Accelerate Development of SNIA Swordfish™

Don Deel
NetApp, Inc.
Agenda

- SNIA Swordfish™ and Open Source Projects
- More open source that can help with Swordfish
- Where to find more information
SNIA Swordfish™ and Open Source Projects

- Swordfish comes from the SNIA Scalable Storage Management Technical Work Group (SSM TWG)
- The SSM TWG maintains several open source projects that can accelerate the development of Swordfish implementations
- These open source projects are on GitHub in open repositories under github.com/SNIA
Swordfish Open Source Projects

- Swordfish API Emulator
- Swordfish Basic Web Client
- Swordfish Datadog Sample Dashboard Integration
- Swordfish Power BI Sample Dashboard Integration
- Swordfish PowerShell Toolkit
- Swordfish Extension to Windows Admin Center
- More open source projects are coming
Swordfish API Emulator
Swordfish API Emulator

- Emulates a Swordfish system with storage services
- Responds to create, read, update, and delete operations
  - POST, GET, PATCH, DELETE
- Extends the DMTF Redfish Interface Emulator
  - Adds code for Swordfish resources
- Link: https://github.com/SNIA/Swordfish-API-Emulator
- Includes installation, user, and developer documentation
Swordfish API Emulator Console Output (Default Configuration)
Swordfish API Emulator

Browser Output (Default Configuration)
More About the Swordfish API Emulator

- Emulator Python Environment
- Installing the Emulator
- Notes About the Emulator
- How the Emulator Works
- Adding New Dynamic Resources
- More About the Emulator
Emulator Python Environment

- Python 3.6 or above
- virtualenv recommended but not required
- Python packages
  - flask flask_restful flask_httpauth
  - requests aniso8601 markupsafe pytz
  - itsdangerous StringGenerator urllib3
Installing the Emulator
(Default Configuration)

- Create a folder/directory for the Emulator
- Copy in the Redfish Interface Emulator
- Copy in the Swordfish API Emulator on top of it
- Install the necessary Python packages
- Run with “python emulator.py”
Emulator Console Output
(Default Configuration)

(SAE180828) C:\Users\ddon\Documents\SAE180828>python emulator.py
['Redfish']
* Redfish endpoint at localhost:5000
{\'rb\': '/redfish/v1/', \'sys_id\': 'System-1'}
{\'rb\': '/redfish/v1/', \'sys_id\': 'System-2'}
{\'rb\': '/redfish/v1/', \'sys_id\': 'System-3'}
{\'rb\': '/redfish/v1/', \'sys_id\': 'System-4'}
{\'rb\': '/redfish/v1/', \'sys_id\': 'System-5'}
{\'rb\': '/redfish/v1/', \'sys_id\': 'System-6'}
{\'rb\': '/redfish/v1/', \'sys_id\': 'System-7'}
* Running in Redfish mode
* Serving Flask app "g" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
Notes About the Emulator

- Read the Redfish Interface Emulator README.md
  - Says how to use emulator.py flags and emulator-config.json
- api_emulator\resource_manager.py establishes which resources are static and which are dynamic
  - Static resources are read-only
  - Dynamic resources support CRUD operations
- Swordfish resources are all dynamic, but four of the default configuration Redfish resources are static
  - AccountService, Registries, SessionService, TaskService
Notes About the Emulator (Continued)

- Static resources are populated by JSON mockup files in the `api_emulator\redfish\static` directory
  - Only uses static resources identified in “resource_manager.py”
  - Dynamic resources are NOT populated or initialized this way

- Dynamic resources can be populated via the emulator API using CRUD operations (POST, GET, PATCH, DELETE)

- The Redfish Interface Emulator also includes a tool called “Infragen” that can prepopulate dynamic resources
  - This tool is used to instantiate the Redfish resources in the emulator’s default configuration
Notes About the Emulator (Continued Again)

- An emulator-only function can populate dynamic objects
  - When defined by an API file for a dynamic resource, a POST with an empty body can create a new default singleton instance:
    ```
    POST http://localhost:5000/redfish/v1/Chassis/NewThing {}
    ```
  - The new instance (named “NewThing” here) is defined by a template file for the dynamic resource (“Chassis” in this case)
- The Swordfish Basic Web Client uses this emulator-only function to create new Redfish and Swordfish singletons
  - It can then use PATCH operations to alter properties and customize the new dynamic object
How the Emulator Works

Flask and Flask-RESTful

- emulator.py
- resource_manager.py
- static_loader.py and resource_dictionary.py

