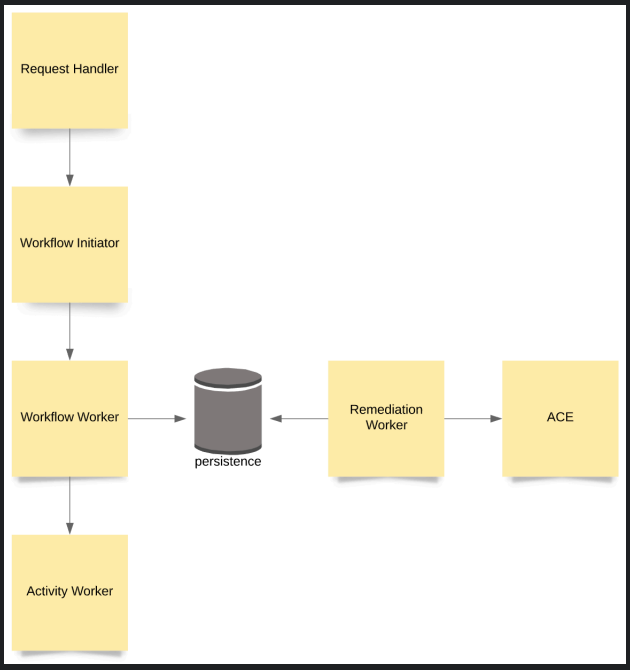
Fulfillment Approach

New name? FOF - Fault Oblivious Fulfillment, DEF - Durable Enterprise Fulfillment

Change log

- May 22 - v0

High-level View



Request Handler

This is the gateway component and should operate at the highest possible availability.

Operations:

- Accept a request
- Validate the request
  - A2A token - depends upon cbis.
  - If bad: 4XX
  - If good: 2XX, and create an event `initiate_workflow` on RTF. Include meta data that will be used by Workflow Initiator to persist.
- No DB calls
- Possibly access consul kv for some configuration.

- Align on api versioning, possibly have a stable entity url, and leverage accept-header to include a format version because its possible that the request payload may have multiple versions of format.
- Discuss
  - how to map incoming version to the appropriate request validator.
  - if request validator needs to be a lamda function

Workflow Initiator

This component subscribes to the `initiate_workflow` event and is a controller.

Operations:

- 200, 500
- Subscribe to `initiate_workflow` on RTF.
- May not need DB.
- Determine journey type from the request.
- Create `initiate_journey` on RTF. Add the filter to indicate the `type_of_journey`
- Discuss - do we need to handle multiple versions of same journey types.

Workflow Worker

- 200, 500
- This may be the first component that persist the state.
- Also called the Journey workflow or a business component.
- Subscribe to `initiate_journey` with appropriate filter on RTF.
- Fetch the journey workflow dag from database. We may need to think whether dag needs a version.
  - Discuss how the workflow dag can be a runtime configuration
  - Should this be consul kv?
- Create a instance of the journey dag for this request - saga id
- Persist the journey instance.
- Create the activity event for the first activity in RTF
- Subscribe to the success and failure for the first activity in RTF.
- Need to handle race condition that persistence may be slightly behind real-time processing.
- Fetch the journey instance dag, and progress to the next activity.
- We should see if the request header can be provided back to us on success or failure.
- Address TTL for persistence (success)

Activity Worker

- These are the bottom service layers, and can be SOR apis, Cornerstone etc.
- 200, 400, 500
- We may have to wrap some SORs to support idempotency. Possibly WCC.
- What happens when activity worker offers a new version of the endpoint?
    - What happens to on-going journey instances

Remediation Worker

This component is a timer-based scheduler, who queries DB for journey instances that are incomplete after a certain elasped time - typically configured or based on past trends of completions.

- May query RTF to determine the status of the journey instance.
    - what can we get from rtf?
    - can we get the servicenow ticket # from rtf?
- Only address journey instances that are in a servicenow state?
- May integrate with ACE to allow human-based remediation.
- May offer a status api that is either fulfilled from persistence or csrt
- We may need access to update the servicenow ticket with the ACE details.
- Address TTL for persistence (failures and incomplete)

Persistence

- We need active / active and we may not need cross-db replication
- Only accessible to the workflow worker and remediation worker.
- We need to handle race condition where persistence is behind real-time events handling.
    - Reads go local, and if not found, go remote, and if still not found, return 503 to RTF to cause exponential backoff.
    - Writes go primary and if fail, then go secondary. This will be configuration based to allow for planned outages which may remove persistence from any DC.
    - May be the saga-id can have an indicator to point to the DC where it was generated. This will allow us to route reads to appropriate DC.
- Figure out which access paths are KV based - Couchbase or Redis could work
    - `saga-id:<journey_instance document>`
    - `journey-type:<dag document>`
- Remediation Worker would need query index to determine failed flows. This includes servicenow, and 400s.

Cornerstone

- Need to establish hive persistence in FDP. This will atleast be 30 mins behind.
- Need to establish querying via CSRT for adhoc needs.

Other observations

- RTF doesn't use ELF, and relies on eCP Splunk
- For local environments,
  - Ability to mock RTF.
  - Option to mock certain activity workers
  - Offer in-memory persistence via docker?
- We may need to supplement persistence to support a variety of querying
  - Shouldn't this be in elf for 15 days and cornerstone for long-term?