FAM Moderation System Design

Background

In previous FAM Moderation Solution Discussion, we explored various potential options and made the decision to build FAM moderation system with Oberon service, which has already provided a set of comprehensive moderation policies to help FAM speed up moderation. Additionally, we finalized requirements and had alignment complaint for the three implementation phases. The initial two phases will be a POC of moderation and periodic manual moderation using catalog data within Oberon. Phase 3 will be major topic in this system design doc, where our efforts will be concentrated on establishing integration between FAM moderation system, Oberon service and Risk Evaluator to process real-time streaming data effectively.

Glossary

- **M&M**: FAM Monitoring and Moderation system.
- **FAM Moderation System**: The service which will build by FAM team to evaluate compliance of extracted object/entity against specific defined policies.
- **Oberon Moderation Service**: The service built by Santos Review team to moderate reviews for specific defined policies.
- **FRE:** FAM Risk Evaluator Service which provides a generic and scalable solution to evaluate and detect different types of risk.
- **INS**: Item Notification Service which owned by Santos catalog team (Bica) to push item related notification to downstream.
- **IAS**: Item Administration Service which owned by Santos catalog team (Bica) which is responsible for product management and is authoritative for Santos product information.

Requirements

In the preceding solution discussion, end feature priority and business requirements have been determined and shared details in Appendix. In this section, we will summarize requirements related FAM moderation system design:

1. Moderation system has inputs as plain text, image URLs, and the FAM S3 folder path to find a list of files. All contents are formatted according to ⚓ FAM Moderation Input Schema
2. Moderation system outputs include the result as moderationResult(APPROVE, REJECT, IN_MANUAL_REVIEW), violationPolices, moderationDetails and moderationSnapshot. The outputs are saved in Santos FAM data lake finally and be used to create paragon case for manual review.
3. Moderation system is developed as a new risk evaluation function, which leverages FRE to link with the Santos domain team or other functions, allowing them to trigger evaluations and take appropriate actions based on the RiskOutcome output within moderation function result.
4. Moderation system provides support for both [p0]asynchronous and [p1]synchronous moderation review processes.
    a. [p0] asynchronous: Evaluate API will trigger moderation task and return IN_PROCESS status. Once we get the moderation results from Oberon, it will asynchronously update risk outcome in FRE.
    b. [p1] synchronous (Depends on P0 behavior):
        i. If the models can generate results within x seconds, we will immediately return the results to the client.
        ii. If the processing time longer than x seconds, we will return the results as IN_REVIEW and send the moderation review result asynchronously.

Out of Scope

- Not cover the part on how we select moderation policy (OFFENSIVE, HATE_SPEECH, BAD_LANGUAGE, SPAM,

SELF_HARM etc) in Oberon service.

- Not cover the part on how we train the Oberon ML model to fit moderation use case.
- Not cover the part on how FAM risk investigators use paragon case to do the manual moderate.

Moderation Function System design - P0

HIGH LEVEL ARCHITECTURE

Source link

Trigger the evaluation (labeled as 1.x): When FRE call evaluate SPI, FAM moderation function triggers moderation by creating moderation tasks in Oberon, and then give FRE response with IN_PROCESS status.
Evaluation Result Update (labeled as 2.x): When Oberon finish the moderation, it will upload moderation outputs in S3 and call FAM to publish results. FAM moderation function returns updated results to FRE. Also, if manual review is required, FAM moderation function will create paragon case for further investigation.

Next sections will give more in-depth insights into the workflow, particularly focusing on data inputs and the sequential processes. Also, the architecture above is for P0 only. P1 will save paragon review results in FAM Next step for Moderation. Besides, we have already covered E2E data flow in M&M high level discussion. Shared in Appendix E2E Moderation Data Flow if you want to know more on how moderation system gets formatted data in catalog and domainURL.

REQUEST INPUTS

We aimed to build a generic moderation function with well-formatted inputs. Referring to the use case, there are two kind of inputs that will activate the Moderation system:

1. [Input 1] A formatted JSON text with one or more contentSets for String and ImageURLs. Catalog data sample.
2. [Input 2] A formatted JSON with a S3 folder path only. The folder includes multiple files and each file contains formatted JSON with one or more contentSets for String and ImageURLs. DomainURL sample.

REQUEST PROCESSING WORKFLOW

For the [Input 1] from catalog, FAM can call Oberon one time with this JSON text to get moderation result directly. However, when dealing with a domain URL moderation request as [Input 2], we need to handle extensive merchant website data in a folder within multiple files. Instead of ingesting an entire folder with huge data in a single request, the review team gives a preference to keep Oberon as a system tailored for singular requests only. Batch processing is a unique use case for FAM which does not align with Oberon current roadmap. More details in concerns to batch in Oberon.

Based on that, FAM is implementing an aggregation layer to call Oberon for individual files. Mapping logic for domain URL is:

- **One FRE call** contains **one folder path**, which includes all contents for one domain URL.
- This domain URL folder includes **N files. Each file is presented as one URL under the domain**. Each file contains a formatted JSON with one or more contentSets for String and ImageURLs. DomainURL sample
- **Each file** will trigger Oberon once as **one moderation task**. **Each task** will moderate whole JSON and generate **one moderation task result**.

The JSON with multiple contentSets in one URL can be really large. So we send Oberon S3 paths instead of extensive contexts in API requests. It reduces the risk of timeouts and improves overall performance with a manageable API payload. Afterward, FAM will consolidate moderation results from all tasks into a unified outcome and update it as the risk outcome in FAM

moderation system. Also, in cases requiring manual review, only one case will be generated with all moderation details in MANUAL/REJECT.

Source link

LOW LEVEL ARCHITECTURE

Source link

LOW LEVEL ARCHITECTURE(NEW - CATALOG)

Catalog Plan Source Link

The components include:

- Core FAM Moderation API service (ALB + Fargate)
  - Implement FAM Domain Service Evaluate SPI to trigger FAM moderation function, which underlying separates them to single record if needed and call Oberon moderation service for each one and create records in dynamo DB tables to track the status.
- DynamoDB tables
  - Moderation Tasks Table to store task level data with moderation result
  - Moderation Assessments Table to save assessment level data with an aggregated moderation result
- SNS-SQS for Oberon team to publish moderation results back.
- Results Aggregator Lambda to update records and aggregate task results and create paragon case if manual review is required.

COMPONENT DETAILS

Evaluate SPI

Moderation Risk Evaluation Function will implement evaluate SPI defined by FRE to handle a risk evaluation request. We will use ALB (Application Load Balancer) and Fargate to create API. Data model for evaluate SPI is shared in Appendix.

Underlying workflow when the SPI is triggered:

1. Preprocess and validate data with FAM moderation schema
   a. When inputs are text contexts, we validate contexts and parse to an Oberon moderation input as a moderation task.
   b. When input is a S3 folder path for a domain URL with multi files, we validate contexts and parse each file as a moderation task to trigger the Oberon.
2. Create a single record in Moderation Assessments Table with risk_outcome_id. total_index is number of tasks and the moderation_results field will be flagged as "IN_PROCESS".
3. For each task:
   a. Call Oberon CreateModerationTask API. Oberon service initiates moderation task and send back the moderation task id.
   b. Create a new record in Moderation Tasks Table with moderation task id.

