IM has boot firmware that runs at power-on or reset and boots the CentOS OS.

- This is not a full ACPI-capable BIOS
- Generally no configuration is required
- Details for configuring and updating the boot firmware is contained in the `RMbiosupdate_MAN.md` man page:
  - Link: `man_pages/RMbiosupdate_MAN.md`

About RackManager Toolkit

The ***RackManager Toolkit*** (RMTK) is a set of utilities and services written by Dell ESI specifically to run on RackManager providing enhanced management functions beyond the previous DSS9000 MC, with tight integration to the DSS9000 rack infrastructure.

- It is installed on top of a CentOS 7.1 image as a yum groupinstall named "Dell RackManager Toolkit Local Repository":

  - Link: `man_pages/rackmanager-toolkit-install_MAN.md`

- RMTK includes several Linux Services used by the other utilities and Dell added services:

  - OpenSSH sshd -- so that customers can ssh to the RackManager
    - a customer can ssh to the RackManager through the mgmt ports, and then run other utilities from the RM's bash command-line shell
  - Apache httpd -- The RMTK's RMRedfishService is front-ended with this Apache httpd
    - Future releases will include additional web GUI APIs
  - dhcpd -- used to give IP address to the sled BMCs, utility nodes, and internal switches on the "Internal Management Network"

        - This is only for the internal management network and therefore not visible outside the rack
        - These devices are on isolated VLANs and dhcp will not serve any device connected to the base untagged management network

- RMTK includes several other pre-installed open-source / Dell utilities that a customer can run from the RM command shell: * ipmitool -- the open-source utility for IPMI-based computer hardware management

    - wsmancli -- the open source CLI utility based on OpenWSMAN for communicating with computers that implement the WSMAN Web Service Interface
    - racadm -- the Dell PowerEdge iDrac CLI utility
    - redfishtool -- the open-source DMTF python-based program that runs intelligent redfish commands from a CLI

- RMTK also provides several additional utilities developed for RackManager:

    - RMconfig -- a basic utility to setup and configure the RackManager Toolkit:
        - includes creating default RM Users: rackmanager_adm, rackmanager_oper, rackmanager_readonly
        - includes creating default RM permission groups: RM_ADMIN, RM_OPER, RM_READONLY
        - sets-up default RMTK config files for sshd, dhcpd, httpd, etc., and puts the path to the RMTK utilities in the standard shell path
        - creates the network stack that allows the RM to communicate with the internal management network isolated from the external Ethernet interfaces.
    - RMg5update -- a utility to update DSS9000 infrastructure firmware: MCs, IM, BC, and G5 Switches
    - RMg5cli -- a utility to connect to the DSS9000 MC and execute legacy G5 CLI commands
    - RMredfishtool -- a version of redfishtool CLI that is optimized for the RackManager toolkit on DSS9000
    - RMbiosupdate -- a utility to update the Stark RackManager's BIOS ROM boot firmware
    - RMadmin -- provides several helpful debug and admin subcommands
    - RMversion -- displays the RM Toolkit version

- RMTK includes several key "Services":

    - RMRedfishService -- a rack-level implementation of the industry standard Redfish RESTful hardware management API. The RMRedfishService runs behind the Apache httpd (as either a reverse proxy or using the Apache mod-wsgi). The service provides:
        - a rack-level Redfish service implementation from which one can manage the entire rack
        - caches that select data for speed
        - node-specific data from the sled BMCs directly over the internal management network
        - chassis, fan and power data from the managed MC over the internal management network
    - RMNodeDiscoveryService -- discovers BMC nodes and creates hosts entries with names that map to block.slot
    - RMMgmtPortMonitor -- an internal service that monitors the state of ports on the internal management network as required based on the network topology
        - The key feature is monitoring the status of the Mgmt1 and Mgmt2 external links that connect to RM via VLANS, so if the link ever drops, the RM will know to re-bringup the link
    - RMTimeService -- used to get localtime from the DSS9000 MC if RM has lost its localtime due to a poweroff.
        - When RM starts if it has lost localtime, it re-initializes its localtime from the MC
        - it routinely synces its RM localtim to the MC so that the MC localtime if valid
        - Note that timezine is not synced, but timezone on the MC is not generally visable to a user
    - RMMgtNetworkStart -- is a script used to startup the namespaced network stacks on the RM. It is not a full service.
        - this subsystem is called by systemd as part of normal network start so that the namespaced network stacks on RM used to implement the vlan tunnel from RM to the physical Mgmt1 and Mgmt2 ports on the DSS9000 IM switch is configured correctly.

RackManager Quickstart MAN page

The RackManager will ship pre-installed with CentOS 7.1 and the latest version of RackManager Toolkit (RMTK), with the standard RMTK features enabled.

- The Quickstart MAN page gives step-by-step instructions:
  - for connecting the RackManager to the customers Ethernet network
  - for connecting to RackManager with ssh or Redfish
  - for changing default passwords
  - for adding users
  - and short "how-to" examples for the key interfaces (e.g., RMconfig, Redfish, RMredfishtool).
- See `RM_Quickstart_MAN.md` man page:
  - Link: `<tbd>`

RackManager Toolkit MAN Pages

For additional details on the RackManager Toolkit (RMTK) utilities and services see the following detailed MAN Pages:

- Pre-installed Open-Source/Dell Standard Utilities:
  - ipmitool `-- the ipmitool man page can be found by typing 'man ipmitool'`
  - wsmancli `-- see wsmancli man page at <link>`
  - racadm `-- the racadm man page can be found by typing 'racadm help'`
  - redfishtool `-- see redfishtool man page at https://github.com/DMTF/Redfishtool/blob/master/README.md`
- RackManager Toolkit specific Utilities:
  - RMversion `-- see /man_pages/RMversion_MAN.md`
  - RMconfig `-- see /man_pages/RMconfig_MAN.md`
  - RMg5cli `-- see /man_pages/RMg5cli_MAN.md`
  - RMredfishtool `-- see /man_pages/RMredfishtool.md`
  - RMg5update `-- see /man_pages/RMg5update_MAN.md`
  - RMbiosupdate `-- see /man_pages/RMbiosupdate_MAN.md`
  - RMadmin `-- see /man_pages/RMadmin_MAN.md`
- RackManager Toolkit **Services** MAN Pages:
  - RMRedfishService `-- see /service_man_pages/RMRedfishService_MAN.md`
  - RMNodeDiscoveryService `-- see /service_man_pages/RMNodeDiscoveryService_MAN.md`
  - RMMgmtPortMonitor `-- see /service_man_pages/RMMgmtPortMonitor_MAN.md`
  - RMMgtNetworkStart `-- see service_man_pages/RMMgtNetworkStart_MAN.md`
  - RMTimeService `--see /service_man_pages/RMTimeService_MAN.md`

RackManager Users Guides

The following additional User Guides and API definitions for RackManager interfaces are available:

- ***Redfish-Users-Guide*** `-- see ./Redfish-Users-Guide.md`
  - this describes the Redfish Service implementation on RackManager and lists supported APIs
- G5 CLI Users Guide `-- see the G5 CLI Users Guide for commands you can execute using RMg5cli`

RackManager Development Docs

Additional detailed development documentation for each included RackManager Toolkit utility is located in the DEV_SPECS folder of the rackmanager-docs repo.

Known limitations in R-1.0

- the Quickstart guide is not available in R-1.0
- L2 System APIs such as processor, memory, simple storage inventory are not implemented in R-1.0

RM_LinuxInstall -- RackManager OS Installation Guide

About

This document explains the step-by-step process of installing **CentOS 7.1** onto the RackManager hardware, also known as Silver Shadow.

Because the Silver Shadow does not contain any interface to connect a monitor, a serial console connection must be established beforehand in order to configure the installation.

With a serial console connection in place, the Silver Shadow supports OS installation through USB.

Required Operating System -- CentOS 7.1

- The RackManager Toolkit requires the Silver Shadow hardware to be running **CentOS 7.1**.
- The **minimal** software installation package of CentOS 7.1 should be installed.
- In order to install the OS, you will need to download the .iso image from a mirror.
  - One such mirror is: http://archive.kernel.org/centos-vault/7.1.1503/isos/x86_64/
  - You can download either the DVD .iso or the Minimal .iso:
    - DVD: http://archive.kernel.org/centos-vault/7.1.1503/isos/x86_64/CentOS-7-x86_64-DVD-1503-01.iso
    - Minimal: http://archive.kernel.org/centos-vault/7.1.1503/isos/x86_64/CentOS-7-x86_64-Minimal-1503-01.iso

Serial Console Connection

In order to configure installation options, you must first establish a serial console connection between Silver Shadow's **SoC COM1** and a separate PC. If this is the first time connecting the PC in use to the Silver Shadow, you may need to first download and install the appropriate drivers.

