End-to-End Processing Flow: Upload Through Preparation Workflow

Overview

This document describes the complete happy-path flow from a user uploading an audio file in the webapp through to the handoff to downstream Agentic Processing. It covers every handoff, queue message, S3 object, DynamoDB write, CloudWatch log entry, and Step Function state along the way.

---

1. User Uploads a File

After logging in via the Cognito-hosted UI (accessible from the webapp at `https://kiro.geiserai.com`), the user uploads an audio file.

Accepted audio formats:

- `.mp3`
- `.wav`
- `.m4a`
- `.aac`

> Video formats (.mp4, .mov, .webm) are recognized by the system but video processing is currently disabled via feature flag. Uploading a video file will result in a processing failure with the reason "Video processing is not currently enabled."

---

2. Upload Service: File Storage and Message Trigger

2.1 Presigned URL Generation

When the user submits the upload form, the webapp calls `POST /submissions` on the Upload Service API (API Gateway → Lambda). The handler:

1. Validates metadata (title, filename, content type, file size)
2. Generates a unique `submission_id`
3. Creates a DynamoDB record in `prescoach-dev-kiro-submissions` with `processing_status = Pending`
4. Generates a presigned S3 PUT URL and returns it to the client

CloudWatch Logs: `/aws/lambda/prescoach-dev-kiro-upload` — logs request validation, submission ID generation, presigned URL creation.

2.2 File Upload to S3

The webapp uses the presigned URL to upload the file directly to S3.

S3 object created:

```
Bucket: prescoach-dev-kiro-uploads
Key:    uploads/{user_id}/{submission_id}/{original_file_name}
```

Example:

```
s3://prescoach-dev-kiro-uploads/uploads/abc123/sub-98765/my-presentation.mp3
```

2.3 S3 Event Triggers Confirm Upload

An S3 PutObject event notification triggers the `confirm_upload` Lambda.

CloudWatch Logs: `/aws/lambda/prescoach-dev-kiro-confirm-upload` — logs the S3 event key, submission lookup, and SQS publish result.

This handler:

1. Parses the S3 key to extract `user_id` and `submission_id`
2. Looks up the submission record in DynamoDB
3. Constructs an SQS message and publishes it to the Preparation Workflow Input Queue

2.4 SQS Message Published

Queue: `prescoach-dev-preparation-input` (Standard Queue)

Message body:

```
{
  "submission_id": "sub-98765",
  "user_id": "abc123",
  "s3_bucket": "prescoach-dev-kiro-uploads",
  "s3_file_key": "uploads/abc123/sub-98765/my-presentation.mp3",
```

```
  "original_file_name": "my-presentation.mp3",
  "presentation_title": "Q4 Review"
}
```

Handoff summary:

From	To	Mechanism	Payload
Upload Service (`confirm_upload` Lambda)	Preparation Workflow	SQS `prescoach-dev-preparation-input`	JSON message above

---

3. EventBridge Pipe: SQS → Step Functions

An EventBridge Pipe (`prescoach-dev-prep-input-to-sfn`) monitors the Input Queue and starts a Step Functions execution for each message.

- **Batch size:** 1 (one execution per message)
- **Invocation type:** FIRE_AND_FORGET

CloudWatch Logs: EventBridge Pipes logs to `/aws/pipes/prescoach-dev-prep-input-to-sfn` (if enabled).

---

4. Step Functions: Preparation Workflow (Happy Path)

State Machine: `prescoach-dev-preparation-workflow` (Standard Workflow)

CloudWatch Logs: `/aws/stepfunctions/prescoach-dev-preparation-workflow` — logs all state transitions, input/output for each state, execution data.

