

AmazonConnectによる自動電話通知（8.周回数の指定）

AWS



×

ZABBIX

×



AmazonConnectによる自動電話通知 （8.周回数の指定）

2021.11.14 2021.11.08

[【前回】 AmazonConnectによる自動電話通知（7.複数連絡先への電話通知〈構築⑨〉）](#)[【次回】 AmazonConnectによる自動電話通知（9.利用料金）](#)[【簡易版】 AmazonConnectによる自動電話通知（まとめ）](#)

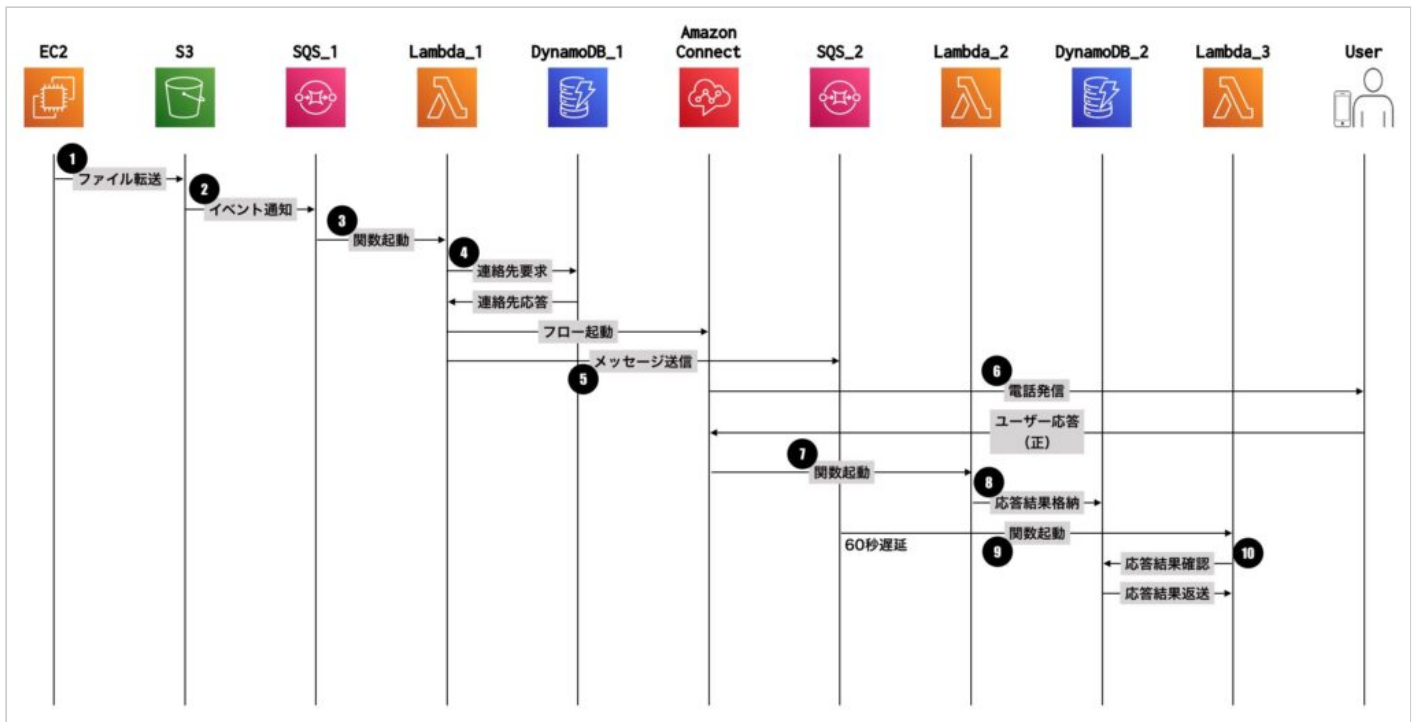
監視サーバーで障害を検知した際に、自動で電話通知できるようにしていきます。ネットワークエンジニアも利用することの多い監視サーバー(Zabbix)で障害検知し、AWS上のAmazonConnectを利用し自動電話を発信します。

