Expanding Development of Swordfish Implementations Using Open Source Tools

Presented by
Don Deel
Agenda

- SNIA Swordfish™ and Open Source Projects
- Swordfish Related Open Source Projects
- SNIA Swordfish Conformance Testing and Open Source
- Where to Find More Information
SNIA Swordfish™ and Open Source Projects
SNIA Swordfish™ and Open Source Projects

- Swordfish comes from the SNIA Scalable Storage Management Technical Work Group (SSM TWG)
- Swordfish is an extension of DMTF Redfish®
- The SSM TWG maintains several open source projects that can help accelerate the development of Swordfish implementations
- These open source projects are kept in open repositories under [github.com/SNIA](https://github.com/SNIA)
SNIA Swordfish Open Source Projects

- Swordfish API Emulator
- Swordfish Basic Web Client
- Swordfish PowerShell Toolkit
- Swordfish Datadog Sample Dashboard Integration
- Swordfish Power BI Sample Dashboard Integration
SNIA Swordfish Mockups

- Mockups are important for understanding how different situations can be handled by Swordfish
  - Point-in-time representations of modeled systems
  - Show the types of information that can be modeled
- Example Swordfish mockups are at: swordfishmockups.com
  - Several different Swordfish storage configurations are shown
    - Standalone, Integrated, Service-Based, NVMe and NVMe-oF
  - Each mockup includes a brief description of the storage system modeled
  - Mockups can be explored using a browser that has a JSON viewing plugin
  - Mockups show representations of implementations, and are *not normative*
SNIA Swordfish Mockups (Continued)

- Swordfish mocksups on swordfishmockups.com can be explored with a browser that has a JSON viewing plugin.
SNIA Swordfish Mockups (Continued Again)

- Mockups are stored as hierarchical directory structures
- Each directory corresponds to a Redfish/Swordfish object
- A file named `index.json` within each directory describes the state elements (properties, links, etc) for the object
- The top-most directory in the hierarchical directory structure represents the Redfish root (/redfish/v1)
- The directory structure reflects the Redfish/Swordfish object hierarchy
Swordfish API Emulator
Swordfish API Emulator

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

```bash
$ python ./emulator.py
INFO:root:Mockup folders
['Mockups']
  * Redfish endpoint at localhost:5000
  * Using dynamic emulation
  * Use HTTP
  * Running in Redfish mode
  * Serving Flask app "g" (lazy loading)
  * Environment: production
    WARNING: This is a development server. Do not use it in a production deployment.
    Use a production WSGI server instead.
  * Debug mode: off
INFO:werkzeug: * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```
Swordfish API Emulator Browser Output (Default Configuration)
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
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](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 PowerShell Toolkit
Swordfish PowerShell Toolkit

- Basic framework for querying a Swordfish service
- Supported on Microsoft Windows, Windows Server, Linux, and MacOS
- Works with a Swordfish target, emulator, or swordfishmockups.com
- PowerShell wrapper for REST API calls to Redfish and Swordfish
- Link: https://github.com/SNIA/Swordfish-Powershell-Toolkit
- Includes installation, user, and developer documentation
Swordfish PowerShell Toolkit Example One