State 1: LoadConfig

- **Type:** Task (Lambda invoke)
- **Lambda:** `prescoach-dev-prep-loadconfig`
- **What it does:** Reads all 10 SSM parameters from `/prescoach/dev/preparation-workflow/` and returns them as a config object
- **Output stored at:** `$.config.value`
- **CloudWatch Logs:** `/aws/lambda/prescoach-dev-prep-loadconfig`

Config values loaded:

Parameter	Example Value
`embedding-model-id`	`amazon.nova-2-multimodal-embeddings-v1:0`
`chunk-size-seconds`	`30`
`chunk-overlap-seconds`	`5`
`max-retry-attempts`	`3`
`video-processing-enabled`	`false`
`vector-store-endpoint`	`prescoach-dev-kiro-uploads`
`vector-store-type`	`s3`
`batch-size`	`10`
`batch-processing-enabled`	`false`
`embeddings-enabled`	`false`

State 2: ParseMessage

- **Type:** Task (Lambda invoke)
- **Lambda:** `prescoach-dev-prep-parsemessage`
- **What it does:** Deserializes the SQS message body JSON into a validated `InputMessage` model
- **Output stored at:** `$.parsed_message.value`
- **CloudWatch Logs:** `/aws/lambda/prescoach-dev-prep-parsemessage`

State 3: UpdateStatusProcessing

- **Type:** Task (DynamoDB SDK integration)
- **What it does:** Updates the submission record in DynamoDB
- **Table:** `prescoach-dev-kiro-submissions`
- **Update:** `SET processing_status = "Processing"`
- **CloudWatch Logs:** Logged in the Step Functions execution history (no separate Lambda log)

State 4: ValidateFileFormat

- **Type:** Task (Lambda invoke)
- **Lambda:** `prescoach-dev-prep-validateformat`

- **What it does:** Checks the file extension against accepted formats and applies the video processing feature flag logic
- **Output stored at:** `$.validation_result.value`
- **CloudWatch Logs:** `/aws/lambda/prescoach-dev-prep-validateformat`

Output example (audio file, happy path):

```
{
  "valid": true,
  "decision": "embed",
  "reason": "Audio file proceeds directly to embedding",
  "file_type": "audio"
}
```

State 5: CheckVideoFlag

- **Type:** Choice
- **What it does:** Routes based on `$.validation_result.value.decision`:
  - `"embed"` → **TranscribeAudio** (audio file — this is the happy path)
  - `"extract_audio"` → ExtractAudio (video with flag enabled)
  - `"fail"` → HandleFailure

State 5b: ExtractAudio (video path only)

- **Type:** Task (Lambda invoke)
- **Lambda:** `prescoach-dev-prep-extractaudio`
- **What it does:** Submits a MediaConvert job to extract the audio track from a video file
- **Output stored at:** `$.extraction_result.value`
- **CloudWatch Logs:** `/aws/lambda/prescoach-dev-prep-extractaudio`
- **Next:** TranscribeAudio

State 6: TranscribeAudio

- **Type:** Task (Lambda invoke)
- **Lambda:** `prescoach-dev-prep-transcribeaudio`
- **What it does:** Starts an Amazon Transcribe job with automatic language detection, polls for completion (up to 5 minutes), and returns the S3 key of the transcript
- **Output stored at:** `$.transcribe_result.value`
- **CloudWatch Logs:** `/aws/lambda/prescoach-dev-prep-transcribeaudio`

S3 object created:

```
Bucket: prescoach-dev-kiro-uploads
Key:    transcripts/{submission_id}/transcript.txt
```

Output example:

```
{
  "transcript_s3_key": "transcripts/sub-98765/transcript.txt"
}
```

State 7: CheckEmbeddingsEnabled

- **Type:** Choice
- **What it does:** Routes based on `$.config.value.embeddings_enabled`:
  - `true` → **ChunkAudio** (proceed with embedding pipeline)
  - `false` → **SetDefaultsForSkippedEmbeddings** (skip to handoff)

When embeddings are disabled, the workflow sets empty defaults for `store_result` and `chunks` via two Pass states (SetDefaultsForSkippedEmbeddings → SetDefaultChunks) and proceeds directly to PublishHandoff. This allows the evaluation agents to work from the transcript alone without requiring embeddings.