今回は下記の条件を満たせるようにAWSの各サービスを利用して自動電話通知の仕組みを導入します。

- 複数の通知先を登録した連絡先リストを持たせる。
- 連絡先リストに優先度(通知順)を設定する。
- 優先度が高い人に最初に電話する。
- 応答が無かった場合、次の優先度の人に順番に電話する。
- 連絡先リストの最後まで電話しても応答が無かった場合、最初に戻って継続する。

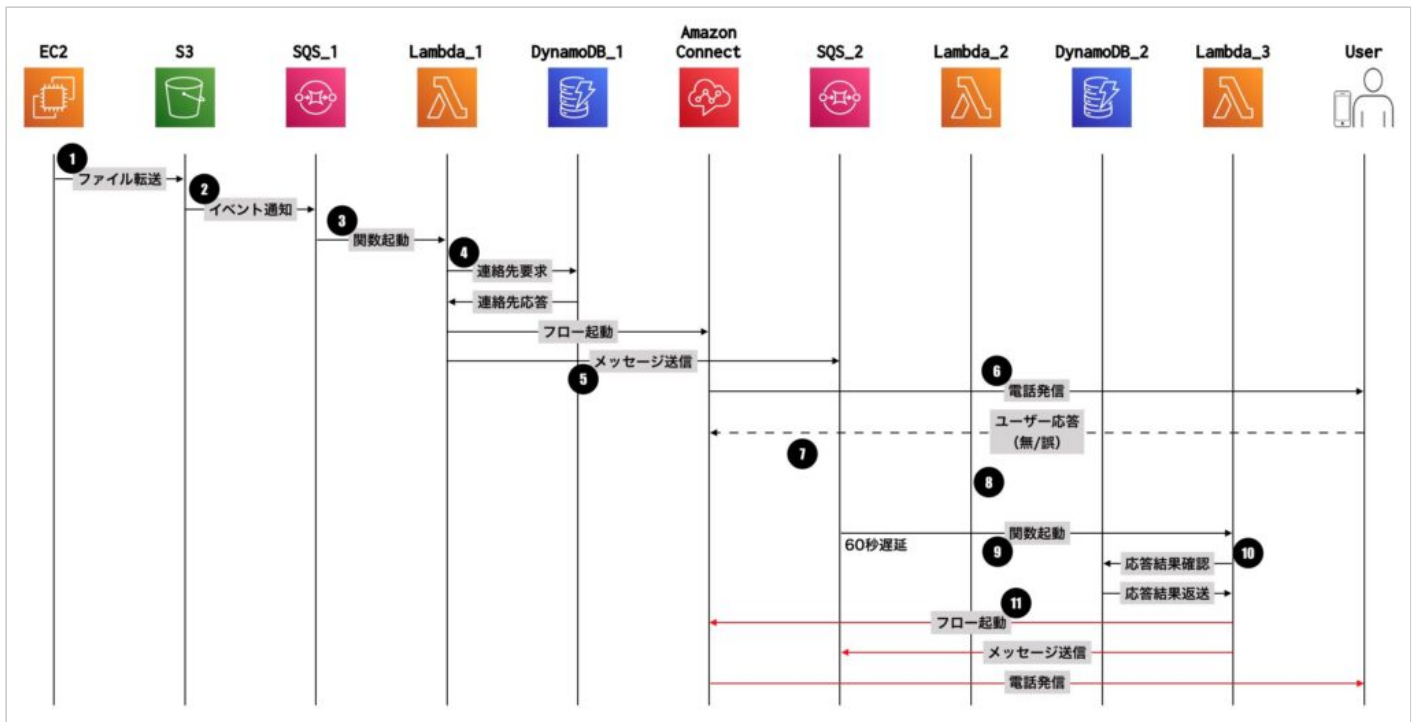
自動電話通知フロー

電話に応答した場合のフロー



1. EC2上の監視サーバーで障害を検知し、S3へトリガーファイルを格納
2. S3のイベント通知機能で、SQS_1にメッセージを送信
3. SQS_1をトリガーとして、Lambda_1を起動
4. Lambda_1がDynamoDB_1から連絡先を取得し、AmazonConnectを起動
5. Lambda_1がAmazonConnectを起動すると同時に、SQS_2へメッセージを送信
6. AmazonConnectがユーザーへ自動電話通知を実施
7. ユーザーが正常応答し、AmazonConnectがLambda_2を起動
8. Lambda_2が応答結果をDynamoDB_2に保存(応答OK)
9. 60秒後にSQS_2をトリガーとしてLambda_3を起動
10. Lambda_3がDynamoDB_2の応答結果を確認(正常応答しているため、何もせずに処理完了)