```powershell
PS C:\Users\Administrator\Desktop\Swordfish-Powershell-Toolkit> import-module \SNIASwordFish.psm1
PS C:\Users\Administrator\Desktop\Swordfish-Powershell-Toolkit> Connect-SwordFishTarget -Target 'localhost' -Port 5000
Base URI = http://localhost:5000/redfish/v1/
@odata.context : /redfish/v1/$metadata#ServiceRoot
@odata.type : #ServiceRoot.1.0.0.ServiceRoot
@odata.id : /redfish/v1/
Id : RootService
Name : Root Service
ServiceVersion : 1.0.0
UUID : 427b01db-96bd-4f53-9ecc-4c9c48a8e635
Links : @([Chassis=; Managers=; TaskServices=; SessionService=; StorageServices=; StorageSystems=; AccountService=
PS C:\Users\Administrator\Desktop\Swordfish-Powershell-Toolkit> ```
Swordfish PowerShell Toolkit Example Two

```
Administrator: Windows PowerShell

PS C:\Users\Administrator\Desktop\Swordfish-Powershell-Toolkit> import-module .\SNIASwordFish.psm1
PS C:\Users\Administrator\Desktop\Swordfish-Powershell-Toolkit> Connect-SwordFishMockup

@Redfish.Copyright : Copyright 2014-2019 Distributed Management Task Force, Inc. (DMTF). All rights reserved.
@odata.context : /redfish/v1/$metadata#ServiceRoot.ServiceRoot
@odata.id : /redfish/v1/
@odata.type : #ServiceRoot.v1_3_0.ServiceRoot
Id : RootService
Name : Root Service
RedfishVersion : 1.0.0
UUID : 92384634-2938-2342-8820-489239905423
Systems : @(odata.id='redfish/v1/Systems')
StorageSystems : @(odata.id='redfish/v1/Systems')
StorageServices : @(odata.id='redfish/v1/StorageServices')
Chassis : @(odata.id='redfish/v1/Chassis')
Managers : @(odata.id='redfish/v1/Managers')
Tasks : @(odata.id='redfish/v1/Tasks')
SessionService : @(odata.id='redfish/v1/SessionService')
AccountService : @(odata.id='redfish/v1/AccountService')
EventService : @(odata.id='redfish/v1/EventService')
Links : @{}
```
Swordfish PowerShell Toolkit Objects

- Everything is returned as objects (and nested objects)
  - Cast to variable
    ```powershell
    $MyVols = Get-SwordFishVolume
    ```
  - Can access like an array, or filter by properties
    ```powershell
    $MyVols[4]
    $MyVols | where {$_ .name -eq 'Volume 5'}
    ```
  - Can dig deeper into single values
    ```powershell
    $MyVols[4].status
    ```
Swordfish PowerShell Toolkit Objects (Continued)

- Can cast the variable back to JSON format

```powershell
PS C:\> $MyVols[4] | convertto-json
{
    "@Redfish.Copyright": "Copyright 2014-2019 SNIA. All rights reserved.",
    "@odata.context": "/redfish/v1/$metadata#Volume.Volume",
    "@odata.id": "/redfish/v1/StorageServices/1/Volumes/5",
    "@odata.type": "#Volume.v1_4_0.Volume",
    "Name": "Volume 5",
    "Id": "5",
    "Description": "Volume 5.",
    "Identifiers": [
        {
            "DurableNameFormat": "NAA",
            "DurableName": "6545675456761001244076100123487"
        }
    ],
    "Manufacturer": "SuperDuperSSD",
    "Model": "Drive Model string",
    "Status": {
        "State": "Enabled",
        "Health": "OK"
    },
    "AccessCapabilities": [
        "Read",
        "Write",
        "Append",
        "Streaming"
    ]
}
```
Swordfish PowerShell cmdlets (current list)

- Connect-SwordFishTarget
- Connect-SwordFishMockup
- Get-SwordfishSessionToken
- Get-SwordfishStorage
- Get-SwordfishStorageService
- Get-SwordfishSystem
- Get-SwordfishChassis
- Get-SwordfishSessionService
- Get-SwordfishZone
- Get-SwordfishTask
- Get-SwordfishSession
- Get-SwordFishChassisPower
- Get-SwordFishChassisThermal
- Get-SwordfishConnection
- Get-SwordfishController
- Get-SwordfishDrive
- Get-SwordfishEndpoint
- Get-SwordfishEthernetInterface
- Get-SwordfishGroup
- Get-SwordFishPool
- Get-SwordfishVolume
- Get-SwordfishSession
- Get-SwordfishManager
- Get-SwordfishClassOfService
- Get-SwordfishDataStorageLinesOfService
- Get-SwordfishDataStorageLoSCapabilities
- Get-SwordfishIOConnectivityLoSCapabilities
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)
Swordfish Related Open Source Projects
Swordfish Related Open Source Projects

