デベロッパーガイド

Amazon Simple Workflow Service

API バージョン 2012-01-25

Amazon Simple Workflow Service: デベロッパーガイド

Table of Contents

Amazon Simple Workflow Service とは？

Amazon Simple Workflow Service (Amazon SWF) を使用すると、並列またはシーケンシャルステップを持つバックグラウンドジョブを構築、実行、スケーリングできます。分散コンポーネント間で作業を調整し、タスクの状態を追跡できます。

Amazon SWF では、タスクはアプリケーションのコンポーネントによって実行される作業の論理単位を表します。間のタスクの調整には、アプリケーションのフローにおけるタスク間の依存関係、スケジューリング、同時実行の管理が含まれます。Amazon SWF を使用すると、進行状況の追跡やタスクの状態の維持など、根本的な複雑さを気にすることなく、タスクを制御および調整できます。

Amazon SWF を使用する場合は、ワーカーを実装してタスクを実行します。ワーカーは、Amazon Elastic Compute Cloud (Amazon EC2) などのクラウドインフラストラクチャまたは独自のオンプレミスで実行できます。長時間実行されるか、失敗、タイムアウト、再起動を必要とする可能性があり、またはさまざまなスループットとレイテンシーで完了する可能性のあるタスクを作成できます。Amazon SWF はタスクを保存し、準備ができたらワーカーに割り当て、進行状況を追跡し、タスクの完了の詳細を含む状態を維持します。

タスクを調整するには、Amazon SWF から最新のタスク状態を取得し、その状態を使用して後続のタスクを開始するプログラムを作成します。Amazon SWF はアプリケーションの実行状態を永続的に維持するので、アプリケーションは個々のコンポーネントの障害に対して回復力があります。Amazon SWF を使用すると、アプリケーションコンポーネントを個別に構築、デプロイ、スケーリング、変更できます。

> **その他の AWS ワークフローサービス**
>
> ほとんどのユースケースでは、ワークフローとオーケストレーションのニーズ AWS Step Functions を考慮して検討することをお勧めします。
> Step Functions を使用すると、ステートマシンとも呼ばれるワークフローを作成して、分散アプリケーションの構築、プロセスの自動化、マイクロサービスのオーケストレーション、データと機械学習パイプラインの作成を行うことができます。VS Code の Step Functions のコンソールまたは AWS ツールキットでは、グラフィカルな Workflow Studio を使用して、アプリケーションのワークフローを視覚化、編集、テスト、デバッグできます。
> 技術的な詳細については、「AWS Step Functions デベロッパーガイド」を参照してください。

Amazon SWF を使用したワークフローコンポーネントの開発

分散アプリケーションを開発するには、多くのコンポーネントを調整し、リモート通信に固有のレイテンシーと信頼性の低さに対処する必要があります。

Amazon Simple Workflow Service (Amazon SWF) を使用すると、分散コンポーネントを調整し、信頼性の高い方法で実行状態を維持するためのプログラミングモデルとインフラストラクチャを提供することで、非同期および分散アプリケーションを開発できます。Amazon SWF に頼ることで、アプリケーションを差別化する側面の構築に集中することができます。

ワークフローのコンポーネント

ワークフローのコンポーネント Amazon SWF の基本的な概念はワークフローです。ワークフローは、目標を実現する一連のアクティビティ、および、そのアクティビティを調整するロジックです。たとえば、ワークフローは顧客の注文を受け取り、注文を満たすために必要なアクションを実行できます。

各ワークフローは、ワークフローの範囲を制御するドメインと呼ばれるリソースで実行されます。AWS アカウントは複数のドメインを持つことができ、それぞれに複数のワークフローを含めることができますが、異なるドメインのワークフローは連動できません。

Amazon SWF ワークフローを設計するときは、必要な各アクティビティを定義します。それから、Amazon SWF でアクティビティタイプとして各アクティビティを登録します。名前、バージョン、タイムアウト値を指定します。たとえば、注文は 24 時間以内に発送されると顧客が期待する場合があります。

ワークフロー実行のプロセス中、いくつかのアクティビティは、おそらく入力内容を変えて、何度も実行する必要があります。たとえば、顧客注文ワークフローで、購入された項目を処理するアクティビティがあります。顧客が複数の項目を購入すると、このアクティビティは複数回実行する必要があります。Amazon SWF には、アクティビティの 1 回の呼び出しを表すアクティビティタスクの概念があります。この例では、各項目の処理は 1 つのアクティビティタスクによって表されます。

アクティビティワーカーは、アクティビティタスクを受信して実行し、結果を提供するプログラムです。タスクは実際には人が実行します。たとえば、統計アナリストは一連のデータを受け取り、データを分析してから、分析を送り返すことができます。

アクティビティタスク、およびそれらを実行するアクティビティワーカーは、同期的に実行することも非同期的に実行することもできます。ワーカーは、1 つの場所で実行することも、複数のコンピュータに分散することもでき、地理的に異なるリージョンに分散することもできます。異なるアクティビティワーカーはさまざまなプログラミング言語で作成されており、さまざまなオペレーティングシステムで実行できます。たとえば、あるアクティビティワーカーはアジアのサーバーで実行され、別のアクティビティワーカーは北米のモバイルデバイスで実行されている可能性があります。

ソフトウェアプログラムに含まれるワークフローの調整ロジックはディサイダーと呼ばれます。ディサイダーは、アクティビティタスクをスケジュールし、アクティビティワーカーに入力を提供し、ワークフローの進行中に到着したイベントを処理し、目標が達成された後にワークフローを終了 (または終了) します。

