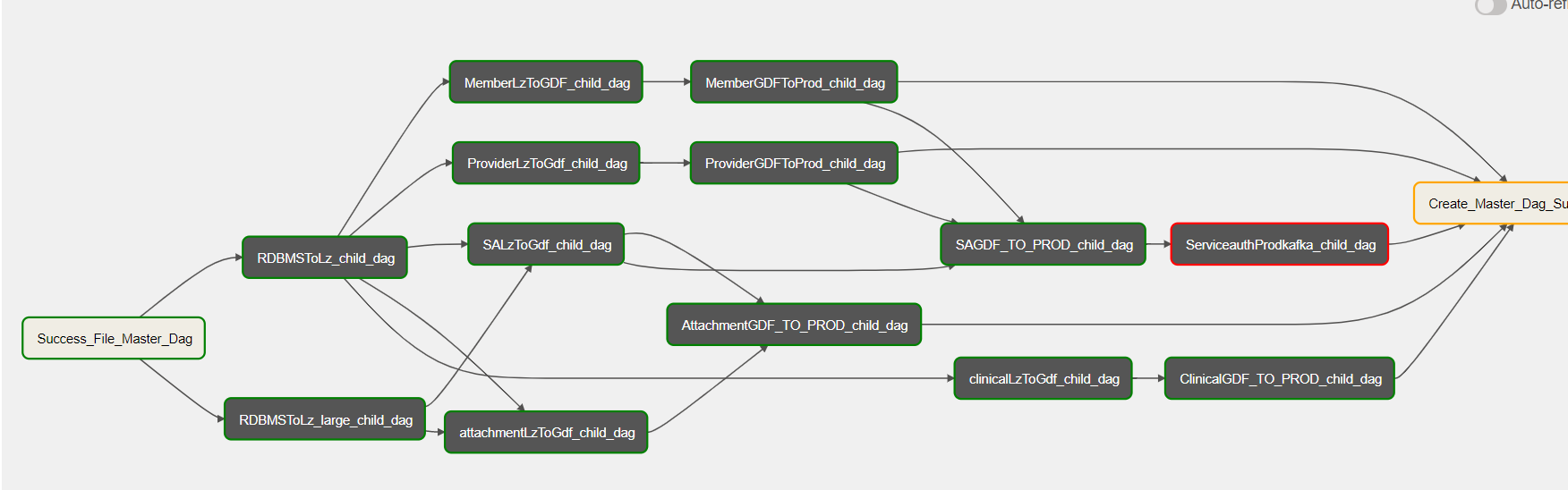
**DATA-8940: Options to create standalone domain specific DAGs – SA**

**Objective:**

As per the current airflow, SA data processing from gdf to prod layer (ccx\_master\_dag\_new\_approach) have dependency on member and provider load. As part of this spike, need to come-up with multiple options to manage dependency.

ccx\_master\_dag\_new\_approach:

**Ref:** [**https://d1undatahub001.ccx.carecentrix.com:6090/graph?dag\_id=ccx\_master\_dag\_new\_approach**](https://d1undatahub001.ccx.carecentrix.com:6090/graph?dag_id=ccx_master_dag_new_approach)



**Acceptance Criteria:**

Given the SA dag

When SA dag is dependent member patient dag

Then provide options to manage this dependency with some technical details and pros/cons for each approach

**Option 1:**

**Create SA in Prod layer but mark it as not ready for consumption. Each load rechecks such SA and if Member/Provider has arrived, then can link and mark it as ready for consumption**

* An existing ServiceAuth (prod.service\_auth) logic -query member & provider tables (from prod layer) and get gdf ids (Ref: ServiceAuthMerge.scala)
  + As per the logic, unmatched members can be reprocessed in next run.
  + Unmatched providers are getting rejected and those can be reprocessed manually.

**Technical Approach:**

* As per this approach, lets make an independent DAGs for member, Provider and ServiceAuth without any dependencies. Create a new process (using Spark/Scala) to monitor the data load into prod (using trigger file or logging.corelake\_checkpoint table) to link SA with member and provider gdfids.

**Required Changes:**

DAG: An existing dag need to be modified for removing dependency or create new dag for SA load.

Create a new process to monitor the prod data load (SA, Member and Provider).

At Code level: SA merge code (ServiceAuthMerge.scala) need to be modified for removing an existing logic for ‘gdfid’ linking and the removed logic need to be

implemented as a new process to link SA with member and provider.

**Pros:**

+ DAG will look very simple as no dependency on member and provider.

+ Processing time will be reduced as ‘gdf id’ linking logic will be removed.

**Cons:**

-ingest SA data quickly but not ready for consumption until its linked.

**Option 2:**

**Flatten and store Member/Provider details in the SA Domain object so linking is not needed. Enough details for Provider/Member is available in the flat structure for client to be able to use to fetch Member/Provider via separate calls to Data Hub.**

**Technical Approach:**

* As per this approach, lets make an independent DAGs for member, Provider and ServiceAuth without any dependencies.
* Remove the logic of linking ‘gdfid’ from SA merge code (ServiceAuthMerge.scala) – as per this gdfids will be populated as ‘NULL’ or ‘gdfid’ field will be removed. (SA data will be ready for consumption without gdfid.)
* The removed logic need to be implemented in downstream system / consumption module.

**Required Changes:**

DAG: As per this approach, an existing dag need to be modified for removing dependency or create new dag for SA load.

At Code level: SA merge code (ServiceAuthMerge.scala) need to be modified for gdf id population (gdfid will be ‘NULL’)

**Pros:**

+ DAG will look very simple as no dependency on member and provider.

+ Processing time will be reduced gdf id logic will be removed.

**Cons:**

- ‘gdfid’ will be populated as NULL / ‘gdfid’ field will not be present in SA.

**Option 3:**

**Do not load into prod and load only if Provider/Member are ready**

* Its an existing flow. No changes required in DAG
  + As per the logic, unmatched members can be reprocessed in next run.
  + Unmatched providers are getting rejected and those can be reprocessed manually.

**Required Changes:** Need to automate the process to reprocess the unmatched provider.

**Option-4:**

**Derive the ‘gdf ids’ in SA as same as member / provider logic.**

* An existing ServiceAuth have required fields from member and provider. As per this approach, gdfid creation logic need to be implemented in SA to derive ‘gdfids’ while processing SA records.

**Required Changes:**

* DAG: As per this approach, an existing dag need to be modified for removing dependency or create new dag for SA load.
* At Code level: SA in gdf need to be modified to query member & provider tables (from prod layer) and derive gdf ids

**Pros:**

+ SA in prod layer will not have any dependencies

+ SA in prod will be ready for consumption

**Cons:**

* More efforts required as changes to be made in code for gdfid derivation.

**Other Options on handling unmatched providers:**

Currently, unmatched providers are rejected and those need to be fixed manually. In order to automate this, mechanism need to be built for reprocessing the rejected providers as same as current member logic or ‘callback’ logic can be created for better performance.