SNS -SQS

Oberon publish moderation results back to FAM by using SNS endpoint instead of the API call. This decision is driven by the aim

to maintain generalization from FAM within Oberon moderation system, while also ensuring the protection of internal implementation details that could be compromised if Oberon were to directly invoke FAM's API for result publication. More details: Do we need to consume Oberon results by API call, or by SNS endpoint?

Underlying workflow for SNS:

1. Once Oberon upload moderation result as S3 file, it publishs SNS to FAM moderation function. We will **support S3 file output only** keep the consistency to save in moderation results.

Moderation Results Aggregator Lambda

In Lambda, we will batch process data from SQS. This is because each task requires updating the associated request in the assessment table, multiples attempts to override one dynamoDB record can lead to ConditionalCheckFailedException under optimistic lock. By adopting a Batch approach, each lambda execution will exclusively update a single record once to decrease the latency and avoid the retries. Why we need to do the batch?

Underlying workflow for Lambda:

- For each lambda batch
  - Update moderation_results and result_path in task table based on unique task_id.
  - Create a map for all updates
    - Key is risk_outcome_id, value is list of updated tasks.
  - For each risk_outcome_id
    - Get **tasks_completion_status** in assessments table, and update with list of updated tasks
    - Check **tasks_completion_status** with **total_index**.
      - If all tasks for this risk assessment are moderated → check Moderation result aggregation logic below to generate aggregated result. Copy results from Oberon to FAM and create paragon case and send back to FRE.
      - If partially are still in progress → skip
  - Query to get oldest IN_PROCESS request
    - If created time is smaller than x days → skip
    - If created time is longer than x days → generate alarms

Moderation result aggregation logic

Case #	Scenario	has approved tasks	has reject tasks	has manual review tasks	Return final result as	Create Paragon Case
1	All tasks are approved	Yes	No	No	Approve	No
2	Some tasks are marked as reject	Yes	Yes	No	Reject	Yes for p0
3	Some tasks are marked as manual review	Yes	No	Yes	Manual review	Yes
4	All tasks are rejected	No	yes	No	Reject	Yes for p0
5	All tasks are manual review	No	No	yes	Manual review	Yes
6	Some approve; some reject; some manual	Yes	Yes	Yes	Reject	Yes for p0

Workflow sample

- **[Initial Request]:** A request contains a folder with 10 files, each requiring moderation. For each file, a corresponding Oberon task is created to initiate moderation.
- **[Task Table Creation]:** Create a task record in the tasks table for each task_id.

- **[Assessment Record Initialization]:** After all tasks are created, create a record in the assessments table with risk_outcome_id = 123.
    - Update the assessments table with total_index_num = 10 and tasks_completion_status = 0 (hexadecimal: 0000000000).
- **[Moderation and Initial Updates]:** Oberon returns moderation results and updates the tasks table.
    - SNS-SQS triggers a Lambda function with a batch of records from the tasks table.
- **[First Batch Processing]:** In the Lambda batch, process moderated results for task_index values <0, 1, 2, 6, 7>.
    - Update tasks_completetion_status to c7 (hexadecimal: 0011000111).
    - As tasks_completetion_status is not 3ff (hexadecimal: 1111111111), skip further processing (calculated by total_index_num).
- **[Second Batch Processing]:** In the subsequent batch, process moderated results for task_index values <3, 4, 5, 8, 9>.
    - Update tasks_completetion_status to 3ff (hexadecimal: 1111111111).
    - All tasks for this risk assessment are now moderated.
- **[Final Assessment and Result Determination]:** Check the tasks table with risk_outcome_id = 123.
    - If 8 tasks are marked as APPROVE and 2 tasks are marked as MANUAL:
        - Return the final result as MANUAL and create a paragon case.
        - Update the assessment table with final moderation_status as complete and moderation_results as MANUAL.

DynamoDB Database

Moderation Tasks Table
Name: ModerationTask
partitionKey: task_id(string)
sortKey: risk_outcome_with_index

task_id	risk_outcome_with_index	risk_outcome_id	task_index	moderation_input_type	moderation_status	moderation_results	result_path	created_at	updated_at	expired_date	version
<UUID>	<UUID>_number	<UUID>	Number	CATALOG	IN_PROCESS	REJECT	String of S3 path	epoch	epoch	epoch	1

GSI - GetByRiskOutcomeId
PartitionKey : risk_outcome_id
RangeKey: task_index
Use case: To find all tasks under one risk assessment request and use **moderation_results** for each to aggregate a result for this risk assessment.

Moderation Risk Outcome Table
Name: ModerationRiskOutcome
PartitionKey: risk_outcome_id

risk_outcome_id	risk_evaluator_id	moderation_input_type	moderation_status	aggregated_moderation_results	tasks_completetion_status	paragon_case_id	total_index	created_at	updated_at	expired_date	version
<UUID>	<UUID>	CATALOG	IN_PROCESS	REJECT	hexadecimal string	string	Number	epoch	epoch	epoch	1

GSI - GetByStatus
PartitionKey : moderation_status
RangeKey: created_at
Use case: to find oldest IN_PROCESS request and generate alarm.

Behavior with Catalog(11/7)

CATALOG WORKFLOW BEHAVIOR

- For each owner, we have one catalog with multiple items. Any updates in one item will trigger an update event.
- If item updates don't have any changes on [catalog title/image url/description], we will skip the moderation.
- New item or item updates on [catalog title, image url, description] will trigger moderation **once or twice**:
    - If there is a new catalog or catalog updates on [catalog title], we will trigger one moderation execution with brand name list to detect brands counterfeit for FOLEX. So if Oberon returns MANUAL, we will create one paragon case as counterfeit moderation paragon case
    - If there is a new catalog or catalog updates on [catalog title, image url, description], we will trigger another one moderation execution to detect any non-compliant info in catalog. If Oberon returns REJECT/MANUAL, we will create one paragon case as **moderation paragon case**
- For MANUAL case, once investigators resolved Paragon case, investigators can add final results(Reject or Approve) in an excel doc(Pending for creation). We can use the data in excel to train the ML model in Oberon, which gives us more accurate results in the future.

Based on the workflow, we have scenarios below:

1. Create multiple paragon cases for one owner.
    a. One catalog will have multiple items. We trigger moderation execution on **item-level**.
    b. For each item event, we may create up to 2 different paragon cases, when 1) brand name is existed in the title 2) and non-compliant info are existed in description/title/imageUrl.
2. If merchant updates the same catalog item which we already moderated, we will trigger another moderation within new paragon case.

