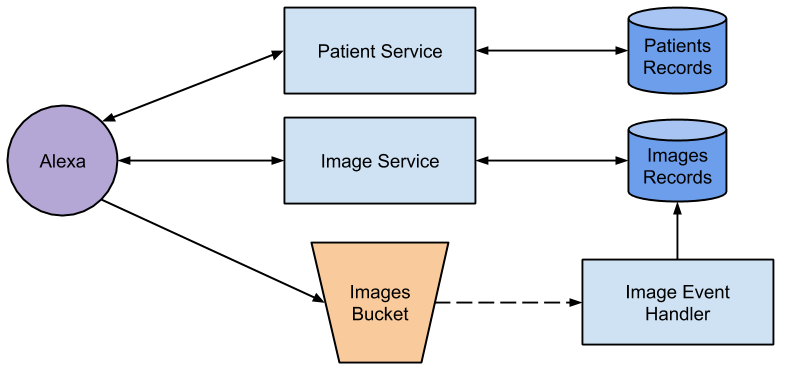
Dr. Cloud

**Core Platform Reference**



Alexa Skill Lambda implementation (*Alexa*) should communicate with Patient Service, Image Service and S3 bucket for images (*Bucket*).

A typical work-flow is:

1. Alexa creates a new Patient or get an existing Patient via the Patient Service.
2. Alexa saves new images to the Bucket on the location provided by the Patient Service in 1.
3. Alexa shows all the images for the Patient via the Image Service.

# Patient Service

The Patient Service is an AWS Lambda providing operations to a patient catalogue.

In the back-end is backed with an Amazon DynamoDB database.

## Request

The request payload has the following format (JSON):

{ action: "list"|"get"|"create"|"update"|"delete",  
 body: OBJECT|null }

## Operations

### List All the Patients

Action: list

Body: <empty>

Response: *Array of Patient objects*  
[ { patientId: STRING,  
 familyName: STRING,  
 givenName: STRING,  
 storage: {   
 bucket: STRING,  
 key: STRING } }, ... ]

### Get an Existing Patient

Action: get

Body: { patientId: STRING }

Response: *Patient object*  
{ patientId: STRING,  
 familyName: STRING,  
 givenName: STRING,  
 storage: {   
 bucket: STRING,  
 key: STRING } }

### Create a new Patient

Action: create

Body:   
{ givenName: STRING,  
 familyName: STRING }

Response: *Created Patient object*  
{ patientId: STRING,  
 familyName: STRING,  
 givenName: STRING,  
 storage: {   
 bucket: STRING,  
 key: STRING } }

### Update an Existing Patient

Action: update

Body:   
{ patientId: STRING,  
 givenName: STRING,  
 familyName: STRING }

Response: *Updated Patient object*  
{ patientId: STRING,  
 familyName: STRING,  
 givenName: STRING,  
 storage: {   
 bucket: STRING,  
 key: STRING } }

### Delete an Existing Patient

Action: delete

Body: { patientId: STRING }

Response: <empty>

## Example

const AWS = require('aws-sdk')  
const lambda = new AWS.Lambda({apiVersion: '2015-03-31'})  
  
const PATIENT\_LAMBDA = process.env.PATIENT\_LAMBDA  
  
exports.handler = async event => {  
   
 const patientCreated = JSON.parse((await lambda.invoke({  
 FunctionName: PATIENT\_LAMBDA,  
 Payload: JSON.stringify({  
 action: 'create',  
 body: {  
 givenName: 'Alfred',  
 familyName: 'Neuman'  
 }  
 })  
 }).promise()).Payload)  
 console.log('patientCreated', patientCreated)  
   
 const patientGotten = JSON.parse((await lambda.invoke({  
 FunctionName: PATIENT\_LAMBDA,  
 Payload: JSON.stringify({  
 action: 'get',  
 body: {  
 patientId: patientCreated.patientId  
 }  
 })  
 }).promise()).Payload)  
 console.log('patientGotten', patientGotten)  
   
 const patientUpdated = JSON.parse((await lambda.invoke({  
 FunctionName: PATIENT\_LAMBDA,  
 Payload: JSON.stringify({  
 action: 'update',  
 body: {  
 patientId: patientCreated.patientId,  
 givenName: 'Alfred EDIT',  
 familyName: 'Neuman EDIT'  
 }  
 })  
 }).promise()).Payload)  
 console.log('patientUpdated', patientUpdated)  
   
 const patientList = JSON.parse((await lambda.invoke({  
 FunctionName: PATIENT\_LAMBDA,  
 Payload: JSON.stringify({  
 action: 'list'  
 })  
 }).promise()).Payload)  
 console.log('patientList', patientList)  
   
 const patientDeleted = JSON.parse((await lambda.invoke({  
 FunctionName: PATIENT\_LAMBDA,  
 Payload: JSON.stringify({  
 action: 'delete',  
 body: {  
 patientId: patientCreated.patientId  
 }  
 })  
 }).promise()).Payload)  
 console.log('patientDeleted', patientDeleted) // null  
}

# Image Service

After saving into the Bucket, the Images are automatically loaded (via the Image Event Handler) into the Image Service.

The Image Service is an AWS Lambda providing operations to an image catalogue.

In the back-end is backed with an Amazon DynamoDB database.

## Request

The request payload has the following format (JSON):

{ action: "list"|"get",  
 body: OBJECT|null }

## Operations

### List All the Images for a Patient

Action: list

Body: { patientId: STRING }

Response: *Array of Image objects*  
[ { imageId: STRING,  
 patientId: STRING,  
 fileName: STRING,  
 s3Bucket: STRING,  
 s3Key: STRING }, ... ]

### Get an Existing Image

Action: get

Body: { imageId: STRING }

Response: *Image object*  
{ imageId: STRING,  
 patientId: STRING,  
 fileName: STRING,  
 s3Bucket: STRING,  
 s3Key: STRING }

## Example

const AWS = require('aws-sdk')  
const lambda = new AWS.Lambda({apiVersion: '2015-03-31'})  
  
const IMAGE\_LAMBDA = process.env. IMAGE\_LAMBDA  
  
exports.handler = async event => {  
 const patientId = event.patientId  
   
 const imageList = JSON.parse((await lambda.invoke({  
 FunctionName: IMAGE\_LAMBDA,  
 Payload: JSON.stringify({  
 action: 'list',   
 body: {  
 patientId  
 }  
 })  
 }).promise()).Payload)  
 console.log('imageList', imageList)  
   
 const imageGotten = JSON.parse((await lambda.invoke({  
 FunctionName: IMAGE\_LAMBDA,  
 Payload: JSON.stringify({  
 action: 'get',  
 body: {  
 imageId: imageList[0].imageId  
 }  
 })  
 }).promise()).Payload)  
 console.log('imageGotten', imageGotten)  
}