電話に応答しなかった場合のフロー



1. EC2上の監視サーバーで障害を検知し、S3へトリガーファイルを格納
2. S3のイベント通知機能で、SQS_1にメッセージを送信
3. SQS_1をトリガーとして、Lambda_1を起動
4. Lambda_1がDynamoDB_1から連絡先を取得し、AmazonConnectを起動
5. Lambda_1がAmazonConnectを起動すると同時に、SQS_2へメッセージを送信
6. AmazonConnectがユーザーへ自動電話通知を実施
7. ユーザーが正常応答せず、AmazonConnectがLambda_2を起動
8. Lambda_2が応答結果をDynamoDB_2に保存(応答NG)
9. 60秒後にSQS_2をトリガーとしてLambda_3を起動
10. Lambda_3がDynamoDB_2の応答結果を確認
11. 正常応答していないため、再度AmazonConnectを起動(以降、5から繰り返す)

周回数の指定

前回までの設定の場合、何らかの理由で正常応答されない場合、永遠に自動電話通知を繰り返してしまいます。ここでは、指定した周回数で自動電話通知が止まるようにLambdaのコードを修正します。

Lambda_1の修正

Lambda_1を下記の通り修正します。（最大周回数を2としています。）

```
import json
import boto3
```

```
from boto3.dynamodb.conditions import Key, Attr

# boto3からDynamoDBへアクセスするためのオブジェクトを取得
dynamodb = boto3.resource('dynamodb')

# "amazonconnect-contact-list"へアクセスするためのオブジェクトを取得
contactlist = dynamodb.Table("amazonconnect-contact-list")

# "amazonconnect-response-status"へアクセスするためのオブジェクトを取得
responsestatus = dynamodb.Table("amazonconnect-response-status")

# CallStatusを"NG"に変更する関数
def status_ng():
    response = responsestatus.update_item(
        Key={
            'No': 1,
            'Name': "Response"
        },
        UpdateExpression="set CallStatus=:c",
        ExpressionAttributeValues={
            ':c': "NG"
        },
        ReturnValues="UPDATED_NEW"
    )
    return response

# "amazonconnect-contact-list"の内容を返す関数
def operation_scan():

    scanData = contactlist.scan()
    items=scanData['Items']

    return scanData["Items"]

# 指定されたプライオリティの電話番号を返す関数
def get_phone_number(json_contactinfo, now_priority):

    for line in json_contactinfo:
```

```
    if line['Priority']==now_priority:
        phone_number = line['Phone']

    return phone_number
```

```
def lambda_handler(event, context):
```

```
    # 応答結果を初期化
    status_ng()
```

```
    # 電話番号リストを取得
    contactinfo = operation_scan()
```

```
    # 初回のプライオリティを"1"に設定
    priority = 1
```

```
    # 周回数を"2"に設定
    cycle = 2
```

```
    # 指定したプライオリティの電話番号を取得
    phone_number = get_phone_number(contactinfo, priority)
```

```
    # boto3からAmazonConnectへアクセスするためのオブジェクトを取得
    connect = boto3.client('connect')
```

```
    # AmazonConnectの問い合わせフローを呼び出し電話発信
    connect.start_outbound_voice_contact(
        DestinationPhoneNumber=phone_number,
        ContactFlowId='*****',
        InstanceId='*****',
        SourcePhoneNumber='+81*****',
    )
```

```
    # boto3からSQSへアクセスするためのオブジェクトを取得
    sqs = boto3.resource('sqs')
```

```
    # "amazonconnect-queue-confirm"へアクセスするためのオブジェクトを取得
    name = 'amazonconnect-queue-confirm'
    queue = sqs.get_queue_by_name(QueueName=name)
```