Amazon SWF サービスのロールは、ディサイダー、アクティビティワーカー、およびワークフローの管理者などの他の関連エンティティ間でデータを交換するための信頼性のある集中ハブとして機能することです。Amazon SWF では、各ワークフロー実行の状態も維持されるため、アプリケーションが永続的に状態を保存する必要がなくなります。

ディサイダーは、Amazon SWF からの決定タスクの受け取り、および、Amazon SWF への決定の返信により、ワークフローを指示します。決定は、ワークフローの次のステップであるアクションまたは一連のアクションを表します。一般的な決定は、アクティビティタスクをスケジューリングすることです。決定は、タイマーを使用したタスクの遅延、進行中のタスクのキャンセルのリクエスト、ワークフローの完了にも使用できます。

アクティビティワーカーとディサイダーの両方がタスク (アクティビティタスクおよびディシジョンタスクのそれぞれ) を受け取るメカニズムは、Amazon SWF サービスのポーリングによります。

Amazon SWF は、各ディシジョンタスクに現在のワークフローの実行履歴のコピーを各ディシジョンタクスに含めることにより、ワークフローの状態をディサイダーに通知します。ワークフローの実行履歴はイベントで構成されていて、イベントはワークフローの実行状態の重要な変更を表します。イベントの例としては、タスクの完了、タスクのタイムアウト、タイマーの有効期限などがあります。履歴は、完全で、一貫性があり、信頼性のあるワークフローの進行の記録です。

Amazon SWF アクセスコントロールは AWS Identity and Access Management (IAM) を使用するため、 AWS リソースへのアクセスを制御できます。たとえば、ユーザーがアカウントへアクセスする際、特定のドメインでの特定のワークフローを実行することのみを許可することができます。

ワークフローの実行

Amazon SWF でワークフローを開発して実行するために必要なステップの概要を以下に示します。

1. ワークフローで処理ステップを実行するアクティビティワーカーを記述します。
2. ワークフローの調整ロジックを処理するディサイダーを作成します。
3. アクティビティとワークフローを Amazon SWF で登録します。

   このステップは、プログラムまたは を使用して実行できます AWS マネジメントコンソール。
4. アクティビティワーカーとディサイダーを開始します。

   これらのアクターは Amazon SWF エンドポイントにアクセスできる任意のコンピューティングデバイスから実行できます。例えば、Amazon Elastic Compute Cloud (Amazon EC2)、データセンターのサーバー、さらにはモバイルデバイスなどといった、クラウド内のコンピューティングインスタンスを使用して、ディサイダーまたはアクティビティワーカーをホストすることができます。起動すると、ディサイダーとアクティビティワーカーはタスクのために Amazon SWF のポーリングを開始します。
5. ワークフローの実行を 1 つ以上開始します。

   ワークフローは、プログラムまたは を介して開始できます AWS マネジメントコンソール。

   各実行は独立して動作し、それぞれに独自の入力データのセットを提供できます。実行が開始すると、Amazon SWF は最初の決定タスクをスケジュールします。これに応じて、ディサイダーはアクティビティタスクを開始する決定の生成を開始します。ディサイダーが実行を閉じることを決めるまで実行は継続します。
6. を使用してワークフロー実行を表示します AWS マネジメントコンソール。

   実行中および完了した実行の完全な詳細をフィルタリングして表示できます。たとえば、開いている実行を選択して、完了したタスクとその結果を確認できます。

開発環境のセットアップ

でサポートされている任意のプログラミング言語で Amazon SWF 用の を開発するオプションがあります AWS。Java 開発者 AWS Flow Framework には、 も利用できます。詳細については、「AWS Flow Frameworkウェブサイト」およびAWS Flow Framework 「for Java デベロッパーガイド」を参照してください。

レイテンシーを短縮し、要件を満たす場所にデータを保存するために、Amazon SWF はさまざまなリージョンにエンドポイントを提供します。

Amazon SWF の各エンドポイントは完全に独立しています。あるリージョンに登録したドメイン、ワークフロー、アクティビティは、別のリージョンのデータや属性と共有されません。

Amazon SWF ドメイン、ワークフロー、またはアクティビティを登録すると、登録したリージョン内にのみ存在します。たとえば、2 つの異なるリージョンSWF-Flows-1に という名前のドメインを登録できますが、それぞれが完全に独立したドメインとして機能するデータや属性は相互に共有されません。

Amazon SWF エンドポイントのリストについては、「Regions and Endpoints」(リージョンとエンドポイント) を参照してください。

AWS SDKs

Amazon SWF は、 AWS SDKs Java、.NET、Node.js、PHP、Python、Ruby でサポートされており、選択したプログラミング言語で Amazon SWF HTTP API を使用する便利な方法を提供します。

これらのライブラリで公開されている API を使用して、ディサイダー、アクティビティワーカー、またはワークフロースターターを開発できます。また、これらのライブラリを通じて可視性オペレーションを使用して、独自の Amazon SWF モニタリングおよびレポートツールを開発できます。

SDKs など AWS、 でアプリケーションを開発および管理するためのツールをダウンロードするには、 デベロッパーセンターにアクセスしてください。

各 SDK での Amazon SWF オペレーションの詳細については、SDK の言語固有のリファレンスドキュメントを参照してください。

を検討する AWS Flow Framework

AWS Flow Framework は、Amazon SWF でワークフローとして実行される分散非同期プログラムを記述するための拡張 SDK です。フレームワークは Java プログラミング言語で使用でき、複雑な分散プログラムを記述するためのクラスを提供します。