Drivers

- Download and install USB-to-UART drivers for Silicon Labs CP210x to the host PC using the following process:

1. Go to: http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx
2. Select and download the appropriate set of drivers based on your OS version.
3. Extract the contents of the zip file.
4. Run the included installer.

Physical Connection

- Using a standard USB Mini-B cable, connect the Silver Shadow nanoBMC Console port to the host computer.

NOTE:

- The Silver Shadow must be powered (+5V LED lit) in order to connect and establish a serial console connection.
- To connect the SoC COM1 to the mini USB connector, the Console Select Jumper (J13) on the Silver Shadow must be in position 2-3.

- Verify the host computer can see two additional serial ports in the Device Manager.

  NOTE:

  - The two additional serial ports should look similar to:
    - Silicon Labs Dual CP210x USB to UART Bridge: Enhanced COM Port (COMxx)
    - Silicon Labs Dual CP210x USB to UART Bridge: Standard COM Port (COMxx)
  - The port marked **Standard** above will be the SoC COM1 port. This is the port you will want to connect to.

Terminal Emulator

- Connect to the SoC COM1 port of the Silver Shadow using a terminal emulator (e.g. TeraTerm, Hyperterminal, PuTTY) and the following settings:

Parameter	Value
Speed	115,200
Data Bits	8
Parity	None
Stop Bits	1
Flow Control	None

NOTE: The preferred emulation mode is **ANSI**

CentOS 7.1 Installation

If physical access to such a network is unavailable, CentOS 7.1 can also be installed via USB with a bootable USB drive.

Installing via USB

The following detailed steps walk you through the installation process via USB:

1. Create a bootable CentOS 7.1 USB drive using your favorite method, and insert the drive into one of the two USB ports on the Silver Shadow hardware.

2. Confirm the Silver Shadow has power, and then connect to the serial console of Silver Shadow via your favorite serial terminal emulator.

   NOTE: Follow the steps in the `Establishing a Serial Console Connection` section to establish a serial console connection.

3. Reboot/Reset the Silver Shadow.

   NOTE: If your setup is correct and functioning, you should see activity on your serial console

4. Before the existing OS begins to boot, press `F12` to go to the boot menu.
5. Press the number corresponding to the inserted USB drive. You should then be prompted with install options.
6. Move the cursor next to the `Install CentOS 7` option, and press `tab` to configure the install to use the serial console. You should then be presented with the following line of text:

   vmlinuz initrd=initrd.img inst.stage2=hd:LABEL=CENTOS\x207\x20X8 rd.live.check quiet

7. Edit the line by adding **`console=ttyS1,115200`** to the end of the line.
8. After editing the line, press `enter` to start the installation process. After some activity, you should eventually be provided with a menu of selections for installation settings. This is the base installation settings menu.
9. Language settings are English (United States) by default. You can skip selection 1 if you are okay with this selection, or you can press `1` and hit `enter` to set a different language setting.
10. Press `2` and hit `enter` to configure Timezone settings. You should then be prompted to select your timezone region.
11. Select the options for your time zone region. For the US, press `11` and hit `enter`. You should then be prompted to select your specific timezone.
12. Select your specific timezone. For the US Central Standard Time, press `3` and hit `enter`. After this step, the base installation settings menu should then be displayed again.
13. As the minimal software installation is set by default, and meets our needs, you can skip selection 3.
14. The installation source has already been specified on local media, thus selection 4 can also be skipped.
15. Press `5` and hit `enter` to configure network settings.
16. Press `1` and hit `enter` to set the host name.
17. Type your desired host name (e.g. `RackManager`) and press `enter`.
18. Press `2` and hit `enter` to configure device enp1s0.
19. The default values for selections 1-6 are acceptable, thus those selections can be skipped. To change the default if desired, press the corresponding selection number and hit `enter`.
20. If selection 7 is not enabled by default, press `7` and press `enter` to connect automatically after reboot.
21. If Silver Shadow is connected to a network via a network cable, you can press `8` and hit `enter` to apply this configuration in the installer, otherwise you can skip this selection.

    NOTE: If Silver Shadow is not connected to a network and you try to apply configuration in installer, you will see an error message that says **Can't apply configuration, device activation failed.**

22. If you are satisfied with the network settings, press `c` and hit `enter` to continue back to the first network settings menu.
23. Press `c` and hit `enter` again to return to the base installation settings menu.
24. Press `6` and hit `enter` to configure the install destination settings.
25. Select the disk that corresponds to sda by pressing the associated number and hitting `enter`.
26. Press `c` and hit `enter` to proceed to the partitioning options.
27. `Use all space` is selected by default, which meets our needs. Press `c` and hit `enter` to continue to the partitioning scheme options.
28. `LVM` is selected by default, which also meets our needs. Press `c` and hit `enter` to continue back to the base installation settings menu.
29. `Kdump` is enabled by default which meets our needs, so selection 7 can be skipped.
30. Users can be created after the installation, so selection 8 can be skipped.
31. Press `9` and hit `enter` to provide a password for root.
32. Type the password (e.g. `password`) and press `enter`. You will then have to retype it to confirm.

    NOTE: If the password is weak, you will see a warning, and then it will ask you if you want to proceed with the password anyway. Type `yes` and hit `enter` to proceed, or type `no` and hit `enter` to provide a stronger password.

33. All the installation settings have now been set, and installation is ready to begin. Press `b` and hit `enter` to begin the installation.
34. Wait for the installation to complete. This will take approximately 10 minutes, but could take longer.
35. Once the installation is complete, you can hit `enter` to quit, and the system will reboot.
36. You should then see the prompt to login.

    NOTE: You may need to press `enter` for the login prompt to display.

rackmanager-toolkit -- a compressed tarball of the rackmanager toolkit repository

About

rackmanager-toolkit is a compressed tarball of the rackmanager toolkit repository used to install various Dell Systems Management Tools

rackmanager-toolkit depends on the tar utility

rackmanager-toolkit depends on the 'yum' tool to install packages

Usage

```
tar -xvzf rackmanager-toolkit-repo-<version>.tar.gz -C /
yum groupinstall "Dell RackManager Toolkit Local Repository"
```

Note: If you do not have an active internet connection, you will see "Could not resolve host: XXXXX" errors. To avoid this, use the "disablerepo" directive to yum like so:

```
yum groupinstall --disablerepo=\* --enablerepo="dellemc-rckmgrtoolkit-local" "Dell RackManager Toolkit Local Repository"
```

###To install the RackManager Toolkit:

- Login as a user with Administration privileges on a RackManager and uncompress the repository

```
tar -xvzf rackmanager-toolkit-repo-<version>.tar.gz -C /
```

- Set the current date time

```
date --set="Fri May 26 13:58:00 CDT 2017"
```

- Install the local rackmanager toolkit repository

```
yum groupinstall "Dell RackManager Toolkit Local Repository"
NOTE: use the yum option -y to default YES to questions.
```

Note: If you do not have an active internet connection, you will see "Could not resolve host: XXXXX" errors. To avoid this, use the "disablerepo" directive to yum like so:

```
yum groupinstall --disablerepo=\* --enablerepo="dellemc-rckmgrtoolkit-local" "Dell RackManager Toolkit Local Repository"
NOTE: use the yum option -y to default YES to questions.
```

###To UPDATE the RackManager Toolkit:

- Login as a user with Administration privileges on a RackManager and uncompress the repository

```
tar -xvzf rackmanager-toolkit-repo-<version>.tar.gz -C /
```

- Set the current date time (very important, RPMs must be earlier time than the system)

```
date --set="Fri May 26 13:58:00 CDT 2017"
```

- Install the local rackmanager toolkit repository

```
yum groupupdate "Dell RackManager Toolkit Local Repository"
NOTE: use the yum option -y to default YES to questions.
```

Note: If you do not have an active internet connection, you will see "Could not resolve host: XXXXX" errors. To avoid this, use the "disablerepo" directive to yum like so:

```
yum groupupdate --disablerepo=\* --enablerepo="dellemc-rckmgrtoolkit-local" "Dell RackManager Toolkit Local Repository"
NOTE: use the yum option -y to default YES to questions.
```

