State Server

1. Proposed Schema for DSADM/ADL

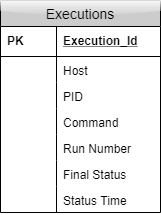
ADL

ADL will be the preferred go to place for any request made to state server in case there is a local cache miss,

ADL will only store the latest state of any execution request and not the drilled down details

i.e.

* Execution\_Id – the one generated by ADMIN ES.
* Host – The final host where the execution was running or is running.
* PID – The final PID with which the execution was holding.
* Command – as the name implies
* Run Number – The corresponding run number at which the execution executed successfully ie. If no retries are mentioned then it’ll always be set to 1.
* Final Status – whatever was the final state that was reached by the execution in its final attempt.
* Status\_time – the time at which the final status time was reached.



DSADM

This is our primary storage and is going to persist every information that is received by the state server, that’s why the schema is separated into two tables, ‘Executions’ table functions like the ADL schema and serves the front for getting the latest state and the ‘Execution\_Status’ serves as the one holding the detailed state changes and information regarding that.

Now depending on the scenario there are a few differences that would be intuitive based on the schema diagrams however the basic details would remain the same i.e.

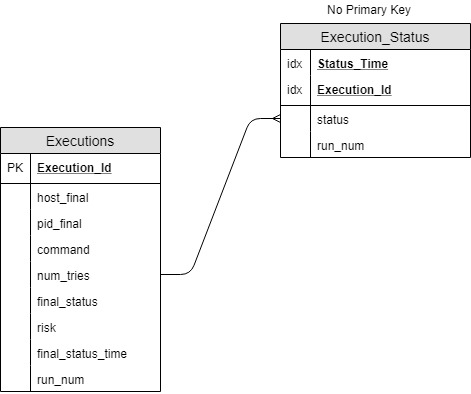
Executions Table

* Execution\_Id – the one generated by ADMIN ES.
* Host\_final – The final host where the execution was running or is running.
* PID\_final – The final PID with which the execution was holding.
* Command – as the name implies
* Num\_Tries – The corresponding run number at which the execution executed successfully ie. If no retries are mentioned, then it’ll always be set to 1.
* Final\_Status – whatever was the final state that was reached by the execution in its final attempt.
* Final\_Status\_time – the time at which the final status time was reached.
* Risk – taken straight off portal form (just felt important to hold this) \*\*\*
* Run\_num – The final attempt which it reached

Execution\_Status Table

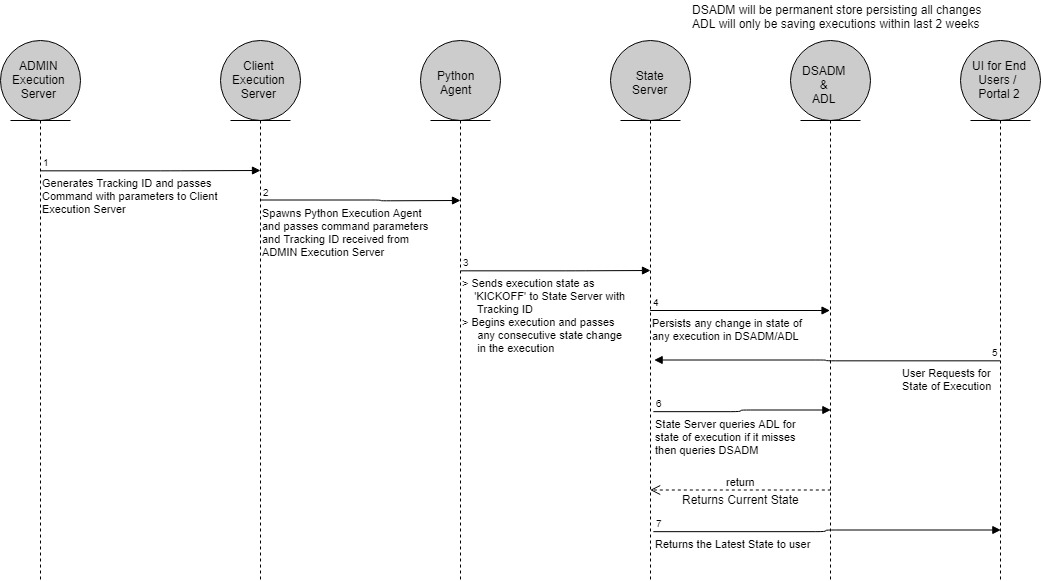
* Execution\_Id – the one generated by ADMIN ES. (to map it with Executions Table )
* status – every state that the execution ever reached
* status\_time – the time at which the status was reached.
* Run\_num – The final attempt which it reached
* Host ( will only be stored in scenario 2)
* PID (will only be stored in scenario 2)