では AWS Flow Framework、事前設定されたタイプを使用して、ワークフローの定義をプログラム内のメソッドに直接マッピングします。は、例外ベースのエラー処理など、標準のオブジェクト指向の概念 AWS Flow Framework をサポートしています。で記述されたプログラムは、任意のエディタまたは IDE 内で完全に作成、実行、デバッグ AWS Flow Framework できます。詳細については、「 AWS Flow Frameworkウェブサイト」およびAWS Flow Framework 「 for Java デベロッパーガイド」を参照してください。

Amazon SWF の開始方法

シーケンシャルに動作する 4 つのアクティビティのセットで構成される次の Amazon Simple Workflow Service ワークフローアプリケーションを開始できます。このチュートリアルでは、以下のトピックについても説明します。

- デフォルトおよび実行時間のワークフローとアクティビティオプションの設定。
- 決定およびアクティビティタスクに関する Amazon SWF のポーリング。
- Amazon SWF を使用した、アクティビティとワークフロー間のデータの引き渡し。
- ヒューマンタスクを待機し、アクティビティタスクから Amazon SWF にハートビートを報告します。
- Amazon SNS を使用したトピックの作成、そのトピックへのユーザーのサブスクライブ、およびサブスクライブしたエンドポイントへのメッセージの発行。

Amazon SWF と Amazon Simple Notification Service (Amazon SNS) を組み合わせて「人間のタスク」ワークフローをエミュレートできます。このワークフローでは、人間のワーカーが何らかのアクションを実行し、Amazon SWF と通信してワークフローで次のアクティビティを起動する必要があります。

Amazon SWF はクラウドベースのウェブサービスであるため、Amazon SWF との通信は、インターネットへ接続が利用できる場所であればどこからでも行うことができます。この場合は、Amazon SNS を使用して、E メール、SMS テキストメッセージ、またはその両方によりユーザーと通信します。

このチュートリアルではAWS SDK for Ruby、 を使用して Amazon SWF と Amazon SNS にアクセスしますが、Amazon SWF との調整と通信を容易にする AWS Flow Framework for Ruby など、多くの開発オプションを利用できます。

> **Note**
>
> このチュートリアルでは を使用しますが AWS SDK for Ruby、 AWS Flow Framework for Java を使用することをお勧めします。

トピック

ワークフローについて

開発するワークフローは、4 つの主要なステップで構成されます。

1. ユーザーからサブスクリプションアドレス (E メールまたは SMS) を取得します。
2. SNS トピックを作成し、トピックに対して提供されたエンドポイントにサブスクライブします。
3. ユーザーによるサブスクリプションの確認を待機します。
4. ユーザーが確認した場合、トピックに対して成功のメッセージを発行します。

これらのステップには、完全に自動化されたアクティビティ (ステップ 2 および 4) と、ワークフローが進行するためには人間によるアクティビティへのデータの提供をワークフローが待機する必要があるその他のアクティビティ (ステップ 1 および 3) が含まれます。

各ステップは前のステップで生成されたデータに基づいていて、トピックにサブスクライブする前にエンドポイントが必要です。また、確認などを待機する前にトピックのサブスクリプションが必要です。このチュートリアルでは、完了時にアクティビティの結果を提供する方法、およびスケジュールされているタスクに入力を渡す方法についても説明します。Amazon SWF は、アクティビティとワークフローの間の情報の調整と配信、およびその逆を行います。

また、キーボード入力と Amazon SNS の両方を使用して、Amazon SWF とワークフローにデータを提供している人間との間の通信を処理しています。実際には、多くの異なる手法を使用して人間のユーザーと通信できますが、Amazon SNS はワークフローのイベントについてユーザーに通知するために、E メールまたはテキストメッセージを使用するための非常に簡単な方法を提供します。

前提条件

このチュートリアルを実行するには、以下が必要です。

- Amazon Web Services アカウント
- Ruby インタープリタ
- AWS SDK for Ruby

既にこれらをセットアップしている場合、続行する準備ができています。この例を実行しない場合は、チュートリアルに従うことができます。このチュートリアルの内容の多くは、選択した開発オプションに関係なくAmazon SWF と Amazon SNS の使用に適用されます。

チュートリアルのステップ

このチュートリアルは、次のステップで構成されます。

1. サブスクリプションワークフローチュートリアルパート 1: での Amazon SWF の使用 AWS SDK for Ruby
2. サブスクリプションワークフローのチュートリアルのパート 2: ワークフローの実装
3. サブスクリプションワークフローのチュートリアルのパート 3: アクティビティの実装
4. サブスクリプションワークフローのチュートリアルのパート 4: アクティビティタスクポーラーの実装
5. サブスクリプションワークフローのチュートリアル: ワークフローの実行

サブスクリプションワークフローチュートリアルパート 1: での Amazon SWF の使用 AWS SDK for Ruby

トピック

- を含める AWS SDK for Ruby
- AWS セッションの設定
- Amazon SWF ドメインの登録
- 次のステップ

を含める AWS SDK for Ruby

utils.rb というファイルを作成して開始します。このファイルのコードは、ワークフローコードとアクティビティコードの両方で使用される Amazon SWF ドメインを取得または作成し、すべてのクラスに共通のコードを配置する場所を提供します。

最初に、aws-sdk-v1 ライブラリをコードに含める必要があります。これにより、SDK for Ruby が提供する機能を使用できるようになります。

```
require 'aws-sdk-v1'
```