Run RMconfig

After either an install or an update, you should run RMconfig. cd /opt/dell/rm-tools/RMconfig ./RMconfig.sh -F config

```
### Reboot
After an update & RMconfig, it is possible that network services
may not be successfully restarted. It is best to reboot the system.

###[OPTIONS]:
* None

---
## Installation, Path, and Dependencies:
* ***rackmanager-toolkit*** is released as a tar file
* Contents:
  * rackmanager yum repository
    * rm-tools RPMs
    * 3rd-party RPMs
* ***rackmanager yum repository*** :
   * /opt/dell/rm-toolkit-repo is where repository files will be located (repository directory)
   * /etc/yum.repos.d/dellemc-rackmanger-toolkit.repo - Configuration file name for RM related yum repositories
     * “[dellemc-rckmgrtoolkit-local]” – the definition header to the local filesystem repository

* ***rm-tools*** is released as an RPM that contains RackManager utilities
   * /opt/dell/rm-tools is the location where rm-tools RPMS data will be stored (rm utilities directory)

---
## Examples:

* First extract the rackmanager toolkit into the root filesystem
```

tar -xvzf rackmanager-toolkit-repo-0.4.2.tar.gz -C /

```
* Set the current date time
```

date --set="Fri May 26 13:58:00 CDT 2017"

```
* Next list packages in the local rackmanager toolkit repository
```

yum --enablerepo="dellemc-rckmgrtoolkit-local" list available

```
* Finally install the rackmanager toolkit repository located on the local filesystem
```

yum groupinstall "Dell RackManager Toolkit Local Repository"

```
Note: If you do not have an active internet connection, you will see "Could not resolve host: XXXXX" errors.  To avoid this, use the "disablerepo" directi
```

yum groupinstall --disablerepo=* --enablerepo="dellemc-rckmgrtoolkit-local" "Dell RackManager Toolkit Local Repository"

```
---
## See Also:
* Rackmanager_Quickstart.md (not in release R-0.5)

## Limitations:
* Requires Administrator privileges
* Remote repository implementation is left for the Adminstrator and would require the following:
  * define a stanza in the /etc/yum.repos.d/dellemc-rackmanager-toolkit.repo that contains a definition to point to an internal/external network server (e

## Known Issues:
```

RMbiosupdate -- utility to flash RackManager's BIOS ROM

About

RMbiosupdate is a python 3.4 based utility that makes it easy to flash the RackManager's BIOS ROM with a new firmware image.

- depends on Python 3.4, Flashrom, and flash_util
- parses command arguments, and flashes the specified image to the BIOS ROM
- currently only supports flashing of Dell ESI's Silver Shadow
- logs events to `/var/log/rackmanager/RMbiosupdate.log`

Usage

```
RMbiosupdate -V              -- display version and return
RMbiosupdate -h              -- display usage and return
RMbiosupdate -i <image>      -- flash the specified image to the RM's BIOS ROM
RMbiosupdate -r -i <image>   -- flash the specified image to the RM's BIOS ROM and auto-reboot
```

To run:

- Login as a root on RackManager
- enter `RMbiosupdate [options]`
- Output:
  - -h option will print the usage
  - -V option will print the RMbiosupdate version
  - -i option will attempt to flash the specified image, displaying progress and completion status
  - -r option will auto-reboot the RackManager upon a successful flash

[OPTIONS]:

```
-h, --help                    --- display usage and exit
-V, --Version                 --- display version and exit
-i <image>, --image=<image>   --- flash the specified image to the RM's BIOS ROM
-r, --reboot                  --- auto reboot after successful flash
```

Installation, Path, and Dependencies:

- ***RMbiosupdate*** is included in the rm-tools RPM, and is installed by default when the Dell ESI RackManager Toolkit is installed on the RackManager
- The utility is part of the rackmanager-tools dev repo
- It is installed on the RM at `/opt/dell/rm-tools/RMbiosupdate/*`
- RMbiosupdate logs can be found at `/var/log/rackmanager/RMbiosupdate.log`
- The first line of the program includes a shebang line `#!/usr/bin/env python3` that will direct execution using the python3.4 executable

- The following dependent libraries are imported:
    - python3.4 built-in sys, getopt, subprocess, time, logging, pwd, and os modules

---

Examples:

```
RMbiosupdate -V
        > RMbiosupdate: Version: 0.3

RMbiosupdate -h
        > usage: RMbiosupdate [-V][-h][-i <image>]
        >   OPTIONS:
        >     -h, --help                   # prints usage and exit
        >     -i <image>, --image=<image>  # flashes the specified image
        >     -V, --Version                # prints program version and exits
        >
        > Warning: If using RMbiosupdate utility in a PXE environment, please be careful
        >   that you do not have a shared folder between PXE nodes. This may
        >   cause the update to scramble DMI information.

RMbiosupdate -r -i SILVERSHADOW-01.00.00.04-nodebug.rom
        > RMbiosupdate: Using BIOS ROM Image = SILVERSHADOW-01.00.00.04-nodebug.rom
        > RMbiosupdate: Please do not remove power from the system during flash.
        > RMbiosupdate: RackManager will auto reboot upon successful flash.
        > RMbiosupdate: Flashing...
        > RMbiosupdate: Flash completed successfully.
        > RMbiosupdate: Rebooting RackManager...
```

Limitations:

- requires root user
- requires Flashrom
- requires flash_util
- currently only supports flashing of Dell ESI's Silver Shadow

Known Issues:

- None

RMconfig -- RackManager Configuration Utility

About

RMconfig is a BASH utility used to setup (or configure) the RackManager (RM) Toolkit (RMTK) services and utilities (referred to as SubSystems) after the toolkit is installed.

This includes:

- creating the RMTK network stack (Ethernet configs specific to G5 and RM with VLANs, /etc/hosts, and namespaced network stacks) to isolate the internal management network from the external network
- setting up default RM users, groups, and credentials used for communications w/ G5 MCs and G5 Mgt Network Switches
- creating the RM default configs for standard linux services: dhcpd, httpd, and sshd
- creating the default config for the RM Tools utilities: RM.conf, mc.conf, redfish.conf, etc.
- configuring linux startup services to start the proper services
- configuring the RM BMC, and boot loader Boot.cfg
- configuring the G5 MCs and internal Switches

RMconfig initially configures the RMTK "RMbase" subsystem which will make sure that the paths to various utilities are setup and that a default RM.conf file is created at /etc/opt/dell/rm-tools/

Once the "RMbase" Subsystem has been configured, ***RMconfig*** by default will read the RM.conf config file to determine which other SubSystems to configure (eg dhcpd, httpd, RMRedfishService, RMUsersGroupsPaths, etc.).

However, users can run the command with specific options/arguments to configure a single SubSystem, or to start, stop, or restart a service for debug (eg `RMconfig -s <SubSystem> config` or `RMconfig -s <SubSystem> restart`).

Usage:

```
RMconfig  -V                        -- display version and exit
RMconfig  -h                        -- display overall RMconfig usage and exit
RMconfig  -h -s <subSys>            -- display usage help for a specific SubSystem and exit
RMconfig  [-v][-F] config           -- config all subSystems in RM.conf to default profiles in RM.conf

RMconfig  [-v][-F][-p <cfgProfile>] -s <subSys> <action>  -- run <action> on specified <subSys>
                  If <action>="config", -p <cfgProfile> can be used to specify the profile.
                  Otherwise, by default the profile specified in RM.conf is used with config.

OPTIONS:
        -h          -- display usage help
        -F          -- force reconfig - without -F, SubSystems previously configured will not configure again
        -s <subSys> -- specify the SubSystem - if -s option is used, the <action> must also be specified -
                       if -s <subSys> is not specified, RMconfig runs \"config\" action on all SubSystems
                       in RM.conf with a valid profile name set other than \"None\".
        -v          -- verbose output - can repeat for additional level of verbosity
              -v    -- verbose level 1 is for high-level debug info from the main script - including progress
              -vv   -- verbose level 2 is for subsystem specific flow progress
              -vvv  -- verbose level 3 is for dumping detailed variable info at any level

<subSys>: one of: { RMbase, sshd, dhcpd, httpd, RMBoot, RMbmc, RMMgtNetwork, RMUsersGroupsPaths,
                       RMCredentials, RMg5mc, RMRedfishService, RMNodeDiscoveryService, RMPortMonitorService,... }
                 ex: RMUsersGroupsPaths - creates default RM users, groups, and sets up environment paths

<action>: one of: { config, restart, stop, start }
                 note: enter 'RMconfig -h -s <subSys>' to get specific usage and actions supported for that SubSystem
```

Installation, Path, and Dependencies:

- ***RMconfig*** is included in the rm-tools RPM, and is installed by default when the "Dell EMC RackManager Toolkit" group install package is installed on the RackManager
- The utility and data is installed at `/opt/dell/rm-tools/RMconfig/*`
- The RPM install creates the RMconfig executable script at `/opt/dell/rm-tools/bin/RMconfig`
- and the RPM also places `/opt/dell/rm-tools/bin/` in the default bash/sh shell path for RM users
- The first line of the program includes a shebang line `#!/usr/bin/bash` that will direct execution using standard linux bash