現在のプライオリティと周回数をセットしてステータス確認用のキューにメッセージ送信

```
response = queue.send_message(MessageBody=json.dumps({"priority": priority,
"cycle": cycle}))
```

ContactFlowId: 問い合わせフローID

InstanceId: インスタンスID

SourcePhoneNumber: 発信元の電話番号

※国番号をつけて記述（日本の050の番号の場合、+8150*****）

```

1 import json
2 import boto3
3 from boto3.dynamodb.conditions import Key, Attr
4
5 # boto3からDynamoDBへアクセスするためのオブジェクトを取得
6 dynamodb = boto3.resource('dynamodb')
7
8 # "amazonconnect-contact-list"へアクセスするためのオブジェクトを取得
9 contactlist = dynamodb.Table("amazonconnect-contact-list")
10
11 # "amazonconnect-response-status"へアクセスするためのオブジェクトを取得
12 responsestatus = dynamodb.Table("amazonconnect-response-status")
13
14 # CallStatusを"NG"に変更する関数
15 def status_ng():
16     response = responsestatus.update_item(
17         Key={
18             'No': 1,
19             'Name': "Response"
20         },
21         UpdateExpression="set CallStatus=:c",
22         ExpressionAttributeValues={
23             ':c': "NG"
24         },
25         ReturnValues="UPDATED_NEW"
26     )
27     return response
28
29
30 # "amazonconnect-contact-list"の内容を返す関数
31 def operation_scan():
32
33     scanData = contactlist.scan()
34     items=scanData['Items']
35
36     return scanData["Items"]
37
38
39 # 指定されたプライオリティの電話番号を返す関数
40 def get_phone_number(json_contactinfo, now_priority):
41
42     for line in json_contactinfo:
43         if line['Priority']==now_priority:
44             phone_number = line['Phone']
45
46     return phone_number
47
48
49 def lambda_handler(event, context):
50
51     # 応答結果を初期化
52     status_ng()
53
54     # 電話番号リストを取得
55     contactinfo = operation_scan()
56
57     # 初回のプライオリティを"1"に設定
58     priority = 1
59
60     # 周回数を"2"に設定
61     cycle = 2
62
63     # 指定したプライオリティの電話番号を取得
64     phone_number = get_phone_number(contactinfo, priority)
65
66     # boto3からAmazonConnectへアクセスするためのオブジェクトを取得
67     connect = boto3.client('connect')
68
69     # AmazonConnectの問い合わせフローを呼び出し電話発信
70     connect.start_outbound_voice_contact(
71         DestinationPhoneNumber=phone_number,
72         ContactFlowId='...',
73         InstanceId='...',
74         SourcePhoneNumber='+81...',
75     )
76
77     # boto3からSQSへアクセスするためのオブジェクトを取得
78     sqs = boto3.resource('sqs')
79
80     # "amazonconnect-queue-confirm"へアクセスするためのオブジェクトを取得
81     name = 'amazonconnect-queue-confirm'
82     queue = sqs.get_queue_by_name(QueueName=name)
83
84     # 現在のプライオリティと周回数をセットしてステータス確認用のキューにメッセージ送信
85     response = queue.send_message(MessageBody=json.dumps({"priority": priority, "cycle": cycle}))
86

```

Lambda_3の修正

Lambda_3を下記の通り修正します。

```
-----  
  
import json  
import boto3  
from boto3.dynamodb.conditions import Key, Attr  
  
# boto3からDynamoDBへアクセスするためのオブジェクトを取得  
dynamodb = boto3.resource('dynamodb')  
  
# "amazonconnect-contact-list"へアクセスするためのオブジェクトを取得  
contactlist = dynamodb.Table("amazonconnect-contact-list")  
  
# "amazonconnect-response-status"へアクセスするためのオブジェクトを取得  
responsestatus = dynamodb.Table("amazonconnect-response-status")  
  
# 残り周回数を取得する関数  
def get_cycle(sqs_event):  
  
    record = sqs_event['Records'][0]  
    json_record = json.loads(record["body"])  
    cycle = json_record["cycle"]  
  
    return cycle  
  
# "amazonconnect-response-status"の内容を返す関数  
def status_scan():  
  
    scanData = responsestatus.scan()  
    items=scanData['Items']  
  
    return scanData["Items"]  
  
# "amazonconnect-contact-list"の内容を返す関数  
def operation_scan():  
  
    scanData = contactlist.scan()  
    items=scanData['Items']  
  
    return scanData["Items"]
```