Paragon Contents

- **Primary Email:** <Merchant Email>
- **SubjectText:** (Need to update)Possible <StationType> Non-Compliance Data for BwP AUP <DataType> moderation - ownerId: <OwnerId>
- **AnnotationText:**
    - Moderation TaskId:
    - Moderation Policies:
    - Moderation Content Set:
    - Moderation History:
    - Catalog Id:
    - Item Id:
    - Catalog Version:
    - SKU:

PARAGON EXAMPLE

Example 1 - Counterfeit
Station type: Counterfeit
Data type: Catalog

SubjectText: Possible Counterfeit Non-Compliance data for FOLEX AUP Catalog moderation - ownerId: <OwnerId>

AnnotationText:

- Moderation TaskId: UUID
- Moderation Policies: `CUSTOMER_OVERRIDE`
- Moderation Content Set: `ModerationContentSet(ContentSet=[ContentObject(Name=text_title, Value=ContentObjectValue(StringValue=APPLE Cotton Solid Fabric Face Mask Reusable Nose Clip Filter Pocket Cloth Face Mask | Pink Butterfly 3Pak))`
- Moderation History: `[ModerationHistory(ModerationStatus=MANUAL, ModerationDescription=Customized reject phrase: [Apple] matched with pattern: [\bAPPLE\b], ModerationTime=2023-11-03T00:19:43Z, Policies=[CUSTOMER_OVERRIDE], Workflow=TEXT)]`
- Catalog Id: ect1dk32zgjby3
- Item Id:8h3t4100gigh1x
- Catalog Version: 1699044549451951000
- SKU: sku-12345
- ~~Merchant id?~~

Example 2 - Content
Content Moderation paragon case
Station type: Content
Data type: Catalog
SubjectText: Possible Contents Non-Compliance data for BwP AUP Catalog moderation - ownerId: <OwnerId>
AnnotationText:

- Moderation TaskId: UUID
- Moderation Policies: `SEXUAL_ACTIVITY`
- Moderation Content Set: `ModerationContentSet(ContentSet=[ContentObject(Name=image-0, Value=ContentObjectValue(MediaValue=MediaValue(SourceUrl=https://amazon-omni-cdn.com/visz97be6o1/1unufzqny4w620/0590-Lifestyle.jpeg, Size=0)))])`
- Moderation History: `[ModerationHistory(ModerationStatus=REJECT, ModerationDescription=RekognitionLabel: Sexual Activity Confidence: 92.8678%. RekognitionLabel: Explicit Nudity Confidence: 92.8678%. , ModerationTime=2023-11-03T22:53:41Z, Policies=[SEXUAL_ACTIVITY], Workflow=IMAGE, ContentId=https://amazon-omni-cdn.com/visz97be6o1/1unufzqny4w620/0590-Lifestyle.jpeg)]`
- Catalog Id: ect1dk32zgjby3
- Item Id:8h3t4100gigh1x
- Catalog Version: 1699044549451951000
- ?SKU:

Oberon Integration

CHANGES IN OBERON SERVICE - P0

Part 0 - Station Setup and Policies

Create one moderation station for FAM risk evaluation id in Prod

1. For FAM, each risk evaluation id should have it's own station id. In P0, We will only create one station id based on FAM's use case. FAM will be the consumer to create tasks. Why we need one station id, not station per each ownerId? Security concerns?
2. Create a policy with list for brand name for Manual review in station

Part 1 - CreateModerationTask API

CreateModerationTask API call can support:

1. **Handle multi images processing:** ModerationContentSet can be a Json with contentSets as multiple string and multiple image url. Catalog data sample
2. **[DomainURL Launch] Handle document processing:** ModerationContentSet can be a Json with one document as a file, which includes contentSets as multiple string and multiple image urls. DomainURL sample
3. **[DomainURL Launch] Handle high TPS with API throttling:** API can handle the high TPS.
4. ~~**Add moderationTaskType:** Add new field moderationTaskType in the request to help us disable the manual review(Or we need to make it configurable in station creation. No Need In Oberon)~~

Part 2 - Publish ModerationTask Results

ModerationTask results publishing can support:

1. Enable SNS endpoint for Approval/Reject/Manual with station id and task id
2. Create results based on each context set.
3. ~~Have a S3 moderation bucket to save moderation results. FAM can have permissions to read the bucket to get the data. (Life cycle)~~
4. ~~Save whole moderationTaskObject(moderationContentSet snapshot and moderationHistories) as a S3 file in S3 moderation bucket, and call FAM SNS endpoint with moderationStatus, policies and S3 file path for overall result.~~

Part 3 - Mantainace

1. Over SLA alarms:
    a. Oberon has the alarms itself when over SLA for tasks.
    b. Provide CTI to FAM that we can auto-cut ticket to Oberon when task over the SLA.

CHANGES IN OBERON SERVICE - P1/P2

1. [P1] FAM can create some new stations for other customized policies in Oberon.
2. [P1] Update model for image rejection threshold for Reject and Manual
3. [P1] FAM can add new policies for allowlist, denylist and manual review list in Oberon.
4. [P2] FAM can moderate content in the form of videos with URLs.
5. [P2] [FAM & Oberon] Feed case results back to ML training so that the ML models can learn and get better.
6. [P2] [FAM & Oberon] Moderate brand logo/trademark image for FOLEX counterfeit brands. - new models in Oberon side

Feel free to leave the comments for any Oberon tasks which we didn't cover above.

Estimation Timeline with milestones

FAM Moderation System Execution Plan

Naming

FAM

SECURITY

Authentication/Authorization

VENDOR CODE

fam-moderation

SERVICE PRINCIPALS

We will use external service principals, as recommended by Identity. (AWS auth interval vs external FAQ). In addition we will use global service principals (versus regional), also as recommended.

Stage	Service Principal	Type
alpha	alpha.fam-moderation.amazonaws.com	External
beta	beta.fam-moderation.amazonaws.com	External
gamma	gamma.fam-moderation.amazonaws.com	External
prod	prod.fam-moderation.amazonaws.com	External

API's and Actions

Resource	API	Description	Access Level	Authorized Callers
FAMModeration	fam-moderation:EvaluateRisk	create	write	ExecutionRole

- We will use Direct Coral Auth integration for authZ
- The service will onboard to SantosJade and use execution roles for evaluate API.

Next step for Moderation

MODERATION P1

Requirements

1. [Oberon] Create new station for customized policies - Oberon side
2. [Oberon] New policies for allowlist, denylist and manual review list - Oberon side
3. [FAM] Customize paragon contents and send paragon results back
4. [FAM] If the processing time longer than x seconds, we will return the results as IN_REVIEW and send the moderation review result asynchronously

Paragon Results Integration (Need to update if we choose SNS publisher)

After completing each case operation, MCMSV2 uses an asynchronous thread to publish operation related information to a SNS topic. Topic contains generic information like caseId, tenantId and operation, and API information like input and output value. If this API request modifies attributes of the case, SNS information will also contain previous/new state of the case to demonstrate the change. Link Also, we can do the backfill by using API to get paragon case status with case id.

