**14. Services Interface Documentation - API Reference**

**REST API Endpoints**

**GET /tasks**

**Purpose**: Retrieve tasks based on user role

**Request:**

GET https://api-gateway-url/prod/tasks

Authorization: <JWT\_TOKEN>

**Headers:**

* Authorization: JWT token from Cognito (Required)
* Content-Type: application/json

**JWT Token Claims:**

{

"sub": "user-uuid",

"email": "user@gmail.com",

"cognito:groups": ["users"] or ["users", "admins"]

}

**Processing Logic:**

1. API Gateway validates JWT via Cognito Authorizer
2. Lambda extracts user ID and groups from claims
3. If user has 'admins' group: Scans entire Tasks table
4. If regular user: Queries by userId partition key

**Response - Success (200):**

[

{

"userId": "abc-123",

"taskId": "def-456",

"title": "Complete documentation",

"description": "Finish project docs",

"priority": "high",

"dueDate": "2025-06-30",

"status": "pending",

"createdAt": "2025-06-19T10:30:00Z",

"userEmail": "user@gmail.com"

}

]

**Service Dependencies:**

* **Cognito**: Token validation
* **Lambda**: TaskHandler execution
* **DynamoDB**: Read from Tasks table

**POST /tasks**

**Purpose**: Create new task with email notification

**Request:**

POST https://api-gateway-url/prod/tasks

Authorization: <JWT\_TOKEN>

Content-Type: application/json

{

"title": "New Task",

"description": "Task description",

"priority": "medium",

"dueDate": "2025-07-01"

}

**Processing:**

1. Generate UUID for taskId
2. Add metadata (userId, createdAt, status, userEmail)
3. Save to DynamoDB
4. Retrieve SNS ARN from Parameter Store
5. Publish notification

**Response - Success (201):**

{

"userId": "user-123",

"taskId": "generated-uuid",

"title": "New Task",

"description": "Task description",

"priority": "medium",

"dueDate": "2025-07-01",

"status": "pending",

"createdAt": "2025-06-19T10:30:00Z",

"userEmail": "user@gmail.com"

}

**SNS Message Format:**

Subject: New Task Created

Message: User user@gmail.com created a new task: New Task

Priority: medium

Description: Task description

**Service Dependencies:**

* **Systems Manager**: Get /taskflow/config
* **DynamoDB**: Write to Tasks table
* **SNS**: Publish notification

**PUT /tasks/{taskId}**

**Purpose**: Update existing task

**Request:**

PUT https://api-gateway-url/prod/tasks/task-uuid

Authorization: <JWT\_TOKEN>

Content-Type: application/json

{

"title": "Updated Title",

"description": "Updated description",

"priority": "high",

"dueDate": "2025-07-15",

"status": "completed"

}

**DynamoDB Update Expression:**

UpdateExpression = "SET #status = :status, title = :title, updatedAt = :updatedAt"

ExpressionAttributeNames = {"#status": "status"} # Reserved word

ExpressionAttributeValues = {

":status": "completed",

":title": "Updated Title",

":updatedAt": "2025-06-19T10:30:00Z"

}

**Response - Success (200):**

{

"message": "Task updated successfully"

}

**Error Cases:**

* 404: Task not found or user doesn't own task
* 500: DynamoDB update failed

**DELETE /tasks/{taskId}**

**Purpose**: Delete task permanently

**Request:**

DELETE https://api-gateway-url/prod/tasks/task-uuid

Authorization: <JWT\_TOKEN>

**Admin vs User Logic:**

if is\_admin:

# Scan to find task owner

scan\_result = tasks\_table.scan(

FilterExpression='taskId = :tid',

ExpressionAttributeValues={':tid': task\_id}

)

# Delete with found userId

else:

# Direct delete with user's ID

tasks\_table.delete\_item(

Key={'userId': user\_id, 'taskId': task\_id}

)

**Response - Success (200):**

{

"message": "Task deleted"

}

**POST /tasks/bulk-import**

**Purpose**: Queue multiple tasks for async processing

**Request:**

POST https://api-gateway-url/prod/tasks/bulk-import

Authorization: <JWT\_TOKEN>

Content-Type: application/json

{

"tasks": [

{

"title": "Task 1",

"description": "Description 1",

"priority": "low",

"dueDate": "2025-07-01"