---

Examples:

```
RMconfig -V              --- prints the version and exits
RMconfig -h              --- prints help info and exits
RMconfig config          --- runs a full configuration based on the config profiles in RM.conf
```

Subsystems managed by RMconfig

- Under RMconfig/, there is a SubSystem-specific directory for each SubSystem with a profile-specific bash config script (`subSystem_config.sh`) and config files for the subsystem/profile
    - RMconfig will copy the config files to the proper directory in `/etc/...` where the SubSystem expects them
    - The RMTK includes the main default configs for httpd, dpchd, RMRedfishService, etc.
- the list of SubSystems include:
    - RMbase -- setup default RM.conf config file, and setup RM utility executables
    - RMUsersGroupsPaths -- create default rackmanager users, groups, and paths as defined by the specified profile
    - sshd -- setup sshd config as defined by the specified profile
    - dhcpd -- setup dhcpd config as defined by the specified profile
    - RMMgtNetwork -- create static hosts entries, ethernet configs, and namespaced network stacks
    - httpd -- setup httpd config as defined by the specified profile
    - RMRedfishService -- setup RMRedfishService config as defined by the specified profile
    - RMCredentials -- create/configure ssh keys and redfish auth credentials for communication with the Managed MC
    - RMg5mc -- configure G5 Managed MCs with ssh keys and the MC.conf file
    - RMNodeDiscoveryService -- setup config as defined by the specified profile for idrac/nodes (not in R-05 release)
    - RMSwitchDiscoveryService -- setup config as defined by the specified profile for G5 switches (not in R-05 release)
    - RMPortMonitorService -- setup config as defined by the specified profile (not in R-05 release)
    - RMBoot -- setup RM /boot config as defined by the specified profile (not in R-05 release)
    - RMbmc -- configure the RM BMC, and start RM host interface with a specified RMbmcConfig profile (not in R-05 release)
- High level description of what lower-level config does for each service (by service)
    - RMbase
        - copy the RM.conf, and RMconfig_times files to proper /etc/opt/dell/rm-tools/.. directory, and
        - copy wrapper scripts for other rackmanager-tools utilities to the bin dir so that they will be in the user's path
    - sshd, dhcpd, httpd:
        - copy the correct RM config files depending on the specified profile to their proper locations for the service
        - execute the standard linux command for each of these services to configure it to auto start on boot
        - additional actions that can be specified by targeting the service: start, stop, restart
    - RMBoot:
        - copy the grub.cfg file into the default /boot/grub.cfg directory based on the specified profile
    - RMbmc:
        - configure the Baseboard Management Controller based on the specified profile
        - copy the bmc.conf default config file to /etc -- contains basic RM BMC features to config -- including 1) host interface: enable/disable, 2) out-of-band: enable/disable, 3) IP settings, and 4) ipmi: enable/disable

- RMMgtNetwork:
    - create a default /etc/hosts file with default management network entries (based on profile in RM.conf)
    - create default `/etc/sysconfig/network-scripts/ifcfg-*` files (with proper vlan eth devices) based on specified profile
    - create VLAN eth devices for silverShadow to tunnel the two ext mgt ports to ext ports on the IM switch
    - configure the RM network stack to totally isolates the internal mgt network from external ports
    - configure firewall rules to allow access only to the proper services on the various RM subnets
- RMUsersGroupsPaths
    - add the RM groups for admin, operator, and readonly roles (RM_ADMIN, RM_OPER, and RM_READONLY)
    - add the default RM users for each role (rackmanager_adm, rackmanager_oper, rackmanager_readonly)
    - setup the default path for all users to pickup /opt/dell/rm-tools/bin
- RMCredentials
    - generate ssh keys and ssh config files for use when communicating with the MCs using ssh passwordless authentication
    - configure the RM credential vault w/ proper credentials that RMg5mc can use to communicate w/ managed MCs
    - configure the RM credential vault w/ proper credentials that RM utilities and services can use to communicate w/ iDracs
    - configure the RM credential vault w/ proper credentials that RM utilities and services can use to communicate w/ RM Management Network Switches
- RMg5mc
    - scp the ssh public key files used by RMg5cli to the the Managed MCs
    - copy the MC.conf and MCredfish.conf file to the Managed MC
    - reset the MC for these changes to take effect
- RMRedfishService
    - copy the Redfish.conf file indicated by the specified profile to /etc/opt/dell/rm-tools/Redfish.conf
    - config Linux boot script to auto start RMRedfishService
- RMNodeDiscoveryService
    - copy the RMNodeDiscovery.conf indicated by the specified profile to /etc/opt/dell/rm-tools/RMNodeDiscovery.conf
    - config Linux boot script to auto start RMModeDiscoveryService
- RMPortMonitorService
    - copy the RMPortMonitorService.conf indicated by the specified profile to /etc/opt/dell/rm-tools/RMPortMonitorService.conf
    - config Linux boot script to auto start RMPortMonitorService

---

See Also:

- add more later

Limitations:

- Support for the different SubSystems will be introduced in phases during development up to R-0.9:
    - R-0.5: RMbase, sshd, dhcpd, httpd, RMCredentials, RMRedfishService, RMUsersGroupsPaths, RMMgtNetwork, RMg5mc:
    - R-0.X: adds: RMNodeDiscoveryService, RMBoot, RMbmc, RMPortMonitorService, others

Known Issues:

- For R-0.5, MC users must be created manually as existing users have incorrect capitalization
- See rackmanager-tools issues with the label `RMconfig`

RMg5cli -- Legacy G5 Commandline Utility

About

RMg5cli is a BASH utility that allows a user to run G5 CLI commands from the RackManager.

- If a specific MC CLI sub-command is not specified, an interactive MC CLI shell is started.
- The privilege of the user depends on the RackManager Role Group that the user is a member of:
    - users with Administrator privilege on the RM will execute commands with Admin privilege on the MC
    - users with Operator privilege on the RM will execute commands with Operator privilege on the MC
    - users with ReadOnly privilege on the RM will execute commands with ReadOnly privilege on the MC
- Commands are executed on the targeted G5 Managed MC (MMC)
    - The default MMC is MMC1 - if no MMC is specified
        - in many cases, there is only one MMC in a rack, so MMC1 is the only MMC
        - MMC1 will always be valid
    - MMCs are numbered from bottom up in the rack: MMC1, MMC2, MMC3,...

Usage

```
RMg5cli -V                                     -- display version and returns
RMg5cli -h                                     -- display help and usage info and exit
RMg5cli  [-m <mmc>] [-v]                       -- connects to MMC <mmc> and starts an interactive MC CLI shell
                                                  <mmc> is MMC1 by default
RMg5cli  [-m <mmc>] [-v] <sub-command and args>   -- run the specified sub-command and args on <mmc> and return
```

###To run:

- Login to the Rackmanager via ssh
- enter `RMg5cli [options]`

Options:

```
-V          --- display the version and exit
-h          --- display help and usage info and exit
-m <mmc>    --- <mmc> is the alias in the RackManager hosts file for the targeted MC:
                valid values:   MMC1, MMC2, MMC3, MMC4, ...
-v          --- verbose flag.  can be repeated multiple times for more verbose output:
          -v      ---gives the RM user group that this user is a member of"
          -vv     ---adds the ssh command sent to the MC,
          -vvv    ---adds progname, version, & verboseLvl"
```

Installation, Path, and Dependencies:

- ***RMg5cli*** is included in the rm-tools RPM, and is installed by default when the "Dell EMC RackManager Toolkit" group install package is installed on the RackManager
    - The utility is part of the `rackmanager-tools` dev repo
    - The utility and data is installed at `/opt/dell/rm-tools/RMg5cli/*`
    - A wrapper script `RMg5cli` is placed in `/opt/dell/rm-tools/bin` by RMconfig RMbase subsystem
    - The first line of the program includes a shebang line `#!/usr/bin/bash` that will direct execution using standard linux bash

---

Examples:

```
    RMg5cli -V                --- prints the version and exit
    RMg5cli -h                --- prints help info and exit
    RMg5cli                   --- start interactive legacy G5 CLI on MMC1
    RMg5cli -m MMC2           --- start interactive legacy G5 CLI on MMC2 (the managed MC in management domain 2)
    RMg5cli SHOW /DEVICEMANAGER/RACK1/Block1/Sled1  --- display properties and targets for Block1/Sled1, and return
    RMg5cli -m MMC2 SHOW /DEVICEMANAGER/Rack1/Block2  -- display properties and targets for  Block2 managed by MMC2
```

See Also:

- RMconfig_MAN.md
- RMCredentials_MAN.md
- G5 CLI Users Guide

Limitations:

- Users must be in one of the RM user groups RM_ADMIN, RM_OPER, or RM_READONLY
  - When executed on the MC, these map to MC users: rackmanager_adm, rackmanager_oper, and rackmanager_readonly
- only "managed" MCs can be targeted. You cannot target an unmanaged MC that monitors a 2nd-ary powerbay
  - use raw ssh user@<MCx_y> to connect to a non-managed MC for development or debug

Known Issues:

- See rackmanager-tools Issues, label `RMg5cli`

RMg5update -- a BASH command utility used to update the G5 infrastructure firmware and manufacturing config files

About

RMg5update is a BASH command script that allows a user to update firmware and manufacturing config files for G5/DSS9000 infrastructure controllers (MCs, BC, IMs, and G5switches) from the RackManager.