電話番号リストの行数を返す関数

```
def get_contactinfo_item_number(json_contactinfo):
```

```
    count = 0
```

```
    for line in json_contactinfo:
```

```
        count += 1
```

```
    return count
```

前回のプライオリティを返す関数

```
def get_before_priority(sqs_event):
```

```
    record = sqs_event['Records'][0]
```

```
    json_record = json.loads(record["body"])
```

```
    priority = json_record["priority"]
```

```
    return priority
```

指定されたプライオリティの電話番号を返す関数

```
def get_phone_number(json_contactinfo, now_priority):
```

```
    for line in json_contactinfo:
```

```
        if line['Priority']==now_priority:
```

```
            phone_number = line['Phone']
```

```
    return phone_number
```

```
def lambda_handler(event, context):
```

```
    # 残り周回数を取得
```

```
    now_cycle = get_cycle(event)
```

```
    # 応答結果を確認
```

```
    statustinfo = status_scan()
```

応答結果がNGの場合

```
if statustinfo[0]['CallStatus'] == "NG":
```

電話番号リストを取得

```
contactinfo = operation_scan()
```

電話番号リストの行数を取得

```
contactinfo_item_number = get_contactinfo_item_number(contactinfo)
```

前回のプライオリティを取得

```
before_priority = get_before_priority(event)
```

前回のプライオリティが電話番号リストの行数と同じ場合（リストの最後まで通知した場合）

```
if before_priority == contactinfo_item_number:
```

プライオリティを1に設定（最初に戻る）

```
priority = 1
```

残り周回数を1減らす

```
now_cycle -= 1
```

前回のプライオリティが電話番号リストの行数と異なる場合（リストの最後まで通知していない場合）

```
else:
```

プライオリティに1を足す（次のプライオリティを設定）

```
priority = before_priority + 1
```

残り周回数が0以外の場合

```
if now_cycle != 0:
```

指定したプライオリティの電話番号を取得

```
phone_number = get_phone_number(contactinfo, priority)
```

boto3からAmazonConnectへアクセスするためのオブジェクトを取得

```
connect = boto3.client('connect')
```

AmazonConnectの問い合わせフローを呼び出し電話発信

```
connect.start_outbound_voice_contact(
    DestinationPhoneNumber=phone_number,
    ContactFlowId='*****',
    InstanceId='*****',
    SourcePhoneNumber='+81*****',
)
```

```
# boto3からSQSへアクセスするためのオブジェクトを取得
sqs = boto3.resource('sqs')
```

```
# "amazonconnect-queue-confirm"へアクセスするためのオブジェクトを取得
name = 'amazonconnect-queue-confirm'
queue = sqs.get_queue_by_name(QueueName=name)
```

