Baiting the Hook: Navigating the Swordfish Waters for Implementers
(How to Get Started Building Swordfish)

George Ericson
Dell Inc.
This Session

- Get pointers to online resources, communities and tools
- Learn the different elements of the Swordfish distribution package how to use them
  - From the spec, to the mockups, to the user’s guide
- Learn resources for building a Swordfish service
- Experience… so far
Building blocks

- The implementer should not expect to just read the Swordfish spec's and then start implementing.
- Both OData and Redfish are critical to understanding Swordfish.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Common Schema Definition Language</th>
<th>Data Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>OData</td>
<td>Swordfish</td>
<td>Redfish</td>
</tr>
</tbody>
</table>

Swordfish

Redfish

OData
## Specifications and examples

Before and during implementation… Read the documentation…

<table>
<thead>
<tr>
<th>OData</th>
<th>Redfish</th>
<th>Swordfish</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>Specifications</td>
<td>Specifications</td>
</tr>
<tr>
<td>URL Conventions</td>
<td>White Papers</td>
<td>User’s Guide</td>
</tr>
<tr>
<td>CSDL XML</td>
<td>Mockups</td>
<td>Mockups</td>
</tr>
<tr>
<td>CSDL JSON</td>
<td>Schemas</td>
<td>Schemas</td>
</tr>
<tr>
<td>JSON Format</td>
<td></td>
<td>Practical Guide</td>
</tr>
<tr>
<td>Vocabularies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tutorials</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The embedded URLs locate content distributed with each of these standards.
Your first steps

- Understand the use cases that you want to support
- Decide what elements of the models need to be implemented
  - Do you need events, logging, task management?
  - What features are you implementing
  - So on…
- Look at the Mockup examples
Mockup considerations

- Mockups are illustrative
  - Do not expect a mockup to represent a complete real system
  - Mockups are simplified snapshots of some portion of an implementation
  - Mockup values represent state at a particular point in time

- Mockups use a simple key structure to make implementation on a filesystem easy
  - Real implementations should use globally unique keys where possible
    - Solves several real-life issues, including scale, reuse, and uniqueness
  - Real implementations should use the parenthesized key style
  - Mockups use the 'key as segment' style
    - OK, but not as robust
    - Mockup keys are often simple ordinal indexes.
Next, choose an implementation infrastructure

- We have worked with several variants:
  - Redfish native
    - RackHD
    - Mockups
    - other
  - OData native
    - Olingo
    - JayStack
    - ODatacpp-Server
- Commercial Redfish frameworks are becoming available
  - Example: Insyde software
Redfish service infrastructures

- We've used several existing infrastructures that already had some Redfish support
  - Each utilized simple URL matching and processing
  - No built-in understanding OData metaschema
  - Some support projection (i.e. Select)
- These implementations were NOT easily extended to support enterprise class use cases
OData service infrastructures

- **Olingo**
  - This is an open source Java implementation
  - Fully supports the OData metaschema
  - Fully supports the OData query language
  - We have implemented a Swordfish API

- **Jaystack**
  - This is an open source Node.js implementation
  - Fully supports the OData metaschema
  - Fully supports the OData query language
  - We do not have implementation experience with Jaystack

- **Odatacpp-server**
  - This is an open source C++ based implementation
  - After some work, we found that it is not sufficiently complete to serve as a base for our implementations
    - Based on GitHub history, we don't expect that this to change
Discovery

- Redfish requires the Simple Service Discovery Protocol (SSDP)
  - Assure that your IT administrators will allow the SSDP traffic on their networks
Implementation considerations
When to use Swordfish extensions

- For management of storage systems including:
  - Arrays
  - Virtualizers
  - File Servers
  - Volume managers

- If the managed systems support
  - Replication
  - Service based provisioning
  - Mapping volumes to hosts
  - Storage pools

- If consistent interfaces are required to span a wide range of supported storage products, from simple to advanced
Swordfish implementation considerations

- A Swordfish data service is a Redfish data service
- Classes of service should be used to avoid exposing implementation details via the user interface
  - Examples include critical data protection and IO performance features
- Implementation feedback is requested
  - Fixes and enhancements can be made rapidly, but only if you request them.
    - For example, one implementer commented that when multiple actions are supported against a resource, the choice of organizing properties by-value vs by-reference can make important differences in ease of use.
      - However, no specific examples were cited… => POST an ISSUE…
Redfish
Implementation considerations
When to use a Redfish data service

- For management of compute infrastructure, including
  - Bare-metal discovery
  - System Configuration
  - Monitoring and management of all common hardware components
- As a base for accessing network management services
- As a base to support Swordfish APIs
Redfish data service considerations

- Redfish requirements specify a subset of an OData data service
- Provides interfaces for common auxiliary services, including authorization, event, log, and task management
- Must allow [http://host/redfish/v1/](http://host/redfish/v1/) to address the Redfish entity type named ServiceRoot
  - ServiceRoot extends (by copy) the OData service root named ServiceContainer
  - The OData service root is at [http://host/redfish/v1/odata](http://host/redfish/v1/odata)
- Implementers should be aware that the value of the "odata.id" of each entity must identify that entity within at most one Singleton or EntitySet
- Most Redfish features are optional.
  - For example, you can decide whether or not to support managing and updating installed software, including operating system software and device drivers
Choosing implementation infrastructure

- Redfish native implementations are sufficient to implement the features specified by Redfish
  - More complex implementations may require additional features
- Recommendation: Services that require additional features should utilize an OData conformant service infrastructure
- OData conformant service implementations must consider
  - Subtypes of the Redfish resource collection have special rules and behaviors
  - The semantics implied by the placement of NavigationProperties, either inside or not inside a complex type named "Links"
Considerations for the 'Links' complex type

- An OData native implementation will not enforce the semantics associated with complex types named Links
  - Redfish divides related entities into two classes
    1. The entity referenced by a NavigationProperty is directly contained (subordinate) in the referencing entity
      - CSDL specifies this case with the attribute `ContainsTarget="true"`
    2. The entity referenced by a NavigationProperty is not directly contained in the referencing entity
  - References to entities of the:
    - First type are placed in the main body of an entity type
    - Second type are placed in the body of a complex type
      - This becomes a property in the main body of an entity type
      - In most cases, this complex type is named "Links"
    - If used, "Links" may only contain NavigationProperties of the second type
Considerations for 'ResourceCollection' types

- An OData native implementation will not natively understand the Redfish defined special rules for ResourceCollection
  - Each subtype has exactly one NavigationProperty.
    - It is a collection named Members
  - A POST to a resource collection resource is specified to have the same OData semantics as a POST to its Members collection
- In all other respects, resource collection subtypes are conformant with OData
Additional considerations

- Backwards compatibility to legacy tools
  - Events can utilize MessageArgs to hold additional information
  - Additional information may be placed in OEM properties

- Be aware of ongoing work in progress
  - For example: Communicating the implementation limits on numbers of Accounts, Roles or Event subscriptions
  - Participate in Redfish and Swordfish standards
End