Hewlett Packard
Enterprise

NEO REDFISH/SWORDFISH MONITORING API

May 15, 2023

Dan Matthews

CS API OVERVIEW

- Redfish/Swordfish Brief Overview
- NEO RFSF API Deployment and Access Details
- Request Data Flow
- Architectural Support for Extension
- Resource Tree Representation of the Cluster
  - Discovering Resources
  - Cluster Component Resource Representation
- API EventService
  - Generic Event Subscriptions
  - MetricReport Event Subscriptions
  - MetricReport Data Flow Example
  - Client Side EventDestination “Receivers”
- API Client Side Libraries
- References

REDFISH/SWORDFISH BRIEF OVERVIEW

- Redfish (DTMF):
  - Standard and application programming interface (API) designed to deliver simple and secure management for converged, hybrid IT, and the Software Defined Data Center (SDDC)
  - Provides RESTful interface semantics to access schema based data model to conduct management operations
  - Suitable for a wide range of devices, from stand-alone servers, to composable infrastructures, and to large-scale cloud environments.
  - Redfish Spec: https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.11.1.pdf
  - Redfish Schemas: https://redfish.dmtf.org/schemas/v1/

- Swordfish (SNIA):
  - Extension to Redfish Scalable Platforms Management API
  - Defines comprehensive, RESTful API for storage management that addresses:
    - Block storage
    - File systems
    - Object storage
    - Storage network infrastructure
  - Swordfish Spec: https://www.snia.org/sites/default/files/technical-work/swordfish/release/v1.2.3/html/Specification/Swordfish_v1.2.3_Specification.html
  - Swordfish Schemas: https://redfish.dmtf.org/schemas/swordfish/

NEO RFSF API DEPLOYMENT AND ACCESS DETAILS - CURRENT

- API deployed on mgmt nodes as set of 6 systemd service managed by HA as an active/standby group
  - Active on mgmt=primary node, standby on mgmt=secondary node
  - Follows failover / failback of md64-group

```
  * Resource Group: grp-rfsf-api:
    * prm-dp-api        (systemd:dp-api):        Started kjlmo1200
    * prm-hw-ec (systemd:hw-ec):         Started kjlmo1200
    * prm-dm-mgmt-ec    (systemd:dm-mgmt-ec):    Started kjlmo1200
    * prm-fs-mgmt-ec    (systemd:fs-mgmt-ec):    Started kjlmo1200
    * prm-raid-mgmt-ec  (systemd:raid-mgmt-ec):  Started kjlmo1200
    * prm-monitoring-ec (systemd:monitoring-ec):         Started kjlmo1200
```

- Accessible over HTTPS via port :8081 across the public, internal mgmt, and localhost networks
  - Use same self signed cert/key that other CS components behind TLS use
  - Users may upload their own CA signed certs if they choose
- Accessible for valid clusterstor admins users that have been granted an access token
  - Token granted via a POST request to /redfish/v1/SessionService/Sessions to create a Session
  - POST request body contains user credentials
  - Credentials validated against local cluster LDAP instance
  - If accepted, response header contains valid JWT
  - Token passed in subsequent request header in order to access API

