Server

**UpdateVDSO()**

* Receives the new VDSO list from scheduler (client).
* (For each client, this list will be displayed)
* (Though the list of VDSO will be same for all)
* If the new list is same as previous one then don’t update.
* Else, updates own list and displays updated Stuff on UI.

**CreateSchedule()**

* Gets the timestamp & corresponding URL data from front end.
* Makes sure that no more than 1 URL provided for the same Timestamp.
* Sorts the timestamp data in ascending order.
* Saves the Timestamps and their corresponding URLs.

**ScheduleContent()**

* Gets the VDSO ids and schedule id from the front end.
* Creates a new ScheduleOrderID (Unique).
* Saves this data in scheduling Information.

**ProvideSchedule()**

* (Schedulers will hit this in every 5 seconds)
* Checks if new VDSO list is provided in request.
* If yes then updates the VDSO list by **UpdateVDSO()**
* Then returns the scheduling information.

Scheduler

**AddVDSO()**

* Appends VDSO object in the list of type [{vdso, URL}, …].
* (Here, second url is content displaying url)
* List’s index is the key for VDSO.
* Set the changeFlag to true. (If true then in polling, new list will be provided to server)
* Returns the VDSO id to respective VDSO.

**GetSchedule()**

* (It is called in every 5 seconds)
* If changeFlag is true, then add VDSO objects list in request.
* Send the request to the server at **ProvideSchedule()**
* Receives the schedule.
* If the scheduleOrderIndex is same than the previous one, then do nothing.
* If not then send the schedule information to all the respective VDSOs mentioned in response.

**ChangeContent()**

* (When VDSO changes the URL, this function is called)
* Updates own VDSO list with new URL by getting the VDSO id.
* Set the changeFlag to true.

**VDSO**

**CreateVDSO()**

* Creates VDSO object with video texture.
* Sets the video texture URL to Default video URL.
* Registers itself in Scheduler Object. (**AddVDSO()** method)
* It will return VDSO id. Save it.
* Also creates observers for receiving messages from scheduler.

**ReceiveSchedule()**

* Receives timings and their respective URLs.
* Gets lowerBound of timings at first.
* Sets the URL accordingly. (If timings index is -1 then still display default content)
* Save the next index of timings with URL.

**Update()**

* (Being called in renderLoop())
* If the current timing is greater or equal to next timing then change the URL to next URL.
* Update next index and nextURL accordingly.
* If no timing is left for next then continue displaying current content.
* If change occurred then call scheduler’s **ChangeContent()** with current VDSO id.