RMg5update includes a feature where any G5.5-DSS9000 rack can be updated from a generic G5 firmware package.

- The generic G5 firmware package is named `RM_G5FW_VERSION-<ver>.tgz`
    - Where `<ver>` is of form `M.mm` (Major.2digit minor), Ex: "1.23"
- The `RM_G5FW_VERSION-<ver>.tgz` package file contains:
    - IM firmware
    - BC firmware
    - MC firmware
    - G5switch firmware
    - All BC config files (for all rack configs)
    - All IM config files (for all rack configs)
    - All MC Manufacturing config files and generic MC.conf files (for all rack configs)
    - All G5switch config files
    - A `RackConfig` config script for each of the defined rack configurations for G5.5
- With the `-R <Rackcfg>` option, the RM creates the specific HW-dependent "G5-package" that is sent to the MC.
    - A "G5-All Controllers" package is created based on the rackConfig and named `G5_VERSION-<ver>.tgz`
        - This is copied to the MC using scp command
        - Then a ssh command is sent to the MC pointing at the copied file to do the update
    - Then the G5 Switches are sequentially updated

Users can still install manually created G5 packages by:

- Creating a "G5 All" package that includes the controller images
- Creating a compressed tar (.tgz) file of the package and naming the file of the form: `G5_<packageName>-<ver>.tgz`
- Copying the package file to the RM
- Running RMg5update w/o the `-R <Rackcfg>` and with the -C option.
- The `-C` option tells RMg5update to include any Hardware Config files that were in the manually created package. Otherwise config files will be stripped before sending the package to the MC.

All packages are stored as zip compressed tar files in .tgz format.

Usage

```
RMg5update -V       -- display version and exit
RMg5update -h       -- display usage and exit
RMg5update [-v][-t <target>] -R <rackcfg> <RMpackage>
                    -- update G5 Firmware from generic package based on rack configuration <rackcfg>
                       <RMpackage> is path to a generic RM G5 FW package /path/RM_G5FW_VERSION-<ver>.tgz
RMg5update [-v][-t <target>] [-C] <G5Package>
                    -- update G5 Firmware from a custom G5 package
                       <G5package> is path to a custom G5 FW package /path/G5_VERSION-<ver>.tgz
```

To run:

- login as a user with Admin permissions on RackManager
- copy the package file to a temp file location on the RM using scp
  - ex: from the RM, copy the package from another server to the RM
  - `scp <mylogin>@<myServer>:/path/to/myPkgFiles/<packagename> /var/g5updates/.`
- enter `RMg5update [OPTIONS] /var/g5updates/<packagename>`

[OPTIONS]:

```
  -h               --- display usage and exit
  -V               --- display version and exit
  -I               --- displays info about the update and package specified and verifies that options
                       and dependencies are met, but does not update any actual firmware
  -R <rackcfg>     --- include Config files based on <rackcfg>
  -C               --- include Config files if in package.  Always done if -R option specified.
  -t <target>      --- specifies the update target.  The default is G5ALL.
                       <target>={ G5ALL, G5MCIMBC, G5IMSWITCH, G5BLKSWITCH[1:10]}
                       - G5ALL -- default, updates all MCs, BCs, IM, and G5switches in the mgt domain.
                       - G5MCIMBC -- only update the MCs, BCs, and IMs
                       - G5IMSWITCH -- only update the G5 IMswitch
                       - G5BLKSWITCH[1:10] -- unly update the specified G5 Block Switch 1 to 10
  -D <mgtDom>     --- specifies the management Domain.   <mgtDom>=[1:4], Default is 1.
  -v              --- verbose output - can repeat for additional level of verbosity
                       -v     -- verbose level-1 shows general progress as the update proceeds
                       -vv    -- verbose level-2 shows detailed progress as the update proceeds
                       -vvv   -- verbose level-3 dumps detailed debug info during execution
  -M              --- update the mc.conf and redfish.conf on the MC from MC.conf and G5Redfish.conf on the RM
                      after all other  updates to insure that the managed MC has RM config settings
```

Installation, Path, and Dependencies:

- ***RMg5update*** is included in the rm-tools RPM, and is installed by default as part of the RackManager Toolkit
- It is installed on the RM at `/opt/dell/rm-tools/RMg5update/*`
- The first line of the program includes a sheebang line `#!/usr/bin/bash`

Examples:

```
RMg5update -V                  --- prints the version and exits
        > Version: 1.0

RMg5update -h                  --- prints the usage similar to MAN page usage section

RMg5update -v -M -R G55_HW_10BLK_2PB  $myFwPkgs/RM_G5FW_VERSION-3.32.tgz
                               --- update G5 with level-1 verbose messages for rack config G55_HW_10BLK_2PB
                                   to ver 3.32 firmware and sync MC.conf and MCredfish.conf from RM to MC

RMg5update -I -R G55_HW_10BLK_2PB  $myFwPkgs/RM_G5FW_VERSION-3.32.tgz
                               --- verifies that the specified pkg and rackcfg exists
                                   before starting the update
```

See Also:

- RMconfig

Limitations:

- Initial R-0.6 implementation only supports updating G5 MCs, BCs, and IM. G5 switch update is not supported.
    - the options are allowed to test code flow, but the switches are not updated
- In R-0.6, only initial test rack configs are supported with -R option
- In R-0.6, the -M and -I option is not implemented

Known Issues:

- See rackmanager-tools Issues, labeled: `RMg5update`

RMredfishtool

About

RMredfishtool is a Dell ESI customized version of the open source ***redfishtool*** that implements the client side of the Redfish RESTful API for Data Center Hardware Management.

Customizations include optional support for:

- setting the remote host IP at localhost to point to the local Rackmanager Redfish service
- TBD: providing G5 physical location ID aliases that to more easily point to well-known G5 resources
- TBD: performance optimizations by:
  - allowing RMredfishtool to make assumptions about how the local RM Redfish service constructs URIs within collections
  - caching some static output
- TBD: a custom AuthLocal authentication mode that uses the user's role for authorization but allows the service to skip checking password since the user has already authenticated
- TBD: providing some Dell G5-specific OEM commands eg Chassis Reseat, etc

Redfish is the new RESTful API for hardware management defined by the DMTF Scalable Platform Management Forum (SPMF). It provides a modern, secure, multi-node, extendable interface for doing hardware management. The initial release included hardware inventory, server power-on/off/reset, reading power draw, setting power limits, reading sensors such as fans, read/write of ID LEDs, asset tags, and went beyond IPMI in functionality to include inventory of processors, storage, Ethernet controllers, and total memory. (The current 0.9.1 version of redfishtool supports these initial features) New Redfish extensions have now been added to the spec and include firmware update, BIOS config, memory inventory, direct attached storage control, and the list grows.

redfishtool makes it simple to use the Redfish API from a BASH script or interactively from a client command shell.

While other generic http clients such as Linux curl can send and receive Redfish requests, ***redfishtool*** goes well beyond these generic http clients by automatically handling many of the hypermedia and Redfish-specific protocol aspects of the Redfish API that require a client to often execute multiple queries to a redfish service to walk the hypermedia links from the redfish root down to the detailed URI of a specific resource (eg Processor-2 of Blade-4 in a computer blade system). Specifically, redfishtool provides the following functions over curl:

- implements Redfish Session Authentication as well as HTTP Basic Auth
- walks the Redfish schema following strict interoperability processors...] to find the targeted instance based on Id, UUID, URL or other attributes
- handles GETs for collections that are returned in multiple pieces--requiring client to read in a loop until the full collection is returned
- handles ETag and If-Match headers when PATCHing a resource to write properties
- implements many common set or action operations with simple commandline syntax (eg server reset, setting LEDs, assetTag, powerLimits, etc)
- negotiates the latest redfish protocol version between client and service (demonstrating the proper way to do this)
- can read specific properties of a resource, or expand collections to include all members of the collection expanded
- supports adding and deleting users, and common Redfish account service operations
- For debug, provides multiple levels of verbose output to add descriptive headers, and show what http requests are being executed
- For debug, includes multiple levels of status display showing http status codes and headers returned and sent
- For easy parsing, outputs all responses in JSON format unless verbose or status debug options were specified

Why redfishtool?