Attaches APIs for Resources to URIs

Redfish/Swordfish API

- Chassis_api.py
- ComputerSystem_api.py
- StorageServices_api.py

Chassis.py
ComputerSystem.py
StorageServices.py

api_emulator\redfish
api_emulator\redfish\templates

Static Resources
Dynamic Resources
Adding New Dynamic Resources to the Emulator

- Dynamic resources are enabled by api/template file pairs
  - The api file sets REST behaviors for Collections and Singletons
  - The template file establishes how to create default singletons
- Example api/template files are in api_emulator\redfish
  - “eg_resource_api.py” and “template\eg_resource.py”
  - “eg_subresource_api.py” and “template\eg_subresource.py”
- The example api files show where to handle applicable REST commands for Collections and for Singletons
  - GET, PUT, POST, PATCH, DELETE
Adding New Dynamic Resources to the Emulator (Continued)

- The example template files show how templates are set up to allow new singleton instances to be created
  - A template is copied, with some things filled in at runtime
- When a new api/template pair is created, it is added to the emulator by editing “resource_manager.py”
  - This will attach the new resource’s APIs to URIs
More About the Emulator

- Areas where the emulator should improve
  - The remaining static resources should be made dynamic
  - A couple of Redfish-defined URIs are not supported yet

- Feedback is desired
  - About possibilities as well as issues
Swordfish Basic Web Client
Swordfish Basic Web Client

- Web client that can connect to multiple Redfish and/or Swordfish services simultaneously
- Presents the Redfish and Swordfish hierarchy in a browser web frame
- Provides basic capabilities for viewing resources and updating properties that are writeable
- Link: https://github.com/SNIA/Swordfish-basic-web-client
- Includes installation, user, and developer documentation
Swordfish Basic Web Client Screen Output (Service Login)
Swordfish Basic Web Client Screen Output (Service Root)
Swordfish Basic Web Client Screen Output (StorageServices)
Swordfish Datadog
Sample Dashboard Integration
Swordfish Datadog Sample Dashboard Integration

- Basic dashboard for the Datadog monitoring service
- Connects to a Swordfish service and provides an integration to the Datadog User Interface
- Displays storage system capacity information and the available storage capacity thresholds
- Can be a starting point for a customized Datadog plugin
- Link: https://github.com/SNIA/Swordfish-datadog-sample-dashboard-integration
- Includes installation, user, and developer documentation
Swordfish Datadog
Sample Dashboard Output
Swordfish Power BI
Sample Dashboard Integration
Swordfish Power BI Sample Dashboard Integration

- Basic dashboard for the Power BI monitoring system
- Connects to a Swordfish service and provides an integration to the Power BI User Interface
- Displays storage system capacity information and the available storage capacity thresholds
- Can be a starting point for a customized Power BI plugin
- Link: https://github.com/SNIA/Swordfish-powerBI-sample-dashboard-integration
- Includes installation, user, and developer documentation
Swordfish Power BI
Sample Dashboard (Main)
Swordfish Power BI Sample Dashboard (Child)
More open source projects that can help developers work with Swordfish
Stay tuned for these talks today

- Accelerating Windows Integrations: Introducing the SNIA Swordfish™ PowerShell Tool Kit and Windows Admin Center Integration
  - Using PowerShell with Swordfish
  - Using Windows Admin Center with Swordfish

- Golang Support for SNIA Swordfish™
  - Using Go with Swordfish and Redfish
Where to find more information
Swordfish and Redfish Standards Information

- Swordfish information on the public SNIA web site
  - Specs, Schema, Mockups, User’s Guide, etc
  - Link: www.snia.org/swordfish

- Redfish information on the public DMTF web site
  - Specs, Schema, Mockups, Educational Material, etc.
  - Links: www.dmtf.org/redfish and https://redfish.dmtf.org

- Redfish (and Swordfish) Public Discussion Forum
  - Link: www.redfishforum.com (or www.swordfishforum.com)
  - Ask questions on this forum and get answers from experts
Swordfish and Redfish Open Source Repositories

- Swordfish open source projects on GitHub
  - Link: https://github.com/SNIA

- Redfish open source projects on GitHub
  - Link: https://github.com/dmtf
Contributing to SNIA Open Source Projects

- SNIA open source projects welcome input!
- Contributors who are not SNIA members need to agree to the terms of the SNIA Contributor License Agreement
  - Link: https://www.snia.org/CLA
- The SNIA Swordfish open source projects are covered by the terms of the BSD 3-clause License
Thank You!
Please take a moment to rate this session.

Your feedback matters to us.