- **DMTF Redfish Open Source Projects**
  - Several useful tools for working with Redfish (and Swordfish)
- **fishem**
  - An emulator that brings Redfish and Swordfish mockups to life
- **Gofish**
  - Golang client library for interacting with DMTF Redfish and SNIA Swordfish
- **Swordfish Ember Client**
  - Frontend client for the Swordfish stack, written in Emberjs
- **Open Fabrics Alliance: Open Fabrics Management Framework**
  - Using mockups and the Swordfish API Emulator for OFMF development
DMTF Redfish Open Source Projects

- **Redfish-Mockup-Creator**
  - Creates a Mockup from a live Redfish or Swordfish service

- **Redfish-Interface-Emulator**
  - Emulates a Redfish service statically or dynamically

- **Redfish-Tacklebox**
  - Python utilities for common management operations on a Redfish service

- **python-redfish-library**
  - Python library for interacting with a Redfish service

- **libredfish**
  - C client library for interacting with a Redfish service

- DMTF open source projects are at [https://github.com/DMTF](https://github.com/DMTF)
fishem -- Fish Emulator

- Brings Redfish and Swordfish mockups to life
  - Reads in a mockup to set the initial state of the emulator, then handles REST operations on all objects in the mockup
  - Basic handling of allowed GET, PUT, POST, PATCH and DELETE operations for all URI-accessible objects defined by Redfish and Swordfish schema
  - Basic handling of Actions defined by Redfish and Swordfish schema
    - Detected, responded to with HTTP responses, and reported to the console
    - Can capture the final state of an emulator run as an output mockup
- Link: [https://github.com/ddeel/fishem](https://github.com/ddeel/fishem)
- Includes installation, user, and developer documentation
fishem Example Console Output
(Startup with an input mockup)

```
(env) C:\Users\Don\Documents\GitHub\fishem>python fishem.py -im temp\nvmeof-mockups
fishem initialization .................................
Loaded the mockup in "temp\nvmeof-mockups"
fishem starting ........................................
............................. (210 API modules)
fishem running .................................
  * Serving Flask app 'fishem' (lazy loading)
  * Environment: production
    WARNING: This is a development server. Do not use it in a production deployment.
    Use a production WSGI server instead.
  * Debug mode: off
  * Running on all addresses.
    WARNING: This is a development server. Do not use it in a production deployment.
  * Running on http://192.168.1.7:5808/ (Press CTRL+C to quit)
127.0.0.1 - - [05/Sep/2021 12:11:46] "GET /redfish/v1 HTTP/1.1" 200 -
```
fishem Example Client Browser Output
(Service root at /redfish/v1)

```json
{
    @Redfish.Copyright: "Copyright 2014-2020 SNIA. All rights reserved.",
    @odata.context: "redfish/v1/$metadata#ServiceRoot.ServiceRoot",
    @odata.id: "redfish/v1/",
    @odata.type: "#ServiceRoot.v1_10_0.ServiceRoot",
    Id: "RootService",
    Name: "Root Service",
    RedfishVersion: "1.12.0",
    UUID: "9238f634-2938-2342-882b-4b92a69c0423",
    Chassis: {
        @odata.id: "redfish/v1/Chassis"
    },
    Fabrics: {
        @odata.id: "redfish/v1/Fabrics"
    },
    NVMEndpoints: {
        @odata.id: "redfish/v1/NVMEndpoints"
    },
    Storage: {
        @odata.id: "redfish/v1/Storage"
    },
    StorageSystems: {
        @odata.id: "redfish/v1/StorageSystems"
    },
    Systems: {
        @odata.id: "redfish/v1/Systems"
    }
}
```
SNIA Swordfish Conformance Testing and Open Source
SNIA Swordfish™ Conformance Test Program

- SNIA's Storage Management Initiative (SMI) Conformance Testing Programs allow manufacturers to test their products with a vendor neutral, open source test suite to validate conformance to SNIA's storage management specifications.