現在のプライオリティと周回数をセットしてステータス確認用のキューにメッセージ送信

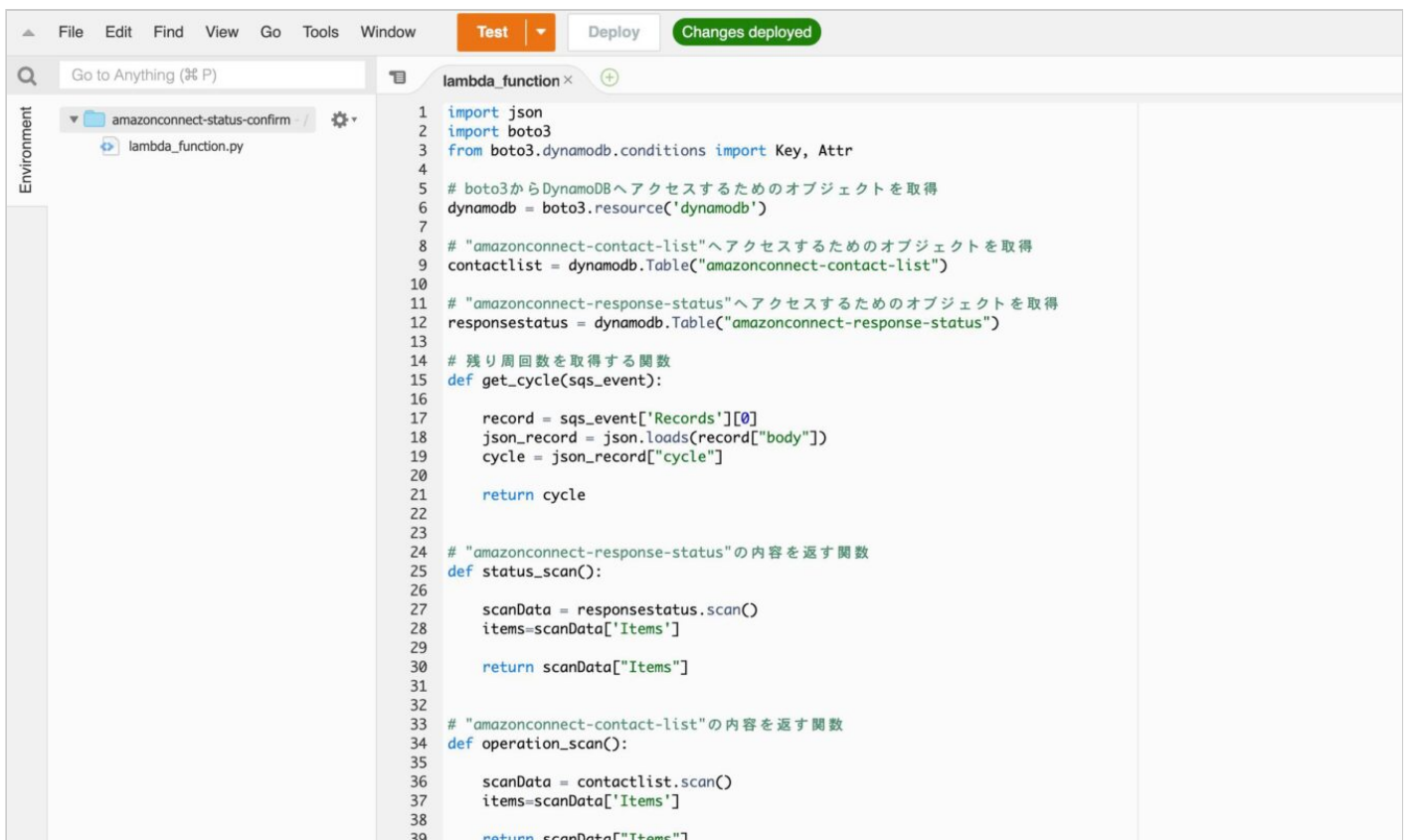
```
response = queue.send_message(MessageBody=json.dumps({"priority":
priority, "cycle": now_cycle}))
```