State 8: ChunkAudio

- **Type:** Task (Lambda invoke)
- **Lambda:** `prescoach-dev-prep-chunkaudio`
- **What it does:** Divides the audio file into overlapping chunks based on config (default: 30s chunks with 5s overlap)
- **Output stored at:** `$.chunks.value`
- **CloudWatch Logs:** `/aws/lambda/prescoach-dev-prep-chunkaudio`

S3 objects created:

```
Bucket: prescoach-dev-kiro-uploads
Keys:
  processed/{user_id}/{submission_id}/chunks/chunk_0000.mp3
  processed/{user_id}/{submission_id}/chunks/chunk_0001.mp3
  processed/{user_id}/{submission_id}/chunks/chunk_0002.mp3
  ...
```

Example for a 60-second audio with 30s chunks and 5s overlap (3 chunks):

```
s3://prescoach-dev-kiro-uploads/processed/abc123/sub-98765/chunks/chunk_0000.mp3
s3://prescoach-dev-kiro-uploads/processed/abc123/sub-98765/chunks/chunk_0001.mp3
```

```
s3://prescoach-dev-kiro-uploads/processed/abc123/sub-98765/chunks/chunk_0002.mp3
```

State 9: CreateEmbeddings

- **Type:** Map (parallel processing)
- **Max concurrency:** 10
- **Lambda:** `prescoach-dev-prep-createembedding` (invoked per chunk)
- **What it does:** Invokes Amazon Bedrock (Nova Multimodal Embeddings) with each audio chunk's S3 URI to create a vector embedding
- **Output stored at:** `$.embeddings`
- **CloudWatch Logs:** `/aws/lambda/prescoach-dev-prep-createembedding` (one log stream per invocation)

Bedrock request payload (per chunk):

```
{
  "inputText": null,
  "inputImage": null,
  "inputAudio": "s3://prescoach-dev-kiro-uploads/processed/abc123/sub-98765/chunks/chunk_0000.mp3"
}
```

State 10: StoreVectors

- **Type:** Task (Lambda invoke)
- **Lambda:** `prescoach-dev-prep-storevectors`
- **What it does:** Writes each embedding vector with metadata to the S3 vector store bucket
- **Output stored at:** `$.store_result.value`
- **CloudWatch Logs:** `/aws/lambda/prescoach-dev-prep-storevectors`

S3 objects created:

```
Bucket: prescoach-dev-kiro-uploads
Keys:
  {submission_id}/embeddings/chunk_0000.json
  {submission_id}/embeddings/chunk_0001.json
  {submission_id}/embeddings/chunk_0002.json
  ...
```

Example JSON content of each embedding file:

```
{
  "embedding_vector": [0.123, -0.456, 0.789, ...],
  "metadata": {
    "submission_id": "sub-98765",
```

```
    "user_id": "abc123",
    "chunk_index": 0,
    "chunk_timestamp_start": 0.0,
    "chunk_timestamp_end": 30.0,
    "embedding_model_version": "amazon.nova-2-multimodal-embeddings-v1:0"
  }
}
```


State 11: PublishHandoff

- **Type:** Task (Lambda invoke)
- **Lambda:** `prescoach-dev-prep-publishhandoff`
- **What it does:** Constructs a `HandoffMessage` and publishes it to the FIFO Handoff Queue for the Agentic Evaluation service. Also triggers the eval-task-launcher Lambda asynchronously (fire-and-forget) to ensure an ECS task starts immediately without waiting for CloudWatch alarm transitions.
- **Output stored at:** `$.handoff_result.value`
- **CloudWatch Logs:** `/aws/lambda/prescoach-dev-prep-publishhandoff`

Queue: `prescoach-dev-preparation-handoff.fifo` (FIFO Queue)

Message body:


```
{
  "submission_id": "sub-98765",
  "user_id": "abc123",
  "s3_file_key": "uploads/abc123/sub-98765/my-presentation.mp3",
  "transcript_s3_key": "transcripts/sub-98765/transcript.txt",
  "vector_store_location": "s3://prescoach-dev-kiro-uploads/sub-98765/embeddings",
  "chunk_count": 3,
  "presentation_title": "Q4 Review"
}
```


FIFO properties:

- `MessageGroupId`: `sub-98765` (submission_id — preserves per-submission ordering)
- `MessageDeduplicationId`: `sub-98765-handoff`

Handoff summary:

From	To	Mechanism	Payload
Preparation Workflow (`publish_handoff` Lambda)	Agentic Processing	SQS FIFO `prescoach-dev-preparation-handoff.fifo`	HandoffMessage JSON above

State 12: UpdateStatusCompleted

- **Type:** Task (DynamoDB SDK integration)
- **What it does:** Updates the submission record
- **Table:** `prescoach-dev-kiro-submissions`
- **Update:** `SET processing_status = "Completed"`
- **CloudWatch Logs:** Logged in the Step Functions execution history

End of happy path. The workflow succeeds.