1. ***redfishtool*** was originally written during the development of the Redfish specification to help find ambiguities in the spec.
2. ***redfishtool*** is now also being used to test inter-operability between redfish service implementations.
3. In addition, ***redfishtool*** provides an example implementation for how a client can execute common server management functions like inventory; power-on/off/reset; setting power limits, indicator LEDs, and AssetTags, and searching a multi-node redfish service to find a specific node (with specific UUID, redfish Id, etc). redfishtool follows strict rules of interoperability. To support this goal, liberal comments are added throughout code to explain why each step is being executed.

4. As described above, it makes it easy to use the Redfish API from a BASH script, or as an easy-to-use interactive CLI -- but WITHOUIT creating a 'new API'. All (rather most) of the responses from ***redfishtool*** are Redfish-defined responses. The properties and resources are defined in the redfish spec. ***redfishtool*** is just a tool to access the Redfish API-not a new interface itself.
    - The exception is that a 'list' operation was added for all collections to display the key properties for each of the members--rather than just the URIs to the members.

Usage

RMredfishtool [***Options***] [***SubCommands***] [***Operation***] [***OtherArgs***]

- ***RMredfishtool*** is a python3.4+ program. It uses the python3 "requests" lib for sending http requests, and a host of other standard libs in python3.4+
- The ***RMredfishtool*** option/optarg parsing strictly follows the well established linux/GNU getopt syntax where arguments and options can be specified in any order, and both short (eg -r) or long (--rhost=) syntax is supported.
- ***options*** are used to pass usernames, passwords, Host:port, authentication options, verbose/status flags, and also to specify how to search to find specific collection members (-I , -a (all), -M :).
- ***subCommands*** indicate the general area of the API (following ipmitool convention), and align with Redfish navigation property names like "Chassis", "Systems", "AccountService", etc.
- ***Operations*** are specify an action or operation you want to perform like `Systems setBootOverride` ..., or `Systems reset`.
- ***OtherArgs*** are any other arguments after the Operation that are sometimes required--like: `Systems <setBootOverride> <enableValue>` "

Common OPTIONS:

```
-V,  --version                   -- show RMredfishtool version, and exit
-h,  --help                      -- show Usage, Options, and list of subCommands, and exit
-u <user>,   --user=<usernm>     -- username used for remote redfish authentication
-p <psswd>, --password=<psswd>   -- password used for remote redfish authentication
-r <rhost>,  --rhost=<rhost>     -- remote redfish service hostname or IP:port
                                 -- by default <rhost> is localhost and thus routed to the RM Redfish service
-t <token>,  --token=<token>     -- redfish auth session token-for sessions across multiple calls
-q,  --quiet                     -- quiet mode--suppress error, warning, and diagnostic messages
-c <cfgFile>,--config=<cfgFile>  -- read options (including credentials) from file <cfgFile>
-T <timeout>,--Timeout=<timeout> -- timeout in seconds for each http request.  Default=10
-P <property>, --Prop=<property> -- return only the specified property. Applies only to all "get" operations
-v,  --verbose                   -- verbose level, can repeat up to 5 times for more verbose output
                                    -v(header), -vv(+addl info), -vvv(Request trace), -vvvv(+subCmd dbg),
                                    -vvvvv(max dbg)
                                    *** use -vvv to see all of the requests being sent for a command
-s,  --status                    -- status level, can repeat up to 5 times for more status output
                                   -s(http_status),
                                   -ss(+r.url, +r.elapsed executionTime ),
                                   -sss(+requestHdrs,data,authType, +respStatus_code, +elapsed exec time,
                                        AuthToken/sessId/sessUri)
                                   -ssss(+response headers for debug), -sssss(+response data for debug)
```

Options used by "raw" subcommand:

```
-d <data>   --data=<data>        -- the http request "data" to send on PATCH,POST,or PUT requests
```

Options to specify top-level collection members: eg: `Systems - I <sysId>`

```
-I <Id>, --Id=<Id>         -- Use <Id> to specify the collection member
-M <prop>:<val> --Match=<prop>:<val> -- Use <prop>=<val> search to find the collection member
-F,  --First               -- Use the 1st link returned in the collection or 1st "matching" link if used with -M
-1,  --One                 -- Use the single link returned in the collection. Return error if more than one member
-a,  --all                 -- Returns all members if the operation is a Get on a top-level collection like Systems
-L <Link>,  --Link=<Link>  -- Use <Link> (eg /redfish/v1/Systems/1) to reference the collection member.
                           --   If <Link> is not one of the links in the collection, and error is returned.
```

Options to specify 2nd-level collection members: eg: `Systems -I<sysId> Processors -i<procId>`

```
-i <id>, --id=<id>         -- use <id> to specify the 2nd-level collection member
-m <prop>:<val> --match=<prop>:val> -- use <prop>=<val> search of 2nd-level collection to specify member
-l <link>  --link=<link>   -- Use <link> (eg /redfish/v1/Systems/1/Processors/1) to reference a 2nd level resource
```

```
                              -- a -I|M|F|1|L option is still required to specify the link to the top-lvl collection
   -a,  --all                 -- Returns all members of the 2nd level collection if the operation is a Get on the
                              --  2nd level collection (eg Processors).
                              --  -I|M|F|1|L still specifies the top-lvl collection.
```

Additional OPTIONS:

```
   -W <num>:<connTimeout>,           -- Send up to <num> {GET /redfish} requests with <connTimeout> TCP connection
       --Wait=<num>:<ConnTimeout>    --   timeouts before sending subcommand to rhost.  Default is -W 1:3
   -A <Authn>,   --Auth <Authn>      -- Authentication type to use:  Authn={None|Basic|Session}  Default is Basic
   -S <Secure>,  --Secure=<Secure>   -- When to use https: (Note: doesn't stop rhost from redirect http to https)
                             <Secure>={Always | IfSendingCredentials | IfLoginOrAuthenticatedApi | Never(default) }
   -R <ver>,  --RedfishVersion=<ver> -- The Major Redfish Protocol version to use: ver={v1(default), v<n>, Latest }
   -C         --CheckRedfishVersion  -- tells RMredfishtool to execute GET /redfish to verify that the rhost supports
                                        the specified redfish protocol version before executing the sub-command.
                                        The -C flag is auto-set if the "-R Latest"  or "-W ..." options were selected.
                                        was specified by the user.
   -H <hdrs>, --Headers=<hdrs>       -- Specify the request header list--overrides defaults. Format "{ A:B, C:D...}
   -D <flag>,  --Debug=<flag>        -- Flag for dev debug. <flag> is a 32-bit uint: 0x<hex> or <dec> format
```

Subcommands:

```
 about                        -- display version and other information about this version of RMredfishtool
 versions                     -- get redfishProtocol versions supported by rhost: GET ^/redfish
 root   |  serviceRoot        -- get serviceRoot resouce: GET ^/redfish/v1/
 Systems                      -- operations on Computer Systems in the /Systems collection
 Chassis                      -- operations on Chassis in the /Chassis collection
 Managers                     -- operations on Managers in the /Managers collection
 AccountService               -- operations on AccountService including user administration
 SessionService               -- operations on SessionService including Session login/logout
 odata                        -- get the Odata Service Document: GET ^/redfish/v1/odata
 metadata                     -- get the CSDL metadata Document: GET ^/redfish/v1/$metadata
 raw                          -- execute raw redfish http methods and URIs (-C option will be ignored)
 hello                        -- RMredfishtool hello world subcommand for dev testing
```

For Subcommand usage, including subcommand Operations and OtherArgs, execute:

```
 RMredfishtool <SubCommand> -h  -- prints usage and options for the specific subCommand
```