Source link

MODERATION P2

Requirements

1. [Oberon] Moderate content in the form of videos with URLs.
2. [FAM & Oberon] Moderate brand logo/trademark image for FOLEX counterfeit brands. - new models in Oberon side
3. [FAM & Oberon] Feed case results back to ML training so that the ML models can learn and get better. - Oberon side to connect with datalake
4. [FAM ops team?] (need to check) As a Merchant, I want to be informed why my account or listing is ineligible for BwP due to a content violation and to be able to appeal or take corrective action to get my account or listing reinstated.

Whys

WHY WE SEND S3 FILE PATH, NOT ENTIRE FOLDER TO OBERON

We have two options to handle a request with S3 folder path:

- [Option 1] FAM handle the multi files aggregation and call Oberon with **single S3 file**
    - Pros:
        - Requires only minor changes in Oberon to enable and moderation of content within a single file.
        - Each file triggers a single call to Oberon, simplifying moderation task creation.
    - Cons:
        - FAM moderation system needs to handle data aggregation.
        - FAM initiates separate calls to Oberon for moderation of each individual file. Oberon needs to handle high TPS API call to create moderation tasks effectively.
- [Option 2] FAM call Oberon with entire folder path and Oberon handle the **multi files** aggregation

Review team worked with us to explore the possibility of processing batches in Oberon: Firstly, processing all files **sequentially** will have high latency with out of memory issue. On the other side, to moderate whole files **in parallel**, a potential solution was to set up an SQS queue to buffer and process an entire folder's files. Once the queue size reaches 0, Oberon can aggregate and deliver results. However, this approach lacks scalability, assuming a single queue for processing one request; Another option involves shifting all aggregation logic in this design from FAM to Oberon. This would require Oberon to create a new table to handle batched data and establish a new lambda function for processing purposes.

    - Pros:
        - No additional data aggregation required in FAM's moderation function.
        - Oberon gains the capability to handle batched requests for other use cases.
    - Cons:
        - The structure of the Oberon service has been established. A significant shift are required additional efforts no matter which solutions we choose.
        - Oberon's result format (single file or folder) necessitates FAM to interpret files for moderation results and

snapshot extraction in order to create a paragon case.

Batch processing is a unique use case in FAM which does not align with Oberon current roadmap. The review team expresses a preference to build Oberon as a moderation system tailored for singular requests, and introduce a fam connection layer to aggregate the results outside Oberon. In the implementation phase, they will also provide support to FAM for handling high TPS in API calls.

DO WE NEED TO CONSUME OBERON RESULTS BY API CALL, OR BY SNS ENDPOINT?

We have two options to get moderation results from Oberon:

- [Option 1] By SNS endpoint - create SNS-SQS-Lambda workflow to process results(Recommended)
    - Pros:
        - SNS-SQS-lambda design loose coupling between FAM and Oberon, which enhances scalability and maintainability for each other.
        - The event-driven architectures can help FAM easily re-drive failures.
        - SNS-SQS-lambda can control the speed(concurrency) and batch process tasks status updating
    - Cons:
        - FAM needs to contact review team manually when we deal with client errors like S3 files is not reachable or not readable.
        - Pub-Sub is well-suited for scenarios where there are multiple subscribers interested in the same event. However, benefits aren't fully utilized since FAM is the only SNS topic consumer.
        - The process on the compute side involves tasks like updating items, verifying results, and creating paragon cases. Due to limitations posed by Lambda, it's necessary to employ extra component like step functions to handle this workflow.

Source link

- [Option 2] By updateModerationResults API - create new API to update moderation results

Implement PublishFAMModTaskResult API for Oberon team to update moderation results back so that we could update records in our Dynamo DB table.

    - Pros:
        - API calls provide synchronous communication, aligning well with scenarios where FAM can provide immediate feedback about S3 file consumption. In cases where the S3 file is unreachable or its contents are unreadable, such situations will be treated as client errors, prompting exception handling.
        - The connection between requests and responses ensures clear debugging flow and reliable delivery.
        - Separation for the API and subsequent processes:
            - The API handles file validation and DynamoDB table updates.
            - The compute side oversees tasks such as risk assessment verification, paragon creation, and invoking FRE for risk outcome updates.
    - Cons:
        - Oberon necessitates the addition of new logic to manage failures and retries.
        - During periods of high traffic, the synchronous nature of communication can result in bottlenecks or reduced performance due to a large number of requests waiting for responses.
        - Expose internal API outside FAM.

Oberon provides the S3 path for moderation results back to FAM by using PublishFAMModTaskResult API. Data model

for PublishFAMModTaskResult are shared in Appendix. Considering our reliance on S3 files as inputs rather than direct text content, an API call with response or client-side exception can offer immediate feedback to Oberon to retry or handle the failures, which also effectively reducing manual operational efforts for both teams after launch. That's the reason we choose Request-Response pattern instead of using Publish-Subscribe pattern to transmit moderation results via SNS in Oberon. Underlying workflow for API:

1. Once Oberon upload moderation result as S3 file, it calls PublishFAMModTaskResult to ask FAM moderation function to process the results. We will **support S3 file output only** keep the consistency to save in moderation results.
2. FAM moderation function will update moderation_results and result_path in task table based on unique task_id.

Source link

To conclude, Oberon facilitates both delivery methods for moderation results. An API call response or client-side exception can offer immediate feedback to Oberon for retry or failure handling. But using SNS decision is driven by the aim to maintain generalization from FAM within Oberon moderation system, while also ensuring the protection of internal implementation details that could be compromised if Oberon were to directly invoke FAM's API for result publication.

WHY PROCESS STREAMING WITH DYNAMODB STREAM + LAMBDA, NOT KINESIS STREAM + FARGATE? (IF WE USE API)

Our choice of DynamoDB Stream with Lambda is rooted in its smooth integration and our decision to implement batch processing to address potential bottlenecks. Why Process streaming with dynamoDB stream + lambda, not Kinesis stream + Fargate? (If we use API) Underlying workflow when the Lambda is triggered:

- [Option 1] Kinesis Streams with Fargate
  - Pros:
    - Scalability: Kinesis Streams can handle large volumes of data and support multiple consumers simultaneously.
    - Parallel Processing: Multiple Fargate tasks can be run in parallel, further improving processing efficiency.
  - Cons
    - Increased Complexity: Setting up and managing Fargate containers can require more operational effort compared to the fully managed Lambda service. Also they involve more components, leading to a potentially more complex deployment and monitoring setup.
    - Higher Cost: Considering FAM's use case of intermittent request spikes rather than continuous processing, the utilization of Fargate becomes less economical due to its requirement for dedicated resources, potentially leading to elevated costs when juxtaposed with the cost-effective serverless Lambda model.
- [Option 2] DynamoDB Streams with lambda (Recommended)
  - Pros:
    - Tightly Integrated: DynamoDB Stream and Lambda are designed to work seamlessly together, making integration straightforward.
    - Real-time Processing: DynamoDB Streams offer real-time event notifications for changes in the database, allowing quick response to data updates.
    - Built-in Retry Mechanism: DynamoDB Streams provide built-in retry mechanisms for handling processing failures.
  - Cons
    - Synchronous Behavior: While Lambda can process events quickly, it's still synchronous in nature, which could lead to bottlenecks during high-traffic situations. (We decide to use batch)

- Limited for Stream Pull and Processing Time:
  - Lambda functions have a maximum execution time (15 minutes_
  - AWS Lambda service polls the stream for new records four times per second. And the number of records to send to the function in each batch, up to 10,000.
  - Lambda passes all of the records in the batch to the function in a single call, as long as the total size of the events doesn't exceed the payload limit for synchronous invocation (6 MB).