これにより、 AWS 名前空間にアクセスできます。これにより、 AWS 認証情報やリージョンなどのグローバルセッション関連の値を設定したり、 AWS サービス APIs にアクセスしたりできます。

AWS セッションの設定

AWS セッションを設定するには、 AWS 認証情報 (AWS サービスへのアクセスに必要) と使用する AWS リージョンを設定します。

AWS SDK for Ruby で AWS 認証情報を設定するには、環境変数 (AWS_ACCESS_KEY_ID および AWS_SECRET_ACCESS_KEY) で設定するか、 で設定する方法がいくつかありますAWS.config。後者の方法を使用して、aws-config.txt という YAML 設定ファイルからロードします。これは次のようになります。

```
---
:access_key_id: REPLACE_WITH_ACCESS_KEY_ID
:secret_access_key: REPLACE_WITH_SECRET_ACCESS_KEY
```

このファイルを今すぐ作成し、REPLACE_WITH_ で始まる文字列を AWS アクセスキー ID とシークレットアクセスキーに置き換えます。 AWS アクセスキーの詳細については、Amazon Web Services 全般のリファレンスの「セキュリティ認証情報の取得方法」を参照してください。

また、使用する AWS リージョンを設定する必要があります。Short Message Service (SMS) を使用して Amazon SNS でユーザーの電話にテキストメッセージを送信するため、Amazon SNS でサポートされているリージョンを使用していることを確認する必要があります。「Amazon Simple Notification Service 開発者ガイド」の「サポートされているリージョンと国」を参照してください。

> **Note**
>
> us-east-1 にアクセスできない場合、または SMS メッセージを有効にしてデモを実行しない場合は、希望するリージョンを自由に使用してください。サンプルから SMS 機能を削除し、Amazon SNS トピックに登録する唯一のエンドポイントとして E メールを使用することができます。
> SMS メッセージの送信の詳細については、Amazon 「Simple Notification Service 開発者ガイド」の「Amazon Simple Notification Service Developer Guide」 (Amazon SNS を使用した SMS 通知の送受信) を参照してください。

`utils.rb` にいくつかのコードを追加して Config ファイルをロードし、ユーザーの認証情報を取得し、認証情報とリージョンの両方を `AWS.config` に提供します。

```
require 'yaml'

# Load the user's credentials from a file, if it exists.
begin
  config_file = File.open('aws-config.txt') { |f| f.read }
rescue
  puts "No config file! Hope you set your AWS credentials in the environment..."
end

if config_file.nil?
  options = { }
else
  options = YAML.load(config_file)
end

# SMS Messaging (which can be used by Amazon SNS) is available only in the
# `us-east-1` region.
$SMS_REGION = 'us-east-1'
options[:region] = $SMS_REGION

# Now, set the options
AWS.config = options
```

Amazon SWF ドメインの登録

Amazon SWF を使用するには、ワークフローとアクティビティを保持する名前の付いたエンティティである ドメイン を設定する必要があります。多くの Amazon SWF ドメインを登録できますが、それらはすべて AWS アカウント内で一意の名前を持つ必要があり、ワークフローはドメイン間でやり取りできません。アプリケーションのすべてのワークフローとアクティビティは、相互にやり取りするために同じドメインに存在する必要があります。

アプリケーション全体で同じドメインを使用するため、`utils.rb` に `init_domain` という関数を作成し、SWFSampleDomain という名前の Amazon SWF ドメインを取得します。

ドメインを登録したら、それを多くのワークフローの実行に再利用できます。ただし、すでに存在するドメインを登録しようとするとエラーになるので、コードはまずドメインが存在するかどうかを確認し、見つかった場合は既存のドメインを使用します。ドメインが見つからない場合は作成します。

SDK forRuby で Amazon SWF ドメインを操作するには、AWS::SimpleWorkflow.domains を使用します。これにより、ドメインの列挙と登録の両方に使用できる DomainCollection が返されます。

- ドメインがすでに登録されているかどうかを確認するには、AWS::Simpleworkflow.domains.registered で提供されているリストを参照してください。
- 新しいドメインを登録するには、AWS::Simpleworkflow.domains.register を使用します。

`utils.rb` の `init_domain` のコードは以下のとおりです。

```
# Registers the domain that the workflow will run in.
def init_domain
  domain_name = 'SWFSampleDomain'
  domain = nil
  swf = AWS::SimpleWorkflow.new

  # First, check to see if the domain already exists and is registered.
  swf.domains.registered.each do | d |
    if(d.name == domain_name)
      domain = d
      break
    end
  end

  if domain.nil?
    # Register the domain for one day.
```

```
    domain = swf.domains.create(
      domain_name, 1, { :description => "#{domain_name} domain" })
  end

  return domain
end
```

次のステップ

次に、サブスクリプションワークフローのチュートリアルのパート 2: ワークフローの実装 でワークフローとスターターコードを作成します。

サブスクリプションワークフローのチュートリアルのパート 2: ワークフローの実装

これまでは、コードはかなり汎用的でした。この部分で、ワークフローが何を行うか、またそれを実装するのにどのようなアクティビティが必要になるかを実際に定義し始めます。

トピック

ワークフローの設計

思い返して頂くと、このワークフローの最初のアイデアは以下のステップで構成されていました。

1. ユーザーからサブスクリプションアドレス (E メールまたは SMS) を取得します。
2. SNS トピックを作成し、トピックに対して提供されたエンドポイントにサブスクライブします。
3. ユーザーによるサブスクリプションの確認を待機します。
4. ユーザーが確認した場合、トピックに対して成功のメッセージを発行します。