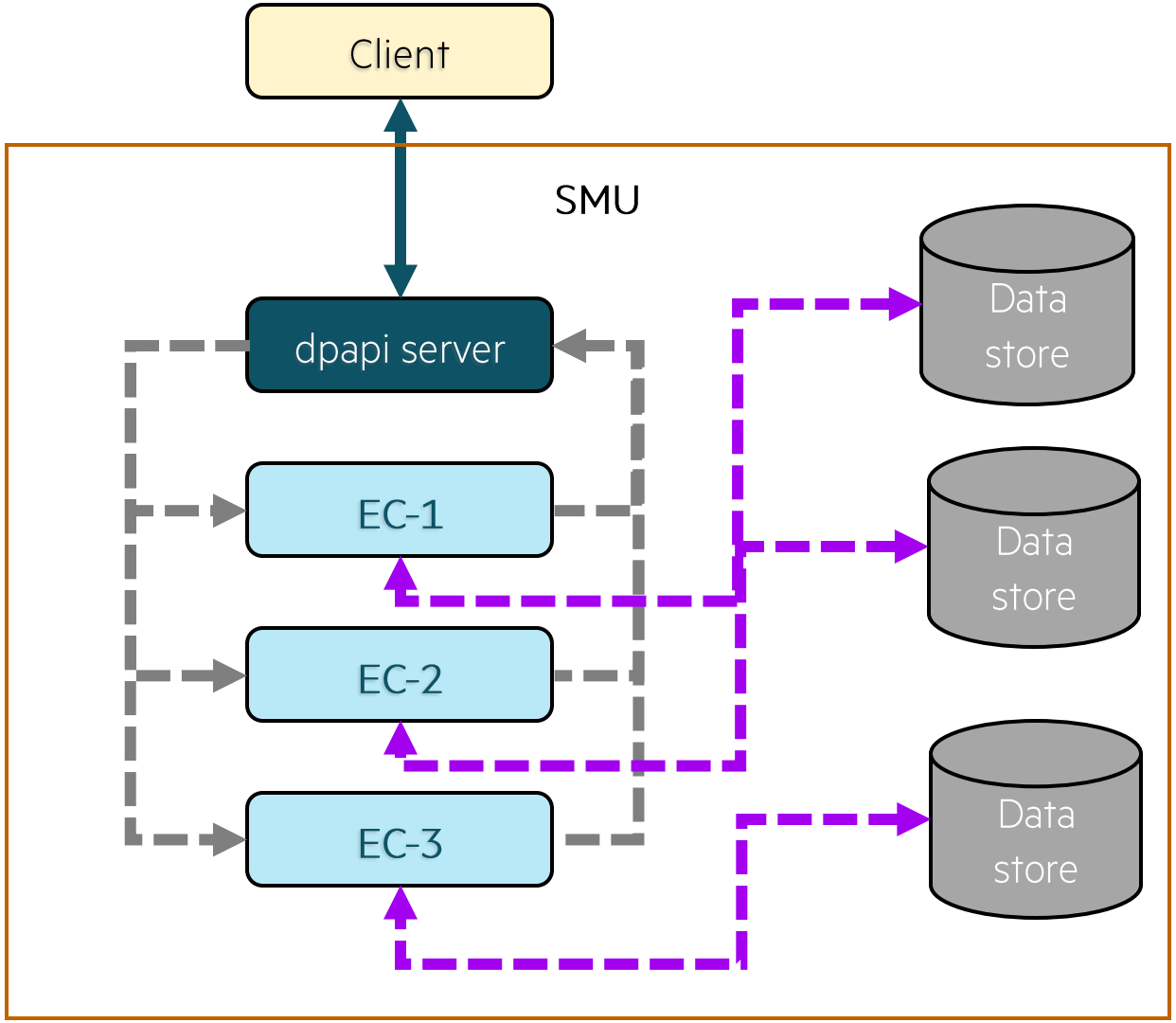
NEO RFSF API DEPLOYMENT - FUTURE

- Dp-api-node instances exist on the nodes to handle requests coming in from mgmt node
  - Design is symmetric i.e. management node vs storage/custom node
  - Goes through exact same router with different EC backend
- Element Controllers moving out to the nodes to provide node local resource reporting and control
  - Element Controllers on ClusterStor have up till now have been resident on the management plane
  - Remove hard to control data collection components to be replace by EC's
- Element controllers and backend data sources can be swapped in and out at node or mgmt level without changing interface / entry point

REQUEST DATA FLOW - CURRENT

- The below is a step by step overview of the flow of requests
  - EC = Element Controller which is responsible for one or more Redfish/Swordfish resources (usually on branch of the resource tree or a subset of a branch) including:
    – GETs, PUTs, POSTs, Events, Actions etc.