Our choice of DynamoDB Stream with Lambda is rooted in its smooth integration and our decision to implement batch processing to address potential bottlenecks highlighted in the disadvantages. Simultaneously, we acknowledge that Kinesis Streams with Fargate presents heightened complexity and increased costs, rendering it less conducive to our project needs.

WHY WE NEED TO DO THE BATCH?

- **Eliminate Retries:** In the scenario where each task requires updating the associated request in the assessment table, simultaneous attempts to override a record within a single lambda execution can lead to ConditionalCheckFailedException under optimistic lock, necessitating retries. By adopting a Batch approach, each lambda execution will exclusively update a single record once, circumventing such issues.
- **Reduced Latency:** Batch processing can decrease the overall processing time compared to handling items individually, potentially improving response times. The estimates for concurrent updates is high as shown below. With optimistic lock, it will be even worse to update one record multiple times.
  - 10 updates: ~1.5s
  - 100 updates: ~2s
  - 1000 updates: ~10-20s
- **Scalability:** Batch processing can help with scalability by processing a larger volume of items efficiently, aligning well with Lambda's auto-scaling capabilities.

WHY USE ALB?

The current official “golden rule” guidance is to rely on ALB: https://builderhub.corp.amazon.com/docs/native-aws/developer-guide/golden-path-ecs.html for ECS services. You can use a NLB and have an ALB target if you wish to use NLB-only features like terminating PrivateLink connections.

SHOULD WE USE LAMBDA OR FARGATE WHEN CONNECT WITH SQS?

	Lambda	Fargate
Event Trigger	Set up the trigger with batch size and batch window	Use long poll with setMaxNumberOfMessages and ReceiveMessageWaitTimeSeconds
Management	handling scaling, patching, and resource provisioning automatically	Need to define container resources (CPU, memory) and networking settings. Can use Auto Scaling to adjust the number of tasks running based on CPU or memory usage.
Cost	Billed based on the number of requests. Easy to handle spike request which is perfect for our use cases.	Billed based on vCPU and memory
Execution time	Default execution timeout of 15 minutes	Suitable for longer-running tasks without limitations
Redrive	Connect DLQ with lambda with 1-click redrive	Fargate to poll DLQ message

HOW DO WE AGGREGATE DATA: CHECK DYNAMODB OR USE STEPFUNCTION MAP?

	Check DynamoDb	StepFunction
Event Trigger	Trigger with batch size and batch window	When a new request flow in, we will create a new StepFunction execution. Then when Oberon service returns results, we will use lambda to trigger the tasks in Stepfunction map until all tasks finished.
Aggregate Behavior	Each batched event updates results and triggers aggregation.	Waiting to aggregate until all Oberon results are ready
Enginnering Work	+++ Need to add logic to check DynamoDb and update records	++ Small SDE efforts to setup stepfunction
Cost	+	+++ Calculated with number of requests and duration.
Limitations	Depends on the batch size and concurrency. No hard limits.	The `Map` state runs each iteration as a child workflow execution, which enables high concurrency of up to 10,000 parallel child workflow executions. **But maximum number of open Map Runs is 1000 as the hard quota.**
OE Management	Oberon re-send the message, or redirve DLQ message	Hard to re-drive: If stepfunction execution failed/timeouted and needs to retrigger, we may either send all moderation task to Oberon again, or use extra function to check completed tasks. More extra efforts in OE maintainance.
OverSLA Check	Set up cloudwatch to have periodically check in DynamoDb with In-progress requests	Set up timeout for step function.

Appendix

END FEATURE PRIORITY

The FAM moderation system will build the following features with priority from high to low:

1. **[Feature 1] Moderate Santos catalog data for LimaPDP and Hosted Widget Post Order Pages:** Santos catalog team will send info to FRE with the catalog data, which includes Lima PDP and hosted Widget Post Order Pages like BwP Checkout Page, Order Confirmation Page, Order Tracking Page. Moderation needs to detect any non-compliant info in catalog.
2. **[Feature 2] Moderate FOLEX listings for Automated Brand Protection and Brand Gating controls:** If content (image or text) that is a trademark or brand logo of a well known brand. Moderation needs to check these results with list of allow listed brands, then human investigator will decide if it is a counterfeit brand. (Extra efforts are needed because current existed ML models didn't cover this use case)
3. **[Feature 3] Moderate merchant declared URLs across all business products:** All merchants onboarded with BwP products need to comply BwP AUP, so after continuously monitoring merchant websites, moderation needs to detect any non-compliant contents.

BUSINESS REQUIREMENTS

1. [P0] As a moderation system user, I can moderate content in the form of texts and images for product attributes level.
2. [P0] As a moderation system user, I can moderate text content for top 20 highest risk brands in FOLEX.
3. [P0] As a moderation system user, I can utilize default policies to trigger the moderation function.
4. [P0] As a moderation system user, I can receive moderation results asynchronously, with confidence scores and other analyzes output from the moderation model and an outcome label as APPROVE, REJECT, or MANUAL_REVIEW.
5. [P0] As a user who initializes a moderation function, I should be able to configure confidence score thresholds to determine the moderation results.
    a. If the confidence score of one content is above the threshold, the outcome should be labeled as the result from moderation model: APPROVE means the content adheres to the moderation guidelines; REJECT indicates that it is not qualified.
    b. Conversely, if a content achieves a confidence score lower than the threshold, with ambiguous/uncertain, the result

should be labeled as MANUAL_REVIEW.

6. **[New] [p0] As Santos FAM Ops Team, I want to be able to overwrite a Moderation decision in case the wrong decision was initially made.**
7. [P0] As a user of the moderation system, I can manually investigate in Paragon whenever the moderation model generates MANUAL_REVIEW results. The paragon case includes corresponding confidence scores and an analysis of the output produced by the moderation model. Each moderation request will only generate one paragon case. For example, the moderation request for a specific domain or merchant, we will generate one paragon case with a list of URLs which marked as MANUAL_REVIEW.
8. **[New] [p0] As a user of the moderation system, I can investigate in Paragon with snapshot on the section which triggers the manual review check.**
9. [P1] As a moderation system user, I can select multiple policies from a predefined list of BwP policies.
10. [P1] As a user who initializes a moderation function, I can configure the elements displayed in the paragon case, including the ability to change the title and customize paragon contents to align with my use case.
11. [P1] As a FAM moderation investigators, I can assign approval or rejection when resolving paragon cases. My investigation decisions will send back to the system saved as the final moderation results.
12. [P1] As a user who initializes a moderation function, I can use a new policy with an allowlist, denylist and manual review list to approve or reject content based on specific keywords, as a straightforward content moderation based on keyword criteria.
13. [P2] As a moderation system user, I can moderate content in the form of videos with URLs.
14. [P2] As a moderation system user, I can moderate brand logo/trademark image for FOLEX counterfeit brands.
15. [P2] As a FAM moderation investigators, I should feed case results back to ML training so that the ML models can learn and get better.