- The **SNIA Swordfish™ Conformance Test Program** (Swordfish CTP) validates that a company's products conform to a particular version of the Swordfish specification using the new Swordfish CTP Test Suite.

- Swordfish CTP is based upon an open source framework that leverages common test tools that support the DMTF Redfish® Specification, which is extended by the Swordfish™ specification.
  - **Redfish-Protocol-Validator**, **Redfish-Service-Validator**
  - **Redfish-Interop-Validator**, **Redfish-URI-Validator**, etc.
SNIA Swordfish™ Conformance Test Program (Continued)

- Swordfish CTP includes extensions to cover storage-specific use cases and validate conformance to Swordfish profiles
  - Uses the Swordfish Features Registry to determine which profiles to test
  - Can also test specific profiles
- Companies with products that pass Swordfish CTP testing can be listed on the public SNIA web site, with information that includes:
  - Version of test taken
  - Software product tested
  - Hardware manageable by the tested software product
- The Swordfish v1.2.2 CTP Test Suite is now available
Where to Find More Information
Where to Find More Information

**SNIA Swordfish™**
- Swordfish Standards
  - Schemas, Specs, Mockups, User and Practical Guides, …
    [https://www.snia.org/swordfish](https://www.snia.org/swordfish)
- Swordfish Specification Forum
  - Ask and answer questions about Swordfish
- Scalable Storage Management (SSM) TWG
  - Technical Work Group that defines Swordfish
  - Influence the next generation of the Swordfish standard
  - Join SNIA & participate: [https://www.snia.org/member_com/join-SNIA](https://www.snia.org/member_com/join-SNIA)
- Join the SNIA Storage Management Initiative
  - Unifies the storage industry to develop and standardize interoperable storage management technologies
    [https://www.snia.org/forums/smi/about/join](https://www.snia.org/forums/smi/about/join)

**DMTF Redfish™**
- Redfish Standards
  - Specifications, whitepapers, guides,…
    [https://www.dmtf.org/standards/redfish](https://www.dmtf.org/standards/redfish)

**Open Fabric Management Framework**
- OFMF Working Group (OFMFWG)
  - Description & Links [https://www.openfabrics.org/working-groups/](https://www.openfabrics.org/working-groups/)
  - OFMFWG mailing list subscription
    [https://lists.openfabrics.org/mailman/listinfo/ofmfwg](https://lists.openfabrics.org/mailman/listinfo/ofmfwg)
- Join the Open Fabrics Alliance
  [https://www.openfabrics.org/membership-how-to-join/](https://www.openfabrics.org/membership-how-to-join/)

**NVM Express**
- Specifications [https://nvmexpress.org/developers/](https://nvmexpress.org/developers/)
- Join: [https://nvmexpress.org/join-nvme/](https://nvmexpress.org/join-nvme/)
Please take a moment to rate this session.

Your feedback is important to us.
More About the Swordfish API Emulator
More About the Swordfish API Emulator

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

- Python 3.6 or above
- Virtual environment 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”

- There is a setup.sh to handle these steps
Swordfish API Emulator Console Output
(Default Configuration)

```
$ python ./emulator.py
INFO:root:Mockup folders
['Mockups']
  * Redfish endpoint at localhost:5000
  * Using dynamic emulation
  * Use HTTP
  * Running in Redfish mode
  * Serving Flask app "g" (lazy loading)
  * Environment: production
    WARNING: This is a development server. Do not use it in a production deployment.
    Use a production WSGI server instead.
  * Debug mode: off
INFO:werkzeug: * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```
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 some 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, PUT, GET, PATCH, DELETE)

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

- Emulator-only operations 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 emulator-only operations 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

Dynamic Resources

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

Static Resources

- Chassis.py
- ComputerSystem.py
- StorageServices.py

api_emulator\redfish

api_emulator\redfish\templates
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 file pair is created, it is added to the emulator by editing `resource_manager.py`
  - This will attach the new resource’s APIs to URIs
Thank You!