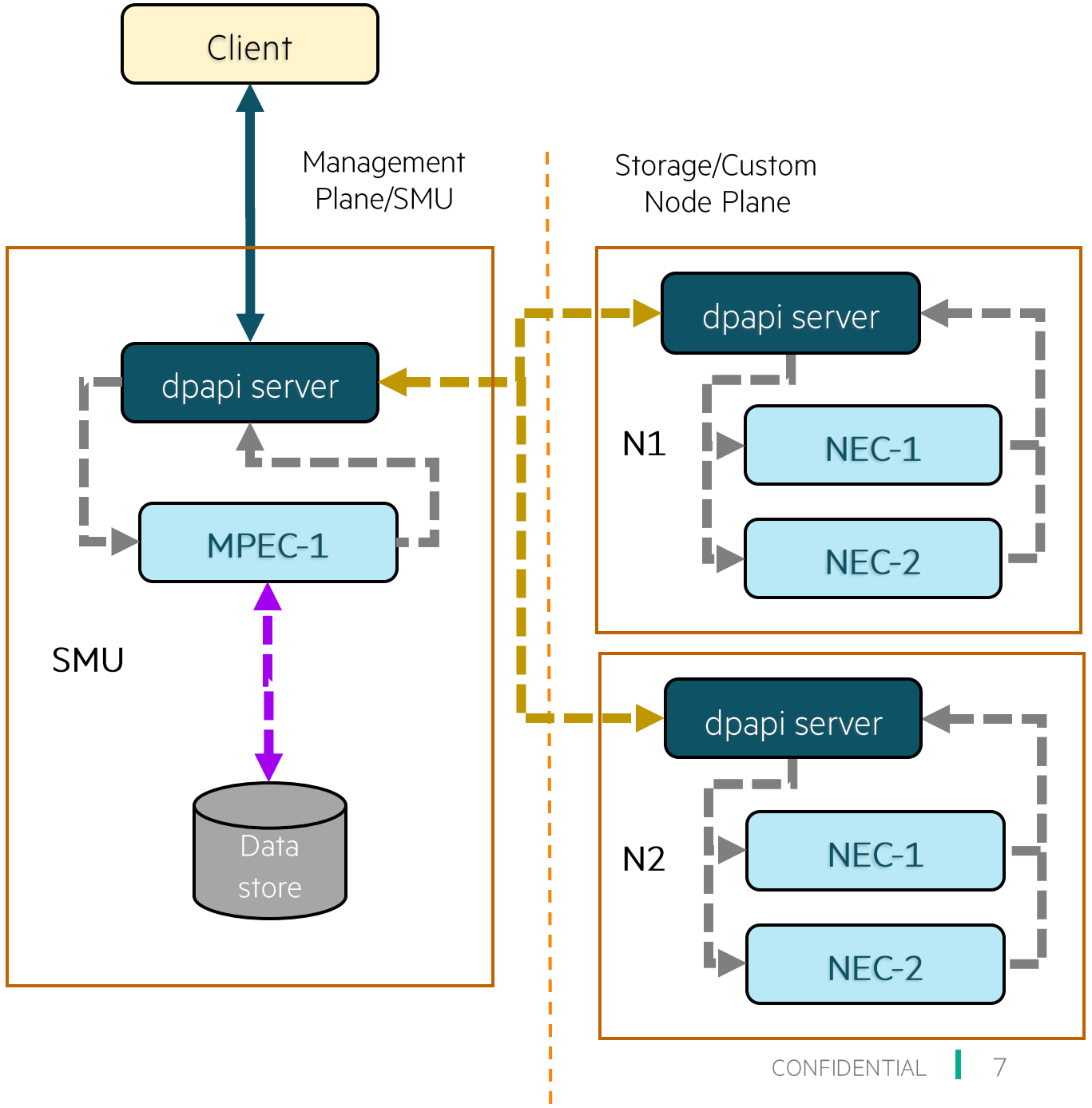
1. Client sends RESTful request to dp-api-server on mgmt node over https
2. API routes request to appropriate EC over gRPC
3. EC fetches data from appropriate backend data source
4. EC returns response to dp-api-server over gRPC
5. dp-api-server returns response to client over https



REQUEST DATA FLOW - FUTURE

- The below is a step by step overview of the flow of requests
  - MPEC = Management Plane Element Controller
  - NEC = Node Element Controller
    Note: an EC is an EC is an EC – the added prefixes are just for clarity

1. Client sends RESTful request to dp-api-server In the Management Plane
2. API routes request to appropriate EC over gRPC or routes the request to the Node dpapi server which routes the request to it's local EC
3. EC fetches data from appropriate backend data source
4. EC returns response to dp-api-server over gRPC which will either be returned directly to the client or pass back to the Management dpapi server for return to the client



ARCHITECTURAL SUPPORT FOR EXTENSION

- API is designed in such a way that it is easily extensible
- API front end interface remains the same
- Backend ECs and data collection sources are modular
- API endpoints and routes are known and remain the same (Resource Tree)
- Data models are known and remain the same

Future

- Extend API instances onto Storage Nodes (in process)
- EC auto insertion (remove fixed gRPC ports)
  - The facilitates easy addition after the fact of new/updated EC's
- Proxying to BMC/iLO to provide single point of data delivery

RESOURCE TREE REPRESENTATION OF THE CLUSTER

- All Cluster Components and Collections represented as a Redfish/Swordfish Resource
  - Known Schema based data model (json format)
  - All Resources are uniquely identifiable
- All resources within the “Resource Tree” are discoverable from the ServiceRoot: /redfish/v1
- ServiceRoot provides path to “top-level” resource collections:
  - /redfish/v1/Chassis
  - /redfish/v1/StorageSystems
  - /redfish/v1/StorageServices
  - /redfish/v1/Events
- ServiceRoot provides path to ”top-level” Service resources and their collections:
  - /redfish/v1/SessionService
    - /redfish/v1/SessionService/Sessions
  - /redfish/v1/EventService
    - /redfish/v1/EventService/Subscriptions
  - /redfish/v1/UpdateService
    - /redfish/v1/UpdateService/SoftwareInventory

RESOURCE TREE REPRESENTATION OF THE CLUSTER CONT.