ワークフローの各ステップのことを、実行する必要のあるアクティビティと考えることができます。ワークフローは、適切なタイミングで各アクティビティをスケジュールし、アクティビティ間のデータ転送を調整します。

このワークフローでは、これらのステップの各々に別個のアクティビティを作成し、それらに記述的に名前を付けます。

1. get_contact_activity
2. subscribe_topic_activity
3. wait_for_confirmation_activity
4. send_result_activity

これらのアクティビティは順番に実行され、各ステップからのデータは、次のステップで使用されます。

コードすべてが 1 つのソースファイルに存在するようにアプリケーションを設計できますが、これは Amazon SWF の設計された方法に反しています。範囲がインターネット全体に及ぶワークフローのために設計されているので、少なくともアプリケーションを 2 つの別個の実行可能ファイルに分割しましょう。

- `swf_sns_workflow.rb` - ワークフローおよびワークフロースターターが含まれています。
- `swf_sns_activities.rb` - アクティビティおよびアクティビティスターターが含まれています。

ワークフローとアクティビティの実装は、別々のウィンドウ、別々のコンピュータ、または世界の異なる場所でも実行できます。Amazon SWF がワークフローとアクティビティの詳細を追跡しているため、ワークフローは、実行中の場所に関係なくアクティビティのスケジューリングとデータ転送を調整できます。

ワークフローコードの設定

`swf_sns_workflow.rb` というファイルを作成して開始します。このファイルでは、SampleWorkflow というクラスを宣言します。クラスの宣言とそのコンストラクタである `initialize` メソッドを次に示します。

```
require_relative 'utils.rb'

# SampleWorkflow - the main workflow for the SWF/SNS Sample
```

```
#
# See the file called `README.md` for a description of what this file does.
class SampleWorkflow

  attr_accessor :name

  def initialize(workflowId)

    # the domain to look for decision tasks in.
    @domain = init_domain

    # the task list is used to poll for decision tasks.
    @workflowId = workflowId

    # The list of activities to run, in order. These name/version hashes can be
    # passed directly to AWS::SimpleWorkflow::DecisionTask#schedule_activity_task.
    @activity_list = [
      { :name => 'get_contact_activity', :version => 'v1' },
      { :name => 'subscribe_topic_activity', :version => 'v1' },
      { :name => 'wait_for_confirmation_activity', :version => 'v1' },
      { :name => 'send_result_activity', :version => 'v1' },
    ].reverse! # reverse the order... we're treating this like a stack.

    register_workflow
  end
```

ご覧のとおり、次のクラスインスタンスデータを保持しています。

- domain - utils.rb の init_domain から取得したドメイン名。
- workflowId - initialize に渡されるタスクリスト。
- activity_list - 実行するアクティビティの名前とバージョンを持つアクティビティリスト。

Amazon SWF がアクティビティタイプを確実に識別するには、ドメイン名、アクティビティ名、アクティビティバージョンで十分であるため、スケジュールするためにアクティビティについて保持する必要があるデータはそれだけです。

タスクリストは、決定タスクをポーリングしアクティビティをスケジュールするために、ワークフローのディサイダーコードによって使用されます。

この関数の最後に、まだ定義していないメソッドである register_workflow を呼び出します。次にこのメソッドを定義します。

ワークフローの登録

ワークフロータイプを使用するには、まずそれを登録する必要があります。アクティビティタイプと同様、ワークフロータイプは、そのドメイン、名前、およびバージョンによって識別されます。また、ドメインもアクティビティタイプも同様ですが、既存のワークフロータイプを再登録することはできません。ワークフロータイプについて何か変更する必要がある場合は、それに新しいバージョンを提供する必要がありますが、それにより本質的に新しいタイプが作成されます。

register_workflow のコードは次のとおりです。以前の実行時に登録した既存のワークフロータイプを取得するため、またはまだ登録されていない場合はそのワークフローを登録するために使用されます。

```
 # Registers the workflow
 def register_workflow
   workflow_name = 'swf-sns-workflow'
   @workflow_type = nil

   # a default value...
   workflow_version = '1'

   # Check to see if this workflow type already exists. If so, use it.
   @domain.workflow_types.each do | a |
     if (a.name == workflow_name) && (a.version == workflow_version)
       @workflow_type = a
     end
   end

   if @workflow_type.nil?
     options =  {
       :default_child_policy => :terminate,
       :default_task_start_to_close_timeout => 3600,
       :default_execution_start_to_close_timeout => 24 * 3600 }

     puts "registering workflow: #{workflow_name}, #{workflow_version},
#{options.inspect}"
     @workflow_type = @domain.workflow_types.register(workflow_name, workflow_version,
options)
   end

   puts "** registered workflow: #{workflow_name}"
 end
```

まず、ドメインの workflow_types コレクションを反復処理することにより、ワークフロー名とバージョンが既に登録されているかどうか確認します。一致が検出されると、既に登録されていたワークフロータイプを使用します。

一致するものが見つからない場合、(ワークフローを検索したのと同じ workflow_types コレクションで register を呼び出すことにより) 名前を「swf-sns-workflow」、バージョンを「1」、オプションを次のようにした新しいワークフロータイプが登録されます。

```
    options =  {
      :default_child_policy => :terminate,
      :default_task_start_to_close_timeout => 3600,
      :default_execution_start_to_close_timeout => 24 * 3600 }
```