E2E Moderation Data Flow

Based on feature requirements, here are two data flows in FAM moderation system design:

1. **Data flow to moderate catalog data:** Using catalog streaming data to moderate newly created/updated catalog records.
2. **Data flow to moderate merchant URLs:** Using merchant websites data sourced from monitoring system to moderate any non-compliant information present within them.

DATA FLOW TO MODERATE CATALOG DATA

Source: link

The catalog provides an established method for accessing streaming data. INS will generate SNS notifications (samples) for newly updated catalog entries, and IAS offers an API to retrieve comprehensive catalog data using information from INS messages. Despite the minimal preprocessing effort required for adding the IAS call to the moderation system, we have chosen to prioritize creating a New Catalog Connector before EventListener. Because Moderation Risk Evaluation Function should follow the generic moderation nature to process properly structured data. Despite the extra development effort needed for the new connector service, we have opted for this approach to avoid introducing a dependency between moderation and IAS calls.

The data flow is:

1. FAMCatalogConnector subscribes Catalog INS merchandise **update-events** SNS topic.
2. Once we receive notification messages which updated attributes contain `it:santos/product:title`, `it:santos/product:images`, `it:santos/product:description`, FAMCatalogConnector will trigger an IAS call to extract `title`, `images`, `description`. Subsequently, this extracted data will be parsed into the moderation

input schema and forwarded to the EventListener.

3. EventListener triggers FRE and then call evaluate SPI.
4. Moderation Risk Evaluation Function will evaluate catalog events and engage the Oberon service for moderate title, images, description.
5. The moderation results will be saved in FRE as risk outcome results.

Why we create catalog handler before EventListener

DATA FLOW TO MODERATION MERCHANT URLS

Source link

When merchants finish BwP onboarding steps, the domain URL list is sent to FRE to monitor first and then trigger moderation. Then Moderation Risk Evaluation Function will be part of a composite AUP Risk Evaluation Functions. The data flow is:

1. EventListener gathers domainUrl events from Tamarama and trigger FRE.
2. FRE calls BwP AUP Risk Evaluation Function to evaluate with Domain URL. Then AUP Function triggers monitoring system to crawl data from merchant website.
3. With the completion of monitoring, BwP AUP Risk Evaluation Function will initiate Moderation Risk Evaluation Function using evaluate SPI to moderate data from merchant websites. This data will comprise a collection of appropriately structured moderation inputs sourced from the monitoring data storage.
4. Moderation Risk Evaluation Function returns the moderation results through updateRiskOutcome? API call.
5. When the necessity arises to moderate the same domain URL once more, AUP risk evaluation function will generate a new moderation request rather than reusing a prior request for the same domain URL.

FAM MODERATION INPUT SCHEMA

The standardized format moderation requires before triggering the evaluation.

```
@documentation("The instance represents the set of moderation content.")
structure moderationContentSet {
    moderationInputs: moderationInputs
    version: String/Integraiont - newly added to handle out of order event
}

list moderationInputs {
    type: String,
    key: String,
    value: String
}

eg:

{
  'moderationInput': [
    {
      'type': 'text',
      'key': 'brandName',
      'value': 'FAM brand',
    },
    {
```

```
      'type': 'text',
      'key': 'productTile',
      'value': 'FAM T-shirt',
    },
    {
      'type': 'image',
      'key': 'productPic',
      'value': 'https://xxxx.jpg',
    },
  ]

}
```

Catalog input Samples to Oberon

```
  {"contentSet":
    [
      {"name": "title",
       "value": {
        "stringValue": "Wyze Cam v3"
        }
      },
      {"name": "description",
       "value": {
         "stringValue": "Our award-winning wired smart camera is back with illuminatir
        }
      },
      {"name": "image",
       "value": {
         "mediaValue":
          {"sourceUrl": "https://amazon-omni-cdn.com/drjh4s7l6mg/nbibtva5tgfsd5/31RrWM
           "mimeType": "jpeg",
           "extension": "jpeg",
           "size": 256}
        }
      }
    ]
  }
```

DomainUrl input Samples to Oberon

S3 folder: `s3://oberon_moderation_results/aup_domain_urls/www.wyze.com/version=1/`

```
{
  "contentSet":
    [
      {"name": "url",
       "value": {
        "documentValue": "s3://oberon_moderation_context/aup_domain_urls/www.wyze.com/
        }
      }
    ]
}
```

```
data samples in related_selling_plan=41618559008930.json

{"contentSet":
    [
      {"name": "brandName",
       "value": {
        "stringValue": "Wyze"
        }
      },
      {"name": "productTile",
       "value": {
         "stringValue": "Wyze Cam v3"
        }
      },
      {"name": "productPic",
       "value": {
         "mediaValue":
          {"sourceUrl": "https://amazon-omni-cdn.com/drjh4s7l6mg/nbibtva5tgfsd5/31RrWM
           "mimeType": "jpeg",
           "extension": "jpeg",
           "size": 256}
        }
      }
   ]
}
```

DATA MODEL

Evaluate SPI

```
@http(uri: "/risk_evaluator/<riskEvaluatorId>/risk_assessment", method: "POST", code:
operation: EvaluateRisk {
   input: EvaluateRiskRequest,
   output: EvaluateRiskResponse,
   errors: [
        BadRequestException,
        InternalServiceException,
        ThrottlingException
   ]
}

structure EvaluateRiskRequest {
    @required
    @httpLabel
    riskOutcomeId: String,
    @required
    inputSignalSet: JSON-schema
}

structre EvaluateRiskResponse {
    @required
    riskAssessment: RiskAssessment
}
```

```
structure RiskAssessment {
    @required
    riskAssessmentId: String,
    @required
    riskEvaluatorId: String,
    @required
    riskEvaluatorFunctionId: String,
    @required
    runtime: epochTime,
    assessment: String,
    @required
    riskOutcome: RiskOutcome,
    @required
    inputSignalSet: JSON-schema
}

structure RiskOutcome {
    @required
    riskOutcomeId: String,
    @optional
    runStatus: String // STARTED, NOT_STARTED, RUNNING, SUCCEEDED, FAILED
    @required
    outcome: String, // Status defined by Function owner
    riskScore: Double,
    confidenceLevel: String,
    riskDetails: JSON // Schema will be defined by function owner.
}

// The POJO for ModerationFunction InputSignalSet.
public class ModerationInputSignalSet {
    @NonNull
    private Enum moderationInputType; //CATALOG, DOMAIN_URL
    @NonNull
    private String moderationInputId; //catalogid-itemId(UUID), domainUrl(self-generat
    private String moderationInputVersion; //catalog version, domainUrl monitoring ver
    private String ownerId;
    private MAP<String, String> InputMap//catalogid-itemId(UUID), domainUrl(self-gener
    @NonNull
    private ModerationContentSet contentSet; //ModerationContentSet Json string
    private ConfigOverrides configOverrides; // Override moderation configs
}

public class ConfigOverrides {
    private boolean paragonEnabled; //Default as true
    private string stationId; //Default as FAM created station id
    ...
}

// The POJO for ModerationFunction RiskDetails.
public class ModerationRiskDetails {
    @NonNull
    private Enum moderationStatus; //IN_PROGRESS, PENDING_MANUAL_REVIEW, COMPLETE
    private Enum moderationResult; //APPROVE, REJECT, MANUAL_REVIEW, MANUAL_APPROVE(Pi
    private String moderationHistories;//Moderation histories, S3 for Moderation histo
```

```
}
```