- Each resource in a collection may provide sub-collections
- Examples:
  - /redfish/v1/StorageSystems/{ComputerSystemId}/NetworkInterfaces
  - /redfish/v1/StorageServices/{StorageServiceId}/FileSystems
  - /redfish/v1/Chassis/{ChassisId}/Thermal/Fans
- More info of full resource tree available in documentation

EVENTING

- Redfish enables clients to receive messages outside of the normal request / response paradigm
- The EventService uses these messages, or events, to asynchronously notify the client of a state change or error condition
- Push-style Eventing
  - When the service detects the need to send an event, it calls HTTP POST to push the event message to the client.
  - Clients can enable reception of events by creating a subscription entry in the Event Service
- Two “categories” of Event subscriptions implemented in NEO RFSF API:
  - “Generic” Events
    - Similar to alerts
    - Sent asynchronously as they occur
    - Occur on an API resources
    - Can filter on specific resources
    - Subscribers are unique
  - MetricReport Events (telemetry and statistic based events)
    - Sent at client defined frequencies
    - Subscribe to specific data sets (defined in EventService)
    - Subscribers start or join a data stream

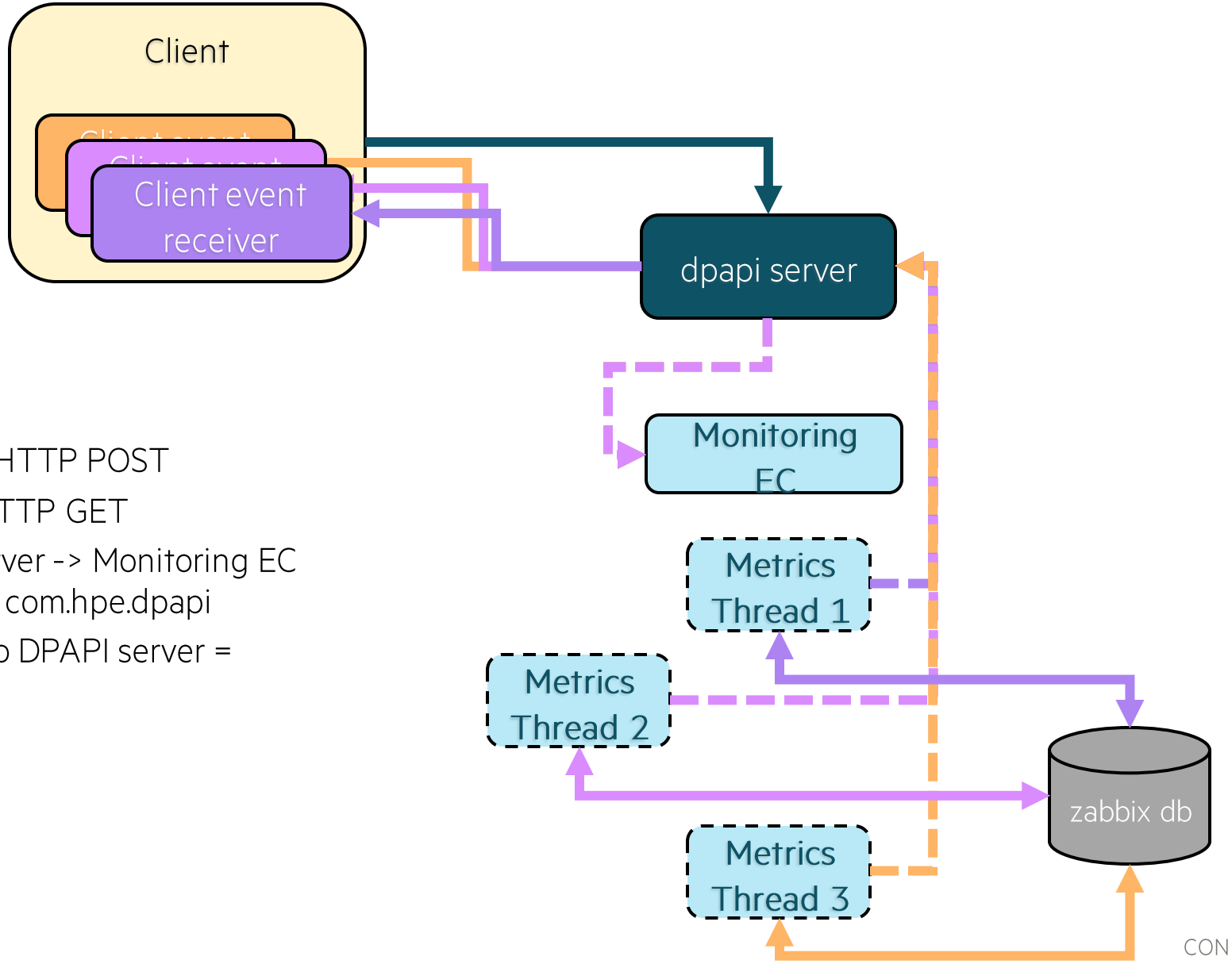
GENERIC EVENTS

- Events that occur on API resources
- Examples:
  - status change of a FRU
  - Start or stop of lustre targets
  - Failovers
  - Etc.
- A subscription to receive Events is created via a POST request to /redfish/v1/EventService/Subscriptions/{EventDestinationId}
- Can specify “filtering” information in EventDestination resource request body
  - Filter on ResourceTypes
  - Filter on OriginResources
- Runtime Event history maintained in /redfish/v1/Events collection
  - Lost on service restart or failover
  - “Rotated” after a certain max number of Events is reached
  - Events meant to be stored off box if required
- More details provided in documentation

METRICREPORT EVENTS

- Special case of EventService notifications to receive metrics and telemetry data
- A subscription to receive Events is created via a POST request to /redfish/v1/EventService/Subscriptions/{EventDestinationId}
  - Must contain EventTypes[“MetricReport”] in request body
  - Must contain MetricReportDefinitions in request body
    - Specifies data_set, send_interval, etc.
- Register to receive statistics for a certain data set
  - LustreStats
  - LinuxStats
  - JobStats
- Register to receive data from that data set at a specified interval