DSADM – Scenario 1



In this scenario, the multiple retries on executions will be handled by the python agent that is spawned by the Client Execution Server.

1. State Server Requirements and Design Documentation



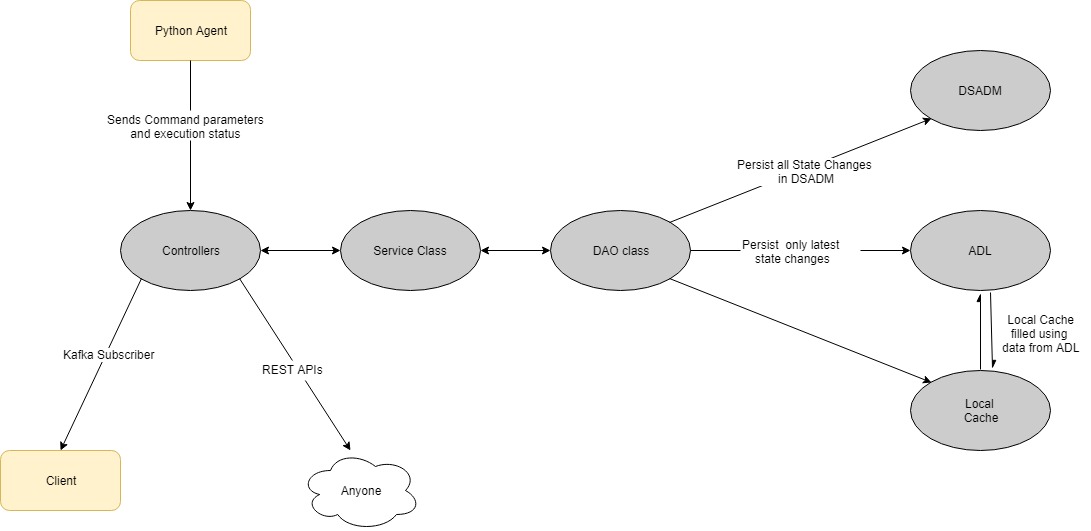
Interactions:

State Server & Data Stores ( Cache, ADL , DSADM )

* State Server will fill up its local cache initially with states of all executions that are currently running and also add any new executions’ status to its local cache, once the execution reaches a final state ie. AGENT FAILURE, SUCCESS or FAILED , then it drops the entry from the cache.
* For any execution that is not currently running, State Server will be fetching any requests for executions up to 30 days old which will be stored in an ADL repository.
* DSADM will be acting as the primary and permanent storage that will persist data regarding all executions and will be used to serve requests related to executions older than 30 days and more

State Server & Execution Agent

* In the initial phase State Server will be listening for the execution agents in a specified BMS source and will persist any state changes sent by the execution agent and storing them in data stores.
* Execution agent will ping the state of any execution that it fires every five minutes or whenever any state change occurs.
* Execution Agent will be the only entity that is allowed to send/update states of any executions in the environment, however in case an agent fails and unable to track a command that it began executing, State Server will wait up to 10 mins and if no update is sent by the execution agent, then it will mark the execution as lost and send an alert.



* To support existing use cases and as the de facto standard there will be BMS source on which State Server will be listening to for updates from the python agent

1. State Server APIs

For Client:

* GET getLatestExecutionState(Execution ID): retrieves latest state from ADL/cache and provides it to the client for the requested execution.
* GET getExecutionTransitions(Execution ID): retrieves all states and stages that the command went through and provides it to client.

For Execution Agent:

* POST saveExecutionState(Execution ID): saves the change in state in DSADM/ADL # semaphore request

State Server:

* State Server will also raise an alert for any execution that is missing or state of which has not been updated for 10 mins or more.