PublishFAMMResults API (If we use API solution)

```
@http(uri: "<base_uri>/PublishFAMModResult/{moderationTaskId}", method: "GET": code: 2
operation PublishFAMModResult {
   input: PublishFAMModResultRequest,
   output: PublishFAMModResultResponse,
   errors: [
        BadRequestException,
        InternalServiceException,
        NotFoundException,
        ThrottlingException,
   ]
}

structure PublishFAMModResultRequest {
    @required
    moderationTaskId: String,
    @required
    moderationStatus: String,
    @required
    moderationStationId: String,
    @required
    moderationResultPath: String //S3 file path which save moderationResult.
    moderationTaskObject: ModerationTaskObject
}


structure PublishFAMModResultResponse {
    @required
    publishStatusCode: String
    publishStatusDescription: String
}

structure ModerationTaskObject {
    @documentation("The identifier referenced by moderation station")
    @required
    moderationStationId: IdString,

    @documentation("The identifier referenced by moderation task")
    @required
    moderationTaskId: IdString,

    @documentation("Contents to be moderated")
    @required
    moderationContentSet: ModerationContentSet,

    @documentation("Metadata of the product to help with the moderation.")
    moderationMetadata: ModerationMetadata,

    @documentation("The external ID for this moderation task from client.")
    externalId: IdString,
```

```
    @documentation("Epoch time at which the review was submitted")
    @timestampFormat("epoch-seconds")
    @required
    receivedTime: Timestamp,

    @documentation("Represent the latest moderation status")
    @required
    moderationStatus: ModerationStatus,

    @documentation("triggered policies corresponding to the moderation decision")
    @required
    policies: Policies,

    @documentation("Record the moderation history for a given review")
    @required
    moderationHistories: ModerationHistories
}
```

Notes

1. SLA: Oberon is planing to have the load tests results.
    a. x for Oberon moderation. Need to know the limitation and bottleneck from Oberon. Pending load tests results.
    b. x business days for investigator manual review.
2. Do we need to return results if it's reject/manual as soon as possible ?
3. What's the manual moderation process?
    a. Once FAM found the non-compliant product after moderation, we will give 7 days grace period for them to update the catalog. If they update in 7 days and pass the moderation check, we will close the paragon cases; if no response from merchant, we will suspend merchant account.
    b. Once suspended merchants update the catalog, FAM will revisit this case and reinstate the account for them.
    c. When FAM suspend merchant account after moderation, Lima PDP and Hosted Widget Post Order Pages will not be set up. Once merchant update the catalog, we will check again.
4. Repeatedly create paragon case
    a. Scenario 1: A paragon case is created for investigator for a specific Domain URL. In the next round moderation, it will create new paragon case for this Domain URL if it still trigger Reject or manual.

Meeting agenda - Aug 29th

Please take a look for the top of the doc.

Meeting agenda - Aug 24th

- 30 mins doc reading
    - Comments reply
    - Component Details
    - Why we send S3 file path, not entire folder to Oberon
    - Do we need to consume Oberon results by API call, or by SNS endpoint?
- Hot debated topics

    - Batch process in FAM or in Oberon
    - Consume moderation results by SNS or API
- AIs
    - Offline sync with team members
    - How to have a intermediate state to move fast

Meeting agenda - Aug 22nd

- Doc reading 30mins(Stop at **Changes in Oberon Service - P0** section)
- Hot debated topics
    - Batch process in FAM or in Oberon
    - consume moderation results by SNS or API
- Go over the comments
    - DTC PDP cover in EverC check, Park - > Lima PDP follow up for EverC
- AI
    - Bucket temp storage - Oberon set up a moderation storage account to save. And permission will be set to limit without bucket sniping in the future.
        - P0: Oberon database S3
        - P1: How to make it available in FAM datalake: **need more discussion.**
    - We need to use something else like ownerId in path to replace URL for security concerns.
    - configurable for sync/ async: latency.
    - configurable to create paragon for manual only or manual and reject.
        - Manual from Oberon: Investigator can check it's Reject/Approval
        - (Optional) Take action items based on Reject/Manual. The action item can be handled in client side, or can handled by FAM Investigator directly. Can configurable. ~~Reject from Oberon: Investigator can identify it's true *negative or false negative*~~
    - Compare Fargate with Lambda to process SQS message Should we use Lambda or Fargate when connect with SQS?

From Akanksha (Synced offline)

1. **What if a task is completed by oberon and aggregator lambda is called to update before assessment record is created?**
2. Have we thought about event bridge + kinesis stream?
3. What is the meaning of expired_date? What is the logic for this field?
4. **Should we define TPS/scale?** Q: What's the estimation for file number from Monitoring?（Aniruddh - PDP） So we could ask Oberon for TPS limits.
5. What does this moderationMetadata?
6. Where is this API defined? PublishFAMModResult
7. Is this the only change required in Oberon? I believe they do have this API but we need modifications like support of multiple images, support high TPS etc?
8. We need to use something else like ownerId in path to replace URL for security concerns.
    a. But ownerId has multi domainURL. May need to be encrypted domainURL or get the exceptions(Follow up with

Appsec)

9. Why not get catalog data from Santalytics

From Pavan (Synced offline)

1. ~~What is the preferred compute solution?~~ Will start from Lambda
2. ~~Will there be another primary key for this table that is generated by our system? Using GUID generated by other systems as our primary index can cause issues that are not known now.~~
3. Will FAM make API call to oberon after receiving the SNS event or will it directly read output from S3 path mentioned in the event?
4. ~~Have we considered how synchronous flow will work with S3 path and the SLA for such call~~
5. How we copy from Oberon to FAM - Data lake
6. Risk evaluator changes

From Aniruddh (Synced offline)