- Initial subscription for a data set “creates” the data stream at defined interval
- Subsequent subscriptions for a data set “join” the data stream

METRICREPORT DATA FLOW EXAMPLE



Connectors Key:

- Solid Uni-directional = HTTP POST
- Solid Bi-directional = HTTP GET
- Dashed from DPAPI server -> Monitoring EC = dbus message across com.hpe.dpapi
- Dashed from EC back to DPAPI server = gRPC POST request

CLIENT SIDE EVENTDESTINATION “RECEIVERS”

- A subscription to receive Events is created via a POST request to /redfish/v1/EventService/Subscriptions/{EventDestinationId}
- Request body passed is an EventDestination Resource
- The EventDestination.Destination field of the resource specifies a client side endpoint for API EventService to receive Events on
  - Events are sent from API via http POST
  - Client side Event “Receiver” must implement an HTTP POST request handler
- EventService defines parameters to control retries to send Events to a subscriber who is not listening

API CLIENT SIDE LIBRARIES

- Provided by rfsf-api-client package
  - Repo: https://github.hpe.com/hpe/hpc-sp-rfsf-api-client
- Provides libraries and config files to:
  - Connect to API
  - Authenticate a user
  - Subscribe to Receive Events
  - Subscriber to Receive MetricReports:
    - LustreStats, LinuxStats, JobStats
  - Discover full API resource tree
- Provides monitoring examples to:
  - Iterate through API resource tree
  - Start threaded client Event Receiver
    - Parse incoming Event messages
    - Send GET request to resource Event occurred on
- Provides example client side event receiver code

WHAT'S NEXT?!

- API instances out to all cluster nodes
- ClusterStor GUI and CLI moving to use API on backend everywhere
- Custom Metrics/Data Collection (POC completed but hasn't been exposed)
- Composability – provision/teardown of mid to high level resource constructs
- Firmware / Software updates via API

REFERENCES

- Redfish Spec: https://www.dmtf.org/sites/default/files/standards/documents/DSP0266_1.11.1.pdf
- Redfish Schemas: https://redfish.dmtf.org/schemas/v1/
- Swordfish Spec: https://www.snia.org/sites/default/files/technical-work/swordfish/release/v1.2.3/html/Specification/Swordfish_v1.2.3_Specification.html
- Swordfish Schemas: https://redfish.dmtf.org/schemas/swordfish/
- Neo Redfish/Swordfish REST API documenation: https://hpe.sharepoint.com/:w:/r/teams/hpc_storage/clusterstorteam/_layouts/15/Doc.aspx?sourcedoc=%7B488c0c22-30ca-4389-afd4-2f3ef4159c40%7D&action=edit&wdPid=534d6646&cid=23406062-e3b4-428b-a60f-45a843dc7fd6
-

FIN

Thank-you!

METRICREPORT DATA FLOW EXAMPLE