登録中に渡されるオプションは、ワークフロータイプのデフォルトの動作を設定するために使用されるので、新しいワークフロー実行を開始するたびにこれらの値を設定する必要はありません。

ここでは、タスクの開始から終了までの最大時間 (1 時間)、およびワークフロー実行が完了するまでにかかる最大時間 (24 時間) というタイムアウト値だけを設定します。これらの時間のどちらかを超過すると、タスクまたはワークフローがタイムアウトします。

タイムアウト値の詳細については、「Amazon SWF タイムアウトの種類」を参照してください。

決定のポーリング

すべてのワークフロー実行の中心には、ディサイダーがあります。ディサイダーには、ワークフロー自体の実行を管理する責任があります。ディサイダーは決定タスクを取得し、新しいアクティビティのスケジューリングやアクティビティのキャンセルや再開を行うか、ワークフロー実行の状態を完了、キャンセル済み、または失敗と設定するかのどちらかでそれに応答します。

ディサイダーはワークフロー実行のタスクリスト名を使用して、応答する決定タスクを取得します。決定タスクをポーリングするには、ドメインの decision_tasks コレクションで poll を呼び出して、使用可能な決定タスクをループします。その後、new_events コレクションに対して反復処理することにより、その決定タスクに新しいイベントがないか確認できます。

返されるイベントは AWS::SimpleWorkflow::HistoryEvent オブジェクトで、その返されたイベントの event_type メンバーを使用することによって、イベントのタイプを取得できます。履歴イベントタイプのリストと説明については、Amazon Simple Workflow Service API リファレンス の「HistoryEvent」を参照してください。

以下に決定タスクポーラーのロジックの冒頭を示します。poll_for_decisions というワークフロークラスの新しいメソッドです。

```
def poll_for_decisions
  # first, poll for decision tasks...
  @domain.decision_tasks.poll(@workflowId) do | task |
    task.new_events.each do | event |
      case event.event_type
```

次に、取得する event_type に基づいてディサイダーの実行を分岐します。取得する可能性が高い最初のものは、WorkflowExecutionStarted です。このイベントを取得した場合、それは Amazon SWF がディサイダーにワークフロー実行を開始するよう合図していることを意味します。まず初めに、ポーリング中に取得したタスクで schedule_activity_task を呼び出すことで、最初のアクティビティをスケジューリングします。

それにアクティビティリストで宣言した最初のアクティビティを渡します。これは、スタックのように使用できるようリストを逆にしたために、リストの last 位置にあります。定義した「アクティビティ」は名前とバージョン番号で構成されたマップにすぎませんが、Amazon SWF がアクティビティは登録済みであると想定してスケジューリング用のアクティビティを識別するために必要とするのはこれだけです。

```
        when 'WorkflowExecutionStarted'
          # schedule the last activity on the (reversed, remember?) list to
          # begin the workflow.
          puts "** scheduling activity task: #{@activity_list.last[:name]}"

          task.schedule_activity_task( @activity_list.last,
            { :workflowId => "#{@workflowId}-activities" } )
```

アクティビティをスケジュールすると、Amazon SWF はスケジューリング中に渡されるアクティビティタスクリストに アクティビティタスク を送り、開始すべきタスクを合図します。「サブスクリプションワークフローのチュートリアルのパート 3: アクティビティの実装」でアクティビティタスクを扱いますが、ここでそのタスクを実行しないことは注目に値します。Amazon SWF にはそれを scheduled (スケジュールする) ようにとだけ指示します。

取り組む必要のある次のアクティビティは ActivityTaskCompleted イベントで、これは Amazon SWF がアクティビティタスクからアクティビティ完了の応答を取得したときに発生します。

```
        when 'ActivityTaskCompleted'
          # we are running the activities in strict sequential order, and
          # using the results of the previous activity as input for the next
          # activity.
          last_activity = @activity_list.pop
```

```
            if(@activity_list.empty?)
              puts "!! All activities complete! Sending complete_workflow_execution..."
              task.complete_workflow_execution
              return true;
            else
              # schedule the next activity, passing any results from the
              # previous activity. Results will be received in the activity
              # task.
              puts "** scheduling activity task: #{@activity_list.last[:name]}"
              if event.attributes.has_key?('result')
                task.schedule_activity_task(
                  @activity_list.last,
                  { :input => event.attributes[:result],
                    :workflowId => "#{@workflowId}-activities" } )
              else
                task.schedule_activity_task(
                  @activity_list.last, { :workflowId => "#{@workflowId}-activities" } )
              end
            end
```

タスクは直線的に実行されており、一度に実行されているアクティビティは 1 つだけであるため、この機会にactivity_listスタックから完了したタスクをポップします。この結果が空のリストなら、ワークフローが完了していることが分かります。この場合、タスクで complete_workflow_execution を呼び出すことにより、ワークフローが完了したことを Amazon SWF に通知します。

リストにまだエントリがあるイベントでは、リストで (この場合もやはり last 位置にある) 次のアクティビティをスケジュールします。ただし今回は、完了時に前のアクティビティが Amazon SWF に結果データを返したかどうかを確認します。そのデータは、イベントの属性、オプションの `result` キーでワークフローに提供されます。アクティビティが結果を生成した場合は、それを `input` オプションとして、アクティビティタスクリストと共に次のスケジュールされたアクティビティに渡します。

完了済みアクティビティの `result` 値を取得することにより、またスケジュールされたアクティビティの `input` 値を設定することにより、1 つのアクティビティから次のアクティビティへとデータを渡したり、アクティビティから取得したデータを使用して、アクティビティの結果に基づいてディサイダー内の動作を変更することができます。