Subcommand Operations and Addl Args

Systems Operations

```
RMredfishtool -r <rhost> Systems -h
Usage:
 RMredfishtool [OPTNS]  Systems  <operation> [<args>]  -- perform <operation> on the system specified
<operations>:
  [collection]               -- get the main Systems collection. (Dflt operation if no member specified)
  [get]                      -- get the computerSystem object. (Default operation if collection member specified)
  list                       -- list information about the Systems collection members("Id", URI, and AssetTag)
  patch {A: B,C: D,...}      -- patch the json-formatted {prop: value...} data to the object
  reset <resetType>          -- reset a system.  <resetType>= On,  GracefulShutdown, GracefulRestart,
                                ForceRestart, ForceOff, ForceOn, Nmi, PushPowerPutton
  setAssetTag <assetTag>     -- set the system's asset tag
  setIndicatorLed  <state>   -- set the indicator LED.  <state>=redfish defined values: Off, Lit, Blinking
  setBootOverride <enabledVal> <targetVal> -- set Boot Override properties. <enabledVal>=Disabled|Once|Continuous
                           -- <targetVal> =None|Pxe|Floppy|Cd|Usb|Hdd|BiosSetup|Utilities|Diags|UefiTarget|
  Processors [list]          -- get the "Processors" collection, or list "id" and URI of members.
   Processors [IDOPTN]         --  get the  member specified by IDOPTN: -i<id>, -m<prop>:<val>, -l<link>, -a #all

  EthernetInterfaces [list] -- get the "EthernetInterfaces" collection, or list "id" and URI of members.
   EthernetInterfaces [IDOPTN]--  get the member specified by IDOPTN: -i<id>, -m<prop>:<val>, -l<link>, -a #all

  SimpleStorage [list]       -- get the ComputerSystem "SimpleStorage" collection, or list "id" and URI of members.
   SimpleStorage [IDOPTN]      --  get the member specified by IDOPTN: -i<id>, -m<prop>:<val>, -l<link>, -a #all

  Logs [list]                -- get the ComputerSystem "LogServices" collection , or list "id" and URI of members.
   Logs [IDOPTN]               --  get the member specified by IDOPTN: -i<id>, -m<prop>:<val>, -l<link>, -a #all
  clearLog   <id>            -- clears the log defined by <id>
  examples                   -- example commands with syntax
  hello                      -- Systems hello -- debug command
```

Chassis Operations

```
 RMredfishtool -r <rhost> Chassis -h
 Usage:
    RMredfishtool [OPTNS]  Chassis  <operation> [<args>]  -- perform <operation> on the Chassis specified
 <operations>:
    [collection]                  -- get the main Chassis collection. (Dflt operation if no member specified)
    [get]                         -- get the Chassis object. (Defalut operation if collection member specified)
    list                          -- list information about the Chassis collection members("Id", URI, and AssetTag)
    patch {A: B,C: D,...}         -- patch the json-formatted {prop: value...} data to the object
    setAssetTag <assetTag>        -- set the Chassis's asset tag
    setIndicatorLed  <state>      -- set the indicator LED.  <state>=redfish defined values: Off, Lit, Blinking
    Power                         -- get the full Power resource under a specified Chassis instance.
    Thermal                       -- get the full Thermal resource under a specified Chassis instance.

    getPowerReading [-i<indx>] [consumed] -- get powerControl resource w/ power capacity, PowerConsumed, and power
                 power limits.  If "consumed" keyword is added, then only current usage of powerControl[indx]
                 is returned. <indx> is the powerControl array index. default is 0.  normally, 0 is the only entry
    setPowerLimit [-i<indx>] <limit> [<exception> [<correctionTime>]]   -- set powerLimit control properties
                 <limit>=null disables power limiting. <indx> is the powerControl array indx (dflt=0)

    Logs [list]                   -- get the Chassis "LogServices" collection , or list "id" and URI of members.
      Logs [IDOPTN]                -- get the member specified by IDOPTN: -i<id>, -m<prop>:<val>, -l<link>, -a #all
    clearLog   <id>               -- clears the log defined by <id>
    examples                      -- example commands with syntax
    hello                         -- Chassis hello -- debug command
```

Managers Operations

```
RMredfishtool -r <rhost> Managers -h
Usage:
   RMredfishtool [OPTNS]  Managers  <operation> [<args>]  -- perform <operation> on the Managers specified
  <operations>:
 [collection]                 -- get the main Managers collection. (Dflt operation if no member specified)
 [get]                        -- get the specified Manager object. (Dflt operation if collection member specified)
 list                         -- list information about the Managers collection members("Id", URI, and UUID)
 patch {A: B,C: D,...}        -- patch the json-formatted {prop: value...} data to the object
 reset <resetType>            -- reset a Manager.  <resetType>= On,  GracefulShutdown, GracefulRestart,
                                  ForceRestart, ForceOff, ForceOn, Nmi, PushPowerPutton
 setDateTime <dateTimeString> --set the date and time
 setTimeOffset <offsetSTring> --set the time offset w/o changing time setting
 NetworkProtocol              -- get the "NetworkProtocol" resource under the specified manager.
 setIpAddress [-i<indx>]...   -- set the Manager IP address -NOT IMPLEMENTED YET

 EthernetInterfaces [list]    -- get the managers "EthernetInterfaces" collection, or list "id",URI, Name of members
  EthernetInterfaces [IDOPTN]   --  get the member specified by IDOPTN: -i<id>, -m<prop>:<val>, -a #all

 SerialInterfaces [list]      -- get the managers "SerialInterfaces" collection, or list "id",URI, Name of members.
  SerialInterfaces [IDOPTN]     --  get the member specified by IDOPTN: -i<id>, -m<prop>:<val>, -l<link>, -a #all

 Logs [list]                  -- get the Managers "LogServices" collection , or list "id",URI, Name of members.
  Logs [IDOPTN]                 --  get the member specified by IDOPTN: -i<id>, -m<prop>:<val>, -l<link>, -a #all
 clearLog   <id>              -- clears the log defined by <id>
 examples                     -- example commands with syntax
 hello                        -- Systems hello -- debug command
```

AccountService Operations

```
RMredfishtool -r <rhost> AccountService -h
Usage:
   RMredfishtool [OPTNS]  AccountService  <operation> [<args>]  -- perform <operation> on the AccountService
<operations>:
   [get]                     -- get the AccountService object.
   patch {A: B,C: D,...}     -- patch the AccountService w/ json-formatted {prop: value...}
   Accounts [list]           -- get the "Accounts" collection, or list "Id", username, and Url
     Accounts [IDOPTN]       --   get the member specified by IDOPTN: -i<Id>, -m<prop>:<val>, -l<link>, -a #all
   Roles [list]              -- get the "Roles" collection, or list "Id", IsPredefined, and Url
     Roles [IDOPTN]          --   get the member specified by IDOPTN: -i<Id>, -m<prop>:<val>, -l<link>, -a #all
   adduser <usernm> <passwd> [<roleId>] -- add a new user to the Accounts collection
                                -- <roleId>:{Admin | Operator | ReadOnlyUser | <a custom roleId}, dflt=Operator
   deleteuser <usernm>       -- delete an existing user from Accouts collection
   setpassword  <usernm> <passwd>  -- set (change) the password of an existing user account
   useradmin <userName> [enable|disable|unlock|[setRoleId <roleId>]] -- enable|disable|unlock.. a user account
   examples                  -- example commands with syntax
   hello                     -- AccountService hello -- debug command
```

SessionService Operations

```
RMredfishtool -r <rhost> SessionService -h
Usage:
   RMredfishtool [OPTNS]  SessionService  <operation> [<args>]  -- perform <operation> on the SessionService
<operations>:
   [get]                       -- get the sessionService object.
   patch {A: B,C: D,...}       -- patch the sessionService w/ json-formatted {prop: value...}
   setSessionTimeout <timeout> -- patches the SessionTimeout property w/ etag support
   Sessions [list]             -- get the "Sessions" collection, or list "Id", username, and Url
     Sessions [IDOPTN]         --   get the member specified by IDOPTN: -i<Id>, -m<prop>:<val>, -l<link>, -a #all
   login <user> <passwd>       -- sessionLogin.  post to Sessions collection to create a session
   logout <sessionId>          -- logout or delete the session identified by <SessionId>
   examples                    -- example commands with syntax
   hello                       -- Systems hello -- debug command
```

raw Operations

```
RMredfishtool -r <rhost> raw -h
Usage:
   RMredfishtool [OPTNS] raw <method> <path>

   RMredfishtool raw -h# for help
   RMredfishtool raw examples  #for example commands

  <method> is one of:  GET, PATCH, POST, DELETE, HEAD, PUT
  <path> is full URI path to a redfish resource--the full path following <ipaddr:port>, starting with forward slash /

   Common OPTNS:
   -u <user>,   --user=<usernm>     -- username used for remote redfish authentication
   -p <passwd>, --password=<passwd> -- password used for remote redfish authentication
   -t <token>,  --token=<token>     -- redfish auth session token-for sessions across multiple calls

   -r <rhost>,  --rhost=<rhost>     -- remote redfish service hostname or IP:port
   -X <method>  --request=<method>  -- the http method to use. <method>={GET,PATCH,POST,DELETE,HEAD,PUT}. Dflt=GET
   -d <data>--data=<data>           -- the http request "data" to send on PATCH,POST,or PUT requests
   -H <hdrs>, --Headers=<hdrs>      -- Specify the request header list--overrides defaults. Format "{ A:B, C:D...}"
   -S <Secure>,  --Secure=<Secure>  -- When to use https: (Note: doesn't stop rhost from redirect http to https)

  <operations / methods>:
    GET           -- HTTP GET method
    PATCH         -- HTTP GET method
    POST          -- HTTP GET method
    DELETE        -- HTTP GET method
    HEAD          -- HTTP GET method
    PUT           -- HTTP GET method

  examples-- example raw commands with syntax
  hello   -- raw hello -- debug command
```

