1. **Audit trail of the requests for ODS**

Current situation:-

Accumulator HUB is the centralized storage system of the Accumulators for Medical, Dental, Vision & Pharmacy cost shares for HIX/PEX and EHB products. Any inquiry or update from different systems to accumulator HUB is done thru Services in real time or batch. The Services will just return the success or failure status to the calling system. It’s up to the source system to correct and resend the request.

Need for Audit trail creation:-

1. Whenever an inquiry or update request fails for some reason (it could be due to in-adequate data in the request, or processing issues or data base issue or application issue), the portal or source system will be notified that there is no data found or failure. There is no mechanism to take a look back and verify if it’s a system issue.
2. During the real time updates from different systems, when the response is slow, source system might be informed with failure status as services have a cut off time to get the response. Because of this, source system will retry the transaction which results in double updates. If there is an audit trail, these duplicate updates can be avoided.
3. In order to measure the usage/volume of requests to see if any improvements needed to support the large capacities.

Proposed solution:-

Proposal is to create the audit trial for any inquiry or update requests that comes to ODS from any system via both online or batch. This will be an asynchronous transaction triggered in the backend to avoid performance overhead. The logical data elements that will be tracked are,

1. Source system which made the request
2. Request type – Inquiry or Update
3. Request function
4. Request model online or batch
5. Job or transaction name
6. Timestamp of the request
7. Request data elements
8. Request status – Success or failure
9. Reason code – to track the reason for the failure
10. Response area

With the log db, reports can be created for all the failure transaction for further research. Based on the number of occurrences for each reason code, further improvement areas could be identified.

Purge process must be created to purge the data every 30 days/15 days as the expected volume will be higher.

Flow diagram:-



Risks:-

1. There will be too many records would be tracked in log DB as the intent is to capture each and every request for ODS. Need a trend analysis to see if anything can be excluded.
2. Possible performance issue on the system due to the additional overhead for audit trail creation.
3. **Duplicate insert into ODS**

Current scenario:-

When there are delays in the ODS gateway for the real time requests, the Service layers sends an incorrect responses to the source system. In those scenarios, the source system would try the same request second time. But, the current process is not capable enough to avoid this duplicate update.

Proposed approach:-

Option-1:

Leverage the Log Db itself to check for whether that transaction has been updated already. If it has been updated, do not allow the current transaction to update ODS.

Risk:-

1. Since the Log Db is to capture all logs for ODS, the size of the data will be too high. So, the records will be retained only for certain period.
2. Any source system wanted the duplicate updates for some reason will have to find a work around to send unique claim ID.

Option-2:

Build a new DUP data base to capture the update transactions. Whenever any transaction for update is received, it will be evaluated against this DUP db. Only when there is no DUP, it will be allowed for update. If at all any source system needs to allow the DUP, then that system must have some logic to generate unique claim ID for the update.

Risk:

1. Additional overhead in updating the DUP DB
2. Any source system wanted the duplicate updates for some reason will have to find a work around to send unique claim ID.