このチュートリアルでは、これら 2 つのイベントタイプがワークフローの動作を定義するのに最も重要です。ただし、アクティビティは ActivityTaskCompleted 以外のイベントを生成することがで

きます。ActivityTaskTimedOut イベントと ActivityTaskFailed イベント、および実行するアクティビティが不足したときに Amazon SWF が行う `complete_workflow_execution` 呼び出しの処理時に生成される WorkflowExecutionCompleted イベントのデモハンドラコードを提供して、ディサイダーコードをまとめます。

```
          when 'ActivityTaskTimedOut'
            puts "!! Failing workflow execution! (timed out activity)"
            task.fail_workflow_execution
            return false

          when 'ActivityTaskFailed'
            puts "!! Failing workflow execution! (failed activity)"
            task.fail_workflow_execution
            return false

          when 'WorkflowExecutionCompleted'
            puts "## Yesss, workflow execution completed!"
            task.workflow_execution.terminate
            return false
        end
      end
    end
  end
```

ワークフロー実行の開始

ワークフローがポーリングするための決定タスクを生成する前に、ワークフロー実行を開始する必要があります。

ワークフローの実行を開始するには、登録済みのワークフロータイプ (AWS::SimpleWorkflow::WorkflowType) で start_execution を呼び出します。クラスコンストラクタで取得した `workflow_type` インスタンスメンバーを使用するため、これに小さなラッパーを定義します。

```
  def start_execution
    workflow_execution = @workflow_type.start_execution( {
      :workflowId => @workflowId } )
    poll_for_decisions
  end
end
```

ワークフローが実行中になると、決定イベントがワークフローのタスクリスト上に含まれ始め、start_execution のワークフロー実行オプションとして渡されます。

ワークフロータイプが登録されるときに提供されるオプションとは異なり、start_execution に渡されるオプションは、ワークフロータイプの一部とは見なされません。これらは、ワークフローのバージョンを変更せずに、ワークフロー実行ごとに自由に変更できます。

ファイルの実行時にワークフローの実行を開始するため、 クラスをインスタンス化し、先ほど定義した start_executionメソッドを呼び出すコードを追加します。

```
if __FILE__ == $0
  require 'securerandom'

  # Use a different task list name every time we start a new workflow execution.
  #
  # This avoids issues if our pollers re-start before SWF considers them closed,
  # causing the pollers to get events from previously-run executions.
  workflowId = SecureRandom.uuid

  # Let the user start the activity worker first...

  puts ""
  puts "Amazon SWF Example"
  puts "------------------"
  puts ""
  puts "Start the activity worker, preferably in a separate command-line window, with"
  puts "the following command:"
  puts ""
  puts "> ruby swf_sns_activities.rb #{workflowId}-activities"
  puts ""
  puts "You can copy & paste it if you like, just don't copy the '>' character."
  puts ""
  puts "Press return when you're ready..."

  i = gets

  # Now, start the workflow.

  puts "Starting workflow execution."
  sample_workflow = SampleWorkflow.new(workflowId)
  sample_workflow.start_execution
end
```

タスクリストの名前の競合を避けるため、SecureRandom.uuid を使用してタスクリスト名として使用できるランダムな UUID を生成し、各ワークフロー実行に異なるタスクリスト名が使用されることを保証します。

> **Note**
>
> タスクリストはワークフロー実行に関するイベントを記録するために使用されるので、同じワークフロータイプの複数の実行に同じタスクリストを使用すると、前の実行中に生成されたイベントを取得する可能性があります。新しいコードの試行やテストの実行中によくあるケースですが、特に互いにすぐ連続して実行される場合に、その可能性があります。

前の実行からのアーティファクトに対処しなければならないという問題を回避するには、各実行に新しいタスクリストを使用して、ワークフロー実行の開始時にそれを指定できます。

ここでもまた、それを実行している人 (おそらくお客様) に手順を示し、タスクリストの「アクティビティ」バージョンを提供するコードが少しあります。ディサイダーはこのタスクリスト名を使用してワークフローのアクティビティをスケジュールし、アクティビティ実装は、このタスクリスト名のアクティビティイベントをリッスンして、そのスケジュールされたアクティビティをいつ開始するかを知り、アクティビティ実行に関する更新を提供します。

また、コードはワークフロー実行を開始する前にユーザーがアクティビティスターターの実行を開始するのを待機します。そのため、アクティビティタスクが提供されたタスクリストに表示され始めるときには、アクティビティスターターは応答する準備が整っています。

次のステップ

ワークフローは実装しました。次に、「サブスクリプションワークフローのチュートリアルのパート 3: アクティビティの実装」では、アクティビティとアクティビティスターターを定義します。

サブスクリプションワークフローのチュートリアルのパート 3: アクティビティの実装

アクティビティをワークフローに実装します。まず、アクティビティコードの一般的な機能を提供する基本クラスから開始します。

トピック

基本アクティビティタイプの定義

ワークフローの設計時に確認したアクティビティは以下のとおりです。

- `get_contact_activity`
- `subscribe_topic_activity`
- `wait_for_confirmation_activity`
- `send_result_activity`

ここで、以上の各アクティビティを実装します。アクティビティではいくつかの機能を共有するため、少し基礎を整え、共有できる一般的なコードを作成しましょう。これを BasicActivity と呼び、`basic_activity.rb` という新しいファイルに定義します。