ContactFlowId: 問い合わせフローID

InstanceId: インスタンスID

SourcePhoneNumber: 発信元の電話番号

※国番号をつけて記述(日本の050の番号の場合、+8150*****)



```
1 import json
2 import boto3
3 from boto3.dynamodb.conditions import Key, Attr
4
5 # boto3からDynamoDBへアクセスするためのオブジェクトを取得
6 dynamodb = boto3.resource('dynamodb')
7
8 # "amazonconnect-contact-list"へアクセスするためのオブジェクトを取得
9 contactlist = dynamodb.Table("amazonconnect-contact-list")
10
11 # "amazonconnect-response-status"へアクセスするためのオブジェクトを取得
12 responsestatus = dynamodb.Table("amazonconnect-response-status")
13
14 # 残り周回数を取得する関数
15 def get_cycle(sqs_event):
16
17     record = sqs_event['Records'][0]
18     json_record = json.loads(record["body"])
19     cycle = json_record["cycle"]
20
21     return cycle
22
23
24 # "amazonconnect-response-status"の内容を返す関数
25 def status_scan():
26
27     scanData = responsestatus.scan()
28     items=scanData['Items']
29
30     return scanData["Items"]
31
32
33 # "amazonconnect-contact-list"の内容を返す関数
34 def operation_scan():
35
36     scanData = contactlist.scan()
37     items=scanData['Items']
38
39     return scanData["Items"]
```

```

40
41
42 # 電話番号リストの行数を返す関数
43 def get_contactinfo_item_number(json_contactinfo):
44
45     count = 0
46
47     for line in json_contactinfo:
48         count += 1
49
50     return count
51
52
53 # 前回のプライオリティを返す関数
54 def get_before_priority(sqs_event):
55
56     record = sqs_event['Records'][0]
57     json_record = json.loads(record['body'])
58     priority = json_record['priority']
59
60     return priority
61
62
63 # 指定されたプライオリティの電話番号を返す関数
64 def get_phone_number(json_contactinfo, now_priority):
65
66     for line in json_contactinfo:
67         if line['Priority']==now_priority:
68             phone_number = line['Phone']
69
70     return phone_number
71
72
73 def lambda_handler(event, context):
74
75     # 残り周回数を取得
76     now_cycle = get_cycle(event)
77
78     # 応答結果を確認
79     statustinfo = status_scan()
80
81     # 応答結果がNGの場合
82     if statustinfo[0]['CallStatus'] == "NG":
83
84         # 電話番号リストを取得
85         contactinfo = operation_scan()
86
87         # 電話番号リストの行数を取得
88         contactinfo_item_number = get_contactinfo_item_number(contactinfo)
89
90         # 前回のプライオリティを取得
91         before_priority = get_before_priority(event)
92
93         # 前回のプライオリティが電話番号リストの行数と同じ場合(リストの最後まで通知した場合)
94         if before_priority == contactinfo_item_number:
95
96             # プライオリティを1に設定(最初に戻る)
97             priority = 1
98
99             # 残り周回数を1減らす
100             now_cycle -= 1
101
102         # 前回のプライオリティが電話番号リストの行数と異なる場合(リストの最後まで通知していない場合)
103         else:
104
105             # プライオリティに1を足す(次のプライオリティを設定)
106             priority = before_priority + 1
107
108         # 残り周回数が0以外の場合
109         if now_cycle != 0:
110
111             # 指定したプライオリティの電話番号を取得
112             phone_number = get_phone_number(contactinfo, priority)
113
114             # boto3からAmazonConnectへアクセスするためのオブジェクトを取得
115             connect = boto3.client('connect')
116
117             # AmazonConnectの問い合わせフローを呼び出し電話発信
118             connect.start_outbound_voice_contact(
119                 DestinationPhoneNumber=phone_number,
120                 ContactFlowId=' ',
121                 InstanceId=' ',
122                 SourcePhoneNumber='+81 ',
123             )
124
125             # boto3からSQSへアクセスするためのオブジェクトを取得
126             sqs = boto3.resource('sqs')
127
128             # "amazonconnect-queue-confirm"へアクセスするためのオブジェクトを取得
129             name = 'amazonconnect-queue-confirm'
130             queue = sqs.get_queue_by_name(QueueName=name)
131
132             # 現在のプライオリティと周回数をセットしてステータス確認用のキューにメッセージ送信
133             response = queue.send_message(MessageBody=json.dumps({"priority": priority, "cycle": now_cycle}))
134

```

テスト

EC2から下記のコマンドを実施し、S3へファイルをアップロードします。※黄色アンダーライン箇所は、作成したS3バケット名を指定してください。

```
touch /tmp/test.txt  
aws s3 cp /tmp/test.txt s3://amazonconnect-alert-notification-bucket
```

```
[ec2-user@ip-10-0-0-100 ~]$ touch /tmp/test.txt  
[ec2-user@ip-10-0-0-100 ~]$  
[ec2-user@ip-10-0-0-100 ~]$ aws s3 cp /tmp/test.txt s3://amazonconnect-alert-  
notification-bucket  
upload: ../../tmp/test.txt to s3://amazonconnect-alert-notification-  
bucket/test.txt  
[ec2-user@ip-10-0-0-100 ~]$
```

以下の観点で確認を行います。

- 指定された周回数で自動電話通知が終了すること。（今回の場合、2回の周回で通知が終了すること。）

以上で、AmazonConnectによる自動電話通知（8.周回数の指定）の説明は完了です ！