---

5. Complete S3 Objects Created (Happy Path)

Step	Bucket	Key Pattern	Content
User upload	`prescoach-dev-kiro-uploads`	`uploads/{user_id}/{submission_id}/{filename}`	Original audio file
Transcription	`prescoach-dev-kiro-uploads`	`transcripts/{submission_id}/transcript.txt`	Full text transcript
Chunking	`prescoach-dev-kiro-uploads`	`processed/{user_id}/{submission_id}/chunks/chunk_{NNNN}.mp3`	Audio chunk segments
Vector storage	`prescoach-dev-kiro-uploads`	`{submission_id}/embeddings/chunk_{NNNN}.json`	Embedding vector + metadata

---

6. Complete CloudWatch Log Groups

Component	Log Group	What's Logged
Upload handler	`/aws/lambda/prescoach-dev-kiro-upload`	Metadata validation, submission creation, presigned URL

Component	Log Group	What's Logged
Confirm upload	`/aws/lambda/prescoach-dev-kiro-confirm-upload`	S3 event parsing, DynamoDB lookup, SQS publish
Step Functions	`/aws/stepfunctions/prescoach-dev-preparation-workflow`	All state transitions, input/output, execution history
LoadConfig	`/aws/lambda/prescoach-dev-prep-loadconfig`	SSM parameter fetch
ParseMessage	`/aws/lambda/prescoach-dev-prep-parsemessage`	Message deserialization
ValidateFormat	`/aws/lambda/prescoach-dev-prep-validateformat`	Format check, decision logic
TranscribeAudio	`/aws/lambda/prescoach-dev-prep-transcribeaudio`	Transcribe job start, polling, completion
ChunkAudio	`/aws/lambda/prescoach-dev-prep-chunkaudio`	Chunk boundaries, S3 uploads
CreateEmbedding	`/aws/lambda/prescoach-dev-prep-createembedding`	Bedrock invocation, response parsing
StoreVectors	`/aws/lambda/prescoach-dev-prep-storevectors`	S3 puts for each embedding
PublishHandoff	`/aws/lambda/prescoach-dev-prep-publishhandoff`	SQS FIFO publish, eval launcher trigger
HandleFailure	`/aws/lambda/prescoach-dev-prep-handlefailure`	DynamoDB update, SNS publish, DLQ routing

---

7. Retry Logic

Every Task state in the Step Function has automatic retry configured:

State	Initial Wait	Backoff Rate	Max Attempts	Errors Caught
LoadConfig	2s	2x	3	Lambda.ServiceException, TooManyRequests, Timeout
ParseMessage	2s	2x	3	Lambda.ServiceException, TooManyRequests, Timeout
UpdateStatusProcessing	1s	2x	3	DynamoDB Throttling, InternalServerError, Timeout
ValidateFileFormat	2s	2x	3	Lambda.ServiceException, TooManyRequests, Timeout
ExtractAudio	30s	2x	3	Lambda.ServiceException, TooManyRequests, Timeout
TranscribeAudio	2s	2x	3	Lambda.ServiceException, TooManyRequests, Timeout
ChunkAudio	2s	2x	3	Lambda.ServiceException, TooManyRequests, Timeout
CreateEmbeddings (per chunk)	5s	2x	3	Lambda.ServiceException, TooManyRequests, Timeout
StoreVectors	2s	2x	3	Lambda.ServiceException, TooManyRequests, Timeout
PublishHandoff	2s	2x	3	Lambda.ServiceException, TooManyRequests, Timeout
UpdateStatusCompleted	1s	2x	3	DynamoDB Throttling, InternalServerError, Timeout

Retry sequence example (ChunkAudio with 2s initial):

1. First attempt fails → wait 2s
2. Second attempt fails → wait 4s
3. Third attempt fails → route to HandleFailure

---

8. Failure States and Error Handling

If any state fails after exhausting retries, the execution routes to the **HandleFailure** state.