1. When can we get the brand name lists and can we add right now in phase 1 or phase 2
    a. 9/6 - We can get more details
2. Now we only have Reject/Approval in moderation. When we need this feature for image threshold? P0 or P1? - P1
3. Do we really need sync procedure based on our use case? If real-time is not needed based on our requirements or even in the future, we can remove
    a. It depends on real behavior in domainURL.
4. What's the maximum size of text and image we should expect in PDP page? Maybe it's hard to have number right now. But can we have some reference later in monitoring?
    a. Shared new fileds.
5. Do we have AI for FAM tech in this requirement? As a Merchant, I want to be informed why my account or listing is ineligible for BwP due to a content violation and to be able to appeal or take corrective action to get my account or listing reinstated. -
    a. Can get all info from FAM side

From Thibault

1. Build the system to support different locale.
    a. Currently, Oberon only support EN-us locale, and will mark text written in other language with [FOREIGN_LANGUAGE] label. We can add a new locale parameter in TLR moderationStation to let client specify the language they need to moderate. Or we can add it to individual moderationTask to allow moderation of different language content within one moderationStation. In the meanwhile, we need new models to support Locale. **Japan is our next step and when we need to support japanese? Will it be fine to target japanese model in P2?**
2. This is a new service right? Will it have a TLR? Control plane? What's the tenancy level?
    a. Moderation service is a new service can handle one or more different cases of moderation like Catalog, domainUrl etc as product level TLR. Currently we only have catalog/domain URL case as tenants and we will hard code for both. In P1/P2, we will have plan to set up control plane to support more tenants here.
3. Have we looked into using distributed map in stepFunction to handle the aggregation/split?
    a. This is a good call out and we create one new section to discuss. In summary, stepFunction provides map state to drive executions in parallel. It can tell us when all tasks are ready to aggregate, and east to set up threshold for Step Function time out. However, we are facing scaling issue as open Map Runs has 1000 hard quota. Also redive and recheck finished tasks will need extra efforts for engineers. More details: How do we aggregate data: check

DynamoDb or use StepFunction map?

   b. So for now, we target to focus on current use case and launch plan. In catalog data moderation and an early stage of domainURL moderation, the design is not facing the scaling concerns. Then once we finish the implementation, we will use load tests to check the performance and decide the next step with potential bottleneck.

4. If there is a large batch of task that complete as the same time, unless you set the concurrency to 1, some task that belong to the same assessment can be distributed alongside multiple batch, and call update at the same time.
   a. Yeah Agree. The batch can mitigate and we still have a risk if we are processing really huge requests at the same time. But we still need to combine with the performance from upstream. After Oberon finishes the load test, we will know the limits from upstream and estimate the maximum number we could handle.

Meeting agenda - Aug 29th

Background

1. FAM Moderation Solution Discussion
2. Catalog - 1st launch
3. URL Pages(Domain)- 2nd launch

Topics

1. The right place to batch processing and aggregation with Oberon's roadmap.
2. Consume moderation results:
   a. [Catalog] - SNS endpoint with task id, and we call Oberon to get results
      i. Geoffery: Currently, Oberon already stored the moderation result in ModerationTask dynamoDb table. It is necessary to storage the same data in S3? Is it possible to re-use current DynamoDb storage to publish to FAM Datalake? If we have moderation result stored in both DynamoDb and S3, we need to make sure data is in sync, and maintaining separate storage for same data might not be ideal.
      ii. 400K limits in dynamoDB: 1333 module results. (x * 2 + y)
   b. [Domain URL] - SNS endpoint with S3 - Temp storage
      i. Save results in S3
3. The highest TPS Oberon can handle.
   a. [Catalog] Currently all APIs except `createModerationTask` has a throttling threshold of 100TPS globally and 50TPS for a single moderation station ID. For `createModerationTask`, the config is stage-wise:
      i. In non-prod stages, it allows 10TPS for both globally and per moderation station;
      ii. In prod stages, it allows 30TPS for global rate and 20TPS per moderation station.
   b. [Domain URL] Optimization to support higher TPS
4. Build the system to support different locale. (From Thibault)
   a. Currently, Oberon only support EN-us locale, and will mark text written in other language with [FOREIGN_LANGUAGE] label. We can add a new locale parameter in TLR moderationStation to let client specify the language they need to moderate. Or we can add it to individual moderationTask to allow moderation of different language content within one moderationStation.
5. [Domain URL] Using step function to aggregate. (From Thibault)
6. Any tasks in Oberon which FAM design may not cover.
7. Next step: The time estimation
8. Next step: Ownership for Oberon task.

ACTION ITEMS

CatalogConnector

1. update time in the field to help us estimate final

Moderation

1. Right place to dedupe or filter the same catalog with diff version
2. Create a lambda to trigger cloudwatch to monitor oldest record
3. <Appconfig> enable rollback alarms
4. ~~Save sku in dynamoDB~~
5. ~~Update Test to validate dao. and clean the test~~
6. ModerationServiceRole can be assumed by ops role, need to update the ops role permissions

Aggregator lambda

1. ~~Map Oberon task failure status and add an allowlist for the expected status (to be mapped to “completed”)~~
2. ~~Call UpdateRiskOutcome (picking up in Sprint Z)~~
3. [Not the launch blocker]Handle number of tasks is one case — a short cut.
4. [Not the launch blocker]SNS signature validation
5. ~~Refractor handler lambda: add a taskProcessor": Currently we are using oberonServiceAccessor, but later we may need to aggregate other inputs. Could we simplify the logic in event handler, and create a taskProcessor to get info with oberonServiceAccessor?~~
6. ~~Integ test+ Dao: delete test records. Also delete the test records generated by the evaluate api integ test (picking up)~~
7. ~~Use external id as riskOutcomeWithIndex and call getItem instead of query (to save Read Capacity Unit)~~
8. [Not the launch blocker] Enable parallel run in Hydra integration tests
9. Fix canary test issue
10. We have ModerationServiceDependencyFailureException and FAMModerationServiceDependencyFailureException. We would better create a one in common model and

Paragon related

1. [RIT] Create an excel doc and SOP to summarize the MANUAL results
2. [RIT] ask for test queue and final prod queue
3. Paragon Feature: image url allow clickable links on their cases
4. Manual work - dedupe and auto close(p1)
5. Handle the image is not reachable in paragon
6. ~~How to get email to with ownerId~~
7.

Others

1. ~~Status for the Privacy review in CRM - Done~~
2. Create a common util package

Oberon

1. ~~Create station for each~~
2. ~~(Optimize) Create a new role to handle station creation.~~
3. ~~Update Jade tests to for the role to create station.~~
4. Silence of task
5. StepFunction migration → Will it block us?
6. Pending task → punted? alarms? (p1 for /p0)
7. Scaling → when they load testing with stepFunction? → check the processing check
8.

Updates

1. Geting the catalog data
    a. We are doing the E2E test with catalog team to retrive test store in prod
    b. Meeting with Anatha today for the fields we need in Paragon case
2. Moderation part
    a. Working on the integration Oberon.
    b. TPS estimation with catalog
        i. Image TPS
        ii. Create Task TPS
    c. Testing - Katrina
    d. Alarms in Phase 3

Have an action list
Have a launch plan
Address Load tests, Check TPS (real + test traffic)

-