Example Usage

System subcommand Examples

```
$ RMredfishtool -r 127.0.0.1:5000 Systems examples
 RMredfishtool -r<ip> Systems              # shows the systems collection
 RMredfishtool -r<ip> Systems list         # lists Id, Uri, AssetTag for all systems
 RMredfishtool -r<ip> Systems -I <id>      # gets the system with Id=<d>
 RMredfishtool -r<ip> Systems -M AssetTag:12345 # gets the system with AssetTag=12345
 RMredfishtool -r<ip> Systems -L <sysUrl>  # gets the system at URI=<systemUrl
 RMredfishtool -r<ip> Systems -F           # get the First system returned (for debug)
 RMredfishtool -r<ip> Systems -1           # get the first system and verify that there is only one system
 RMredfishtool -r<ip> Systems -I <id> patch {A: B,C: D,...}   # patch the json-formatted {prop: value...}
                                                              data to the object
 RMredfishtool -r<ip> Systems -I <id> reset <resetType>       # reset a system.  <resetType>=the redfish-defined
                                                             _# values: On, Off, gracefulOff...
 RMredfishtool -r<ip> Systems -I <id> setAssetTag <assetTag>  # set the system's asset tag
 RMredfishtool -r<ip> Systems -I <id> setIndicatorLed  <state>  # set the indicator LED.
```

```
                                                              <state>=redfish defined values: Off, Lit, Blinking
 RMredfishtool -r<ip> Systems -I <id> setBootOverride <enabledVal> <targetVal> # set Boot Override properties.
                                                                 <enabledVal>=Disabled|Once|Continuous
 RMredfishtool -r<ip> Systems -I<Id> Processors# get the processors Collection
 RMredfishtool -r<ip> Systems -I<Id> Processors list   # lists Id, Uri, & Socket for all processors in system w/ Id=<Id>
 RMredfishtool -r<ip> Systems -I<Id> Processors -i 1   # get the processor with id=1 in system with Id=<Id>
 RMredfishtool -r<ip> Systems -L <sysUrl> Processors -m Socket:CPU_1  # get processor with property Socket=CPU_1,
                                                                     on system at url <sysUrl>
```

Chassis subcommand Examples

```
$ RMredfishtool -r 127.0.0.1:5000 Chassis examples
 RMredfishtool -r<ip> Chassis  # shows the Chassis collection
 RMredfishtool -r<ip> Chassis list # lists Id, Uri, AssetTag for all Chassis
 RMredfishtool -r<ip> Chassis -I <id>  # gets the Chassis with Id=<d>
 RMredfishtool -r<ip> Chassis -M AssetTag:12345# gets the Chassis with AssetTag=12345
 RMredfishtool -r<ip> Chassis -L <sysUrl>  # gets the Chassis at URI=<systemUrl
 RMredfishtool -r<ip> Chassis -F   # get the First Chassis returned (for debug)
 RMredfishtool -r<ip> Chassis -1   # get the first Chassis and verify that there is only one system
 RMredfishtool -r<ip> Chassis -I <id> patch {A: B,C: D,...} # patch the json-formatted {prop: value...} data
                                                           to the object
 RMredfishtool -r<ip> Chassis -I <id> setAssetTag <assetTag># set the system's asset tag
 RMredfishtool -r<ip> Chassis -I <id> setIndicatorLed  <state>  # set the indicator LED.
                                                             <state>=redfish defined values: Off, Lit, Blinking
 RMredfishtool -r<ip> Chassis -I<Id> Power # get the full chassis Power resource
 RMredfishtool -r<ip> Chassis -I<Id> Thermal   # get the full chassis Thermal resource
 RMredfishtool -r<ip> Chassis -I<Id> getPowerReading[-i<indx> [consumed]   # get chassis/Power powerControl[<indx>]
                              resource if optional "consumed" arg, then return only the PowerConsumedWatts prop
 RMredfishtool -r<ip> Chassis -L<Url> setPowerLimit [-i<indx>] <limit> [<exception> [<correctionTime>]]
                              _# set power limit
```

Managers subcommand Examples

```
$ RMredfishtool -r 127.0.0.1:5000 Managers examples
 RMredfishtool -r<ip>   # shows the Managers collection
 RMredfishtool -r<ip> Managers list             # lists Id, Uri, AssetTag for all Managers
 RMredfishtool -r<ip> Managers -I <id>          # gets the Manager with Id=<d>
 RMredfishtool -r<ip> Managers -M AssetTag:12345 # gets the Manager with AssetTag=12345
 RMredfishtool -r<ip> Managers -L <mgrUrl>      # gets the Manager at URI=<mgrUrl
 RMredfishtool -r<ip> Managers -F               # get the First Manager returned (for debug)
 RMredfishtool -r<ip> Managers -1               # get the first Manager and verify that there is only one Manager
 RMredfishtool -r<ip> Managers -I <id> patch {A: B,C: D,...} # patch the json-formatted {prop: value...} data
                                                            to the object
 RMredfishtool -r<ip> Managers -I <id> reset <resetType>     # reset a Manager.
                                        <resetType>=the redfish-defined values: On, Off, gracefulOff...
 RMredfishtool -r<ip> Managers -I<Id> NetworkProtocol         # get the NetworkProtocol resource under the
                                                             specified manager
 RMredfishtool -r<ip> Managers -I<Id> EthernetInterfaces list # lists Id, Uri, and Name for all of the NICs
                                                             for Manager w/ Id=<Id>
 RMredfishtool -r<ip> Managers -I<Id> EthernetInterfaces -i 1 # get the NIC with id=1 in manager with Id=<Id>
 RMredfishtool -r<ip> Managers -L <Url> EthernetInterfaces -m MACAddress:AA:BB:CC:DD:EE:FF # get NIC with MAC AA:BB...
                                                                                        for manager at url <Url>
```

AccountService subcommand Examples

```
$ RMredfishtool -r 127.0.0.1:5000 AccountService examples
 RMredfishtool -r<ip> AccountService               # gets the AccountService
 RMredfishtool -r<ip> AccountService patch { "AccountLockoutThreshold": 5 } ]# set failed login lockout threshold
 RMredfishtool -r<ip> AccountService Accounts      # gets Accounts collection
 RMredfishtool -r<ip> AccountService Accounts list  # list Accounts to get Id, username, url for each account
 RMredfishtool -r<ip> AccountService Accounts -mUserName:john # gets the Accounts member with username: john
 RMredfishtool -r<ip> AccountService Roles  list    # list Roles collection to get RoleId, IsPredefined, & url
                                                   for each role
 RMredfishtool -r<ip> AccountService Roles   -iAdmin          # gets the Roles member with RoleId=Admin
 RMredfishtool -r<ip> AccountService adduser john 12345 Admin # add new user (john) w/ passwd "12345" and role: Admin
 RMredfishtool -r<ip> AccountService deleteuser john          # delete user "john"s account
 RMredfishtool -r<ip> AccountService useradmin john disable   # disable user "john"s account
 RMredfishtool -r<ip> AccountService useradmin john unlock    # unlock user "john"s account
```

SessionService subcommand Examples

```
$ RMredfishtool -r 127.0.0.1:5000 SessionService examples
 RMredfishtool -r<ip> SessionService                            # gets the sessionService
 RMredfishtool -r<ip> SessionService setSessionTimeout <timeout> # sets the session timeout property
 RMredfishtool -r<ip> SessionService Sessions                   # gets Sessions collection
 RMredfishtool -r<ip> SessionService Sessions -l<sessUrl>       # gets the session at URI=<sessUrl
 RMredfishtool -r<ip> SessionService Sessions -i<sessId>        # gets the session with session Id <sessId>
 RMredfishtool -r<ip> SessionService patch {A: B,C: D,...}      # patch the json-formatted {prop: value...}
                                                                data to the sessionService object
 RMredfishtool -r<ip> SessionService login <usernm> <passwd>    # login (create session)
 RMredfishtool -r<ip> SessionService logout <sessionId>         # logout (delete session <sessId>
```

Known Issues, and ToDo Enhancements

1. modifications to make PATCH commands work better with Windows cmd shell quoting
2. support clearlog
3. add additional APIs that have been added to Redfish after 1.0---this version supports only 1.0 APIs
4. add custom role create and delete