HandleFailure performs three actions:

1. **Updates DynamoDB:** Sets `processing_status = "Failed"` in `prescoach-dev-kiro-submissions`
2. **Publishes SNS notification** (best-effort — if SNS publish fails, it's logged but doesn't block):
3. **Routes to Dead Letter Queue:**
   - Failures from input processing → `prescoach-dev-preparation-dlq-input`
   - Failures from handoff publishing → `prescoach-dev-preparation-dlq-handoff.fifo`

Failure scenarios:

Failure Point	Cause	DLQ Used
ParseMessage	Invalid/malformed JSON, missing fields	DLQ Input
ValidateFileFormat	Unsupported format (.exe, .pdf, etc.)	DLQ Input
ValidateFileFormat	Video file + flag disabled	DLQ Input
ExtractAudio	MediaConvert job failure after retries	DLQ Input
TranscribeAudio	Transcribe job failure or timeout after retries	DLQ Input
ChunkAudio	S3 read/write failure after retries	DLQ Input
CreateEmbeddings	Bedrock throttling/failure after retries	DLQ Input
StoreVectors	Vector store write failure after retries	DLQ Input
PublishHandoff	FIFO queue unavailable after retries	DLQ Handoff

SNS Error Notification Format

Published to topic: `prescoach-dev-preparation-errors`

```
{
  "submission_id": "sub-98765",
  "step_name": "CreateEmbeddings",
  "error_type": "BedrockServiceError",
```

```
  "error_message": "Model invocation failed: ThrottlingException",
  "retry_count_exhausted": 3,
  "timestamp": "2026-06-12T03:45:22+00:00",
  "queue_name": "DLQ_Input"
}
```

---

9. DynamoDB State Transitions

The `processing_status` field in `prescoach-dev-kiro-submissions` progresses through:

```
Pending → Processing → Completed
                    ↘ Failed (on any unrecoverable error)
```

State	Set By	When
`Pending`	Upload Service (`POST /submissions`)	User initiates upload
`Processing`	Step Functions (UpdateStatusProcessing state)	Workflow begins
`Completed`	Step Functions (UpdateStatusCompleted state)	Handoff published successfully
`Failed`	HandleFailure Lambda	Any state fails after retries

---

10. How to Verify End-to-End Processing

After uploading a file, check these in order:

Check 1: DynamoDB status

```
aws dynamodb get-item \
  --table-name prescoach-dev-kiro-submissions \
  --key '{"submission_id": {"S": "YOUR_SUBMISSION_ID"}}' \
  --query 'Item.processing_status.S' \
  --output text \
  --region us-east-1
```

Expected: `Completed`

Check 2: Step Functions execution

```
aws stepfunctions list-executions \
  --state-machine-arn arn:aws:states:us-east-1:<YOUR_ACCOUNT_ID>:stateMachine:prescoach-dev-preparati
  --max-results 5 \
  --query 'executions[*].[name,status,startDate]' \
  --output table \
  --region us-east-1
```

Expected: Status = `SUCCEEDED`

Check 3: Vector embeddings stored

```
aws s3 ls s3://prescoach-dev-kiro-uploads/YOUR_SUBMISSION_ID/embeddings/
```

Expected: List of `chunk_NNNN.json` files (only present if `embeddings-enabled` is `true`)

Check 4: Handoff message on queue

```
aws sqs get-queue-attributes \
  --queue-url https://sqs.us-east-1.amazonaws.com/<YOUR_ACCOUNT_ID>/prescoach-dev-preparation-handoff
  --attribute-names ApproximateNumberOfMessages \
  --region us-east-1
```

Expected: At least 1 message (unless Agentic Processing has already consumed it)