**Sunfish SC 23 Demonstration Flow**

Step 1

Description:

1. The Sunfish Server is up-and running in a clean state with no resources to be reported
2. The GUI shows a graphical representation of the RedFish tree with no resources available

Requirements:

* GUI showing RF tree state
* Sunfish Server

Missing:

* GUI showing RF tree state

Comments:

* In defining the GUI we should decide also how to represent the RF tree
  + Do we want the actual tree to be visible? It might be overwhelming
  + Do we want a list based approach were resources are shown per type? Perhaps an ad-hoc GUI could also show the device types we want to interact with in the demo.
  + Something to start from: <https://codesandbox.io/s/github/hoangph271/try-out-react-tree-graph>

Step 2

Description:

1. An Hardware Agent is started (CXL and/or NVMeOF)
2. The Agent registers with Sunfish by sending a *AggregationSourceDiscovered* event
3. Sunfish processes the event and pulls all resources reported into the RF tree under the new fabric created.
4. Sunfish creates an *AggregationSource* for the agent
5. The GUI shows the updated RedFish tree view

Requirements:

* GUI showing RF tree state
* Sunfish Server capable of supporting agent registration and upload of agent’s resources
* Hardware agents for CXL and NVMEoF fabrics

Missing:

* GUI showing RF tree state
* Sunfish server functionalities are available but not on the public GH yet. PR needs to opened and code merged into the main repo.
* Hardware agents for CXL and NVMEoF fabrics are available but not on the public GH yet. PR needs to opened and code merged into the main repo.

Comments:

* We can use the code developed by Michele for the registration as well as populating the RF by pulling the resources exposed by each agent
* We can re-use the mock agents created for the SDC presentation demo

Step 3

Description:

1. Users can request to attach either a memory chunk or a networked storage device to an existing system via the GUI.
2. GUI sends composition request to composition service
3. Composition service receiving the final state of the composition:
   1. the target system where devices are attached to
   2. Which CXL to use for serving the memory chunk and how big a memory chunk it is needed and/or
   3. The storage to be used
4. The Composition service translates the request into the relevant RF commands to materialize the composition, and forwards all commands to Sunfish
5. Sunfish identifies which agent is controlling which device and transparently redirects commands to the proper agent.
6. The agent replies a fake OK
7. Sunfish shows the updated view of the Composed system

Requirements:

* GUI showing RF tree state
* GUI allowing the possibility of creating a composition request
* Format for the composition request to be sent to the composition service
* Composition service
* Sunfish Server capable of identifying the agents to contact to execute each command

Missing:

* GUI
* Composition request format
* Composition Service
* Sunfish Server capable of identifying the agents to contact to execute each command

Comments:

* For this demo it is fair to assume that the user plays the role of the infrastructure manager and they will also decide exactly which devices to use in each composition request. In other words, no intelligence in the composition service.