Agenda

- Firmware Update Model
  - Update Service
  - Firmware Inventory
- Simple Update - pull update
- Multi-part HTTP Push update
- Unstructured HTTP Push update
Firmware Update Model

- **Update Service**
  - The service for updating firmware
  - Firmware updates are accomplished by invoking a Update Service action or performing HTTP Post's

- **Firmware Inventory**
  - Software components generally referred to as platform firmware

- **Software Inventory**
  - Software components executed in the context of a host operating system
  - Such as device drivers, applications, or offload workloads
Firmware Package Transfer Methods

- **Pull update**
  - The Redfish service is provided a URI from which to obtain the firmware update package
  - Also known as Simple Update

- **Push update**
  - The Redfish service is provided the firmware update package as part of the HTTP POST request
Simple Update - pull update

- This fragment is present in the UpdateService resource when SimpleUpdate action is supported.

```
{
   "Actions": {
      "#UpdateService.SimpleUpdate": {
         "target": "/redfish/v1/UpdateService/Actions/SimpleUpdate",
         "@Redfish.ActionInfo": "/redfish/v1/UpdateService/SimpleUpdateActionInfo"
      }
   }
}
```

- The 'target' property specifies the target of a HTTP POST.
- The SimpleUpdateActionInfo resource describes the parameters that can be included in the HTTP POST message.
Parameters for Simple Update

- If these parameters are present in SimpleUpdateActionInfo resource, they can be included in the SimpleUpdate HTTP POST message
  - **ImageURI** specifies the URI of the firmware update package
  - **TransferProtocol** specifies the transfer protocol associated with the URI
  - **Targets** specifies a list of resources to be updated with the firmware update image
  - **User** and **Password** specify the credentials for accessing the firmware update package
Multi-part HTTP Push Update

- This fragment is present in the UpdateService resource when the Multi-part HTTP Push Update action is supported

```
{
    "MultipartHttpPushUri": "/redfish/v1/UpdateService/update-multipart",
    "UpdateParameters": { … }
}
```

- The 'MultipartHttpPushURI' property specifies the target of a HTTP POST
- The parts of the multi-part message are described in the table

<table>
<thead>
<tr>
<th>Part</th>
<th>Req'd</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update parameters</td>
<td>Yes</td>
<td>JSON-formatted part for passing the update parameters. See the &quot;UpdateParameters&quot; property.</td>
</tr>
<tr>
<td>Update file</td>
<td>Yes</td>
<td>The opaque blob which contains the firmware image(s) use for the update.</td>
</tr>
</tbody>
</table>
Example: Multipart HTTP push update request

POST /redfish/v1/UpdateService/update-multipart HTTP/1.1
Content-Type: multipart/form-data; boundary=-----------------------------d74496d66958873e
Content-Length: <computed-length>

-----------------------------d74496d66958873e
Content-Disposition: form-data; name="UpdateParameters"
Content-Type: application/json

{}

-----------------------------d74496d66958873e
Content-Disposition: form-data; name="UpdateFile"; filename="bmc_update.bin"
Content-Type: application/octet-stream

<opaque blob>

• Note: the UpdateFile can contain one or more firmware images
Unstructured HTTP Push Update - deprecated

- This fragment is present in the UpdateService resource when the Unstructured HTTP push update is supported

```json
{
   "HttpPushUri": "/redfish/v1/UpdateService/update",
   "HttpPushUriTargets": [],
   "HttpPushUriTargetsBusy": false,
   "HttpPushUriOptions": {
      "HttpPushUriApplyTime": {
         "ApplyTime": "Immediate",
         "MaintenanceWindowStartTime": "2018-12-01T03:00:00+06:00",
         "MaintenanceWindowDurationInSeconds": 600
      }
   },
   "HttpPushUriOptionsBusy": false,
   ...
}
```

- Due to the vendor-specific details of this operation, this method was deprecated in favor of multipart HTTP push update
Redfish Client TaskService Flow

1. **Client starts an update** (push or pull)
   - Task1 is created
   - POST returns 202 (Accepted) with a TaskMonitorURI

2. **Client monitors updates**
   - Client monitors (GETs) TaskMonitorURI
   - Task1 may include TaskStarted and TaskProgressChanged messages

3. **Update completes**
   - Task1 may include TaskCompletedOK message
   - GET on URI returns a 200 (with messages)

4. **Update faults**
   - Task1 may include TaskCompletedWarning message
   - GET on URI returns a 2xx or 4xx (with messages), 5xx

Task Message Registry
- "TaskStarted" {taskId}
- "TaskCompletedOK"
- "TaskCompletedWarning"
- "TaskProgressChanged" {taskId, %}
Thank you for watching!

- Redfish Community
  - Bulletin board
  - http://redfishforum.com

- Redfish Developer Hub
  - Redfish Explorer, Hosted schema at Namespace, Educational Material
  - http://redfish.dmtf.org

- Redfish Forum
  - Schemas, Specs, Mockups, White Papers, FAQ & more
  - http://www.dmtf.org/redfish
Firmware Update Methods

• Simple Update - pull update
  • Redfish 2016.3 model extensions

• Multi-part HTTP Push Update
  • Redfish 2019.2 model extensions

• Unstructured HTTP Push Update (deprecated)
  • Redfish 2017.1 model extensions