}

]

}

**SQS Message Format:**

{

"action": "create\_task",

"userId": "user-123",

"userEmail": "user@gmail.com",

"task": {

"title": "Task 1",

"description": "Description 1",

"priority": "low",

"dueDate": "2025-07-01"

}

}

**Response - Success (202):**

{

"message": "Successfully queued 25 tasks for import",

"total": 25

}

**Service Flow:**

1. Get SQS URL from Parameter Store
2. Send messages in batches of 10
3. Send admin notification via SNS
4. SQS triggers SQSProcessor Lambda

**GET /admin/analytics**

**Purpose**: Calculate system-wide statistics (admin only)

**Request:**

GET https://api-gateway-url/prod/admin/analytics

Authorization: <JWT\_TOKEN\_WITH\_ADMIN\_GROUP>

**Authorization Check:**

groups = claims.get('cognito:groups', '').split(',')

if 'admins' not in groups:

return 403 Forbidden

**Response - Success (200):**

{

"totalTasks": 150,

"tasksByStatus": {

"pending": 100,

"completed": 50

},

"tasksByPriority": {

"high": 30,

"medium": 70,

"low": 50

},

"tasksByUser": {

"user1@gmail.com": 75,

"user2@gmail.com": 75

},

"recentTasks": [

{

"taskId": "task-123",

"title": "Recent Task",

"userEmail": "user@gmail.com",

"createdAt": "2025-06-19T10:30:00Z"

}

]

}

**DynamoDB Operations:**

* Scan entire Tasks table
* Calculate aggregations in memory
* Store results in Analytics table

**Lambda Functions - Detailed Interfaces**

**TaskFlow-TaskHandler**

**Runtime**: Python 3.9  
**Memory**: 256 MB  
**Timeout**: 30 seconds

**Event Structure from API Gateway:**

{

"httpMethod": "GET|POST|PUT|DELETE",

"path": "/tasks",

"pathParameters": {

"taskId": "uuid" // For PUT/DELETE

},

"headers": {

"Authorization": "JWT\_TOKEN"

},

"requestContext": {

"authorizer": {

"claims": {

"sub": "user-id",

"email": "user@gmail.com",

"cognito:groups": "users,admins"

}

}

},

"body": "{JSON\_STRING}" // For POST/PUT

}

**Environment Variables**: None (uses Parameter Store)

**IAM Permissions:**

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"dynamodb:PutItem",

"dynamodb:GetItem",

"dynamodb:UpdateItem",

"dynamodb:DeleteItem",

"dynamodb:Query",

"dynamodb:Scan"

],

"Resource": "arn:aws:dynamodb:\*:\*:table/TaskFlow-Tasks"

},

{

"Effect": "Allow",

"Action": "sns:Publish",

"Resource": "arn:aws:sns:\*:\*:TaskFlow-Notifications"

},

{

"Effect": "Allow",

"Action": [

"sqs:SendMessage",

"sqs:SendMessageBatch"

],

"Resource": "arn:aws:sqs:\*:\*:TaskFlow-ProcessingQueue"

},

{

"Effect": "Allow",

"Action": "ssm:GetParameter",

"Resource": "arn:aws:ssm:\*:\*:parameter/taskflow/\*"

}

]

}

**TaskFlow-Analytics**

**Triggers**: EventBridge (daily) or API Gateway

**EventBridge Event:**

{

"source": "aws.events",

"detail-type": "Scheduled Event",

"time": "2025-06-19T00:00:00Z"

}

**Processing Steps:**

1. Determine trigger type (EventBridge vs API)
2. If API: Verify admin group
3. Scan entire Tasks table
4. Calculate statistics
5. Store in Analytics table

**Analytics Table Item:**

{

"date": "2025-06-19",

"metric": "daily\_stats",

"timestamp": "2025-06-19T00:00:00Z",

"stats": {

"totalTasks": 150,

"tasksByStatus": {...},

"tasksByPriority": {...},

"tasksByUser": {...}

}

}

**TaskFlow-SQSProcessor**

**Trigger**: SQS Queue  
**Batch Size**: 10  
**Visibility Timeout**: 300 seconds

**SQS Event Structure:**

{

"Records": [

{

"messageId": "msg-123",

"body": "{\"action\":\"create\_task\",\"userId\":\"...\",\"task\":{...}}",

"receiptHandle": "handle-123"

}

]

}

**Processing:**

for record in event['Records']:

message = json.loads(record['body'])

if message.get('action') == 'create\_task':

# Create task in DynamoDB

# Track success/failure

# Send completion notification via SNS

**Return Format:**

{

"statusCode": 200,

"body": {

"created": 8,

"failed": 2

}

}

**TaskFlow-PreSignupValidation**

**Trigger**: Cognito Pre Sign-up

**Event Structure:**

{

"userPoolId": "us-east-1\_xxxxx",

"request": {

"userAttributes": {

"email": "user@gmail.com",

"name": "User Name"

}

}

}

**Validation Logic:**

if not email.endswith('@gmail.com'):

raise Exception('Registration is restricted to Gmail users only')

return event # Allow registration

**TaskFlow-AutoAssignGroup**

**Trigger**: Cognito Post Confirmation

**Event Structure:**

{

"userPoolId": "us-east-1\_xxxxx",

"userName": "user@gmail.com",

"request": {},

"response": {}

}

**Group Assignment:**

cognito.admin\_add\_user\_to\_group(

UserPoolId=user\_pool\_id,

Username=username,

GroupName='users'

)

**Supporting Services Configuration**

**Amazon SNS**

**Topic**: TaskFlow-Notifications  
**Message Attributes**: None  
**Delivery Protocol**: Email

**Message Types:**

1. New Task Created
2. CSV Import Started
3. CSV Import Completed

**Amazon SQS**

**Queue**: TaskFlow-ProcessingQueue  
**Configuration:**

{

"VisibilityTimeout": 300,

"MessageRetentionPeriod": 1209600,

"MaximumMessageSize": 262144,

"ReceiveMessageWaitTimeSeconds": 0,

"DelaySeconds": 0

}

**AWS Systems Manager Parameter Store**

**Parameter**: /taskflow/config  
**Type**: String  
**Value Structure:**

{

"snsTopicArn": "arn:aws:sns:us-east-1:123456:TaskFlow-Notifications",

"sqsQueueUrl": "https://sqs.us-east-1.amazonaws.com/123456/TaskFlow-ProcessingQueue"

}

**Access Pattern:**

response = ssm.get\_parameter(Name='/taskflow/config')

config = json.loads(response['Parameter']['Value'])

**Amazon EventBridge**

**Rule**: TaskFlow-DailyAnalytics  
**Schedule**: cron(0 0 \* \* ? \*)  
**Target**: Analytics Lambda

**Input Template**: None (uses default event)

**DynamoDB Table Schemas**

**TaskFlow-Tasks Table**

**Keys:**

* Partition Key: userId (String)
* Sort Key: taskId (String)

**Item Structure:**

{

"userId": "user-123",

"taskId": "task-456",

"title": "Task Title",

"description": "Task Description",

"priority": "high|medium|low",

"dueDate": "2025-06-30",

"status": "pending|completed",

"createdAt": "2025-06-19T10:30:00Z",

"updatedAt": "2025-06-19T11:00:00Z",

"userEmail": "user@gmail.com"

}

**Access Patterns:**

1. Get user tasks: Query by userId
2. Get specific task: Query by userId + taskId
3. Admin view all: Scan entire table
4. Find task by ID only: Scan with filter

**TaskFlow-Analytics Table**

**Keys:**

* Partition Key: date (String, YYYY-MM-DD)
* Sort Key: metric (String)

**Item Structure:**

{

"date": "2025-06-19",

"metric": "daily\_stats",

"timestamp": "2025-06-19T00:00:00Z",

"stats": {

"totalTasks": 150,

"tasksByStatus": {

"pending": 100,

"completed": 50

},

"tasksByPriority": {

"high": 30,

"medium": 70,

"low": 50

},

"tasksByUser": {

"user1@gmail.com": 75,

"user2@gmail.com": 75

},

"recentTasks": []

}

}