他のソースファイルのように、`utils.rb` を含めて `init_domain` 関数にアクセスし、サンプルドメインをセットアップします。

```
require_relative 'utils.rb'
```

次に、基本的なアクティビティクラスと、各アクティビティで関心があるいくつかの一般的なデータを宣言します。クラスの属性に、アクティビティの AWS::SimpleWorkflow::ActivityType インスタンス、名前、および結果を保存します。

```
class BasicActivity

  attr_accessor :activity_type
  attr_accessor :name
  attr_accessor :results
```

これらの属性は、アクティビティ名を取得するクラス initialize メソッドで定義されたインスタンスデータ、およびアクティビティを Amazon SWF に登録するときに使用されるオプションのバージョンとマップにアクセスします。

```
    def initialize(name, version = 'v1', options = nil)

      @activity_type = nil
      @name = name
      @results = nil

      # get the domain to use for activity tasks.
      @domain = init_domain

      # Check to see if this activity type already exists.
      @domain.activity_types.each do | a |
        if (a.name == @name) && (a.version == version)
          @activity_type = a
        end
      end

      if @activity_type.nil?
        # If no options were specified, use some reasonable defaults.
        if options.nil?
          options = {
            # All timeouts are in seconds.
            :default_task_heartbeat_timeout => 900,
            :default_task_schedule_to_start_timeout => 120,
            :default_task_schedule_to_close_timeout => 3800,
            :default_task_start_to_close_timeout => 3600 }
        end
        @activity_type = @domain.activity_types.register(@name, version, options)
      end
    end
```

ワークフロータイプの登録と同様に、アクティビティタイプが既に登録されている場合、ドメインの activity_types コレクションを表示してそれを取得できます。アクティビティが見つからない場合は、登録されます。

また、ワークフロータイプと同様に、登録時にアクティビティタイプとともに保存されるデフォルトオプションを設定できます。

基本アクティビティで最後に取得するのは、それを実行するための一貫した方法です。アクティビティタスクを受け取る do_activity メソッドを定義します。示されているように、渡されたアクティビティタスクを使用して、input インスタンス属性経由でデータを受け取ることができます。

```
  def do_activity(task)
    @results = task.input # may be nil
    return true
  end
end
```

これで BasicActivity クラスの説明を終わります。これを使用して、アクティビティの定義を簡単で一貫性のあるものにできます。

GetContactActivity の定義

ワークフロー実行で実行される最初のアクティビティは get_contact_activity です。このアクティビティでは、ユーザーの Amazon SNS トピックのサブスクリプション情報を取得します。

get_contact_activity.rb という名前の新しいファイルを作成し、Amazon SWF に渡す文字列を準備するために使用する yaml と、この GetContactActivity クラスの基礎として使用する basic_activity.rb の両方を必要とします。

```
require 'yaml'
require_relative 'basic_activity.rb'

# **GetContactActivity** provides a prompt for the user to enter contact
# information. When the user successfully enters contact information, the
# activity is complete.
class GetContactActivity < BasicActivity
```

BasicActivity にアクティビティ登録コードを配置するため、GetContactActivity のinitializeメソッドは非常に簡単です。基本クラスのコンストラクタを単純にアクティビティ名 get_contact_activity で呼びます。これで、アクティビティを登録するために必要なすべての操作を実行しました。

```
  # initialize the activity
  def initialize
    super('get_contact_activity')
  end
```

次に、do_activity メソッドを定義します。このメソッドはユーザーの E メール、電話番号、またはその両方の入力を求めます。

```
    def do_activity(task)
      puts ""
      puts "Please enter either an email address or SMS message (mobile phone) number
to"
      puts "receive SNS notifications. You can also enter both to use both address
types."
      puts ""
      puts "If you enter a phone number, it must be able to receive SMS messages, and
must"
      puts "be 11 digits (such as 12065550101 to represent the number
1-206-555-0101)."

      input_confirmed = false
      while !input_confirmed
        puts ""
        print "Email: "
        email = $stdin.gets.strip

        print "Phone: "
        phone = $stdin.gets.strip

        puts ""
        if (email == '') && (phone == '')
          print "You provided no subscription information. Quit? (y/n)"
           confirmation = $stdin.gets.strip.downcase
           if confirmation == 'y'
             return false
           end
        else
           puts "You entered:"
           puts "  email: #{email}"
           puts "  phone: #{phone}"
           print "\nIs this correct? (y/n): "
           confirmation = $stdin.gets.strip.downcase
           if confirmation == 'y'
             input_confirmed = true
           end
        end
      end
```

```
      # make sure that @results is a single string. YAML makes this easy.
      @results = { :email => email, :sms => phone }.to_yaml
      return true
    end
  end
```

do_activity の最後に、ユーザーから取得した E メールと電話番号を使用し、それをマップに置き、to_yaml を使用してマップ全体を YAML 文字列に変換します。これには重要な理由があります。アクティビティの完了時に Amazon SWF に渡す結果は、文字列データのみ である必要があります。オブジェクトを簡単に YAML 文字列に変換し、再びオブジェクトに戻す Ruby の機能は、この目的に最適です。

これで get_contact_activity の実装が終わりました。このデータは、次に subscribe_topic_activity の実装で使用します。

SubscribeTopicActivity の定義

Amazon SNS の詳細を説明し、ユーザーを Amazon SNS トピックにサブスクライブするために get_contact_activity によって生成された情報を使用するアクティビティを作成します。

subscribe_topic_activity.rb という名前の新しいファイルを作成します。get_contact_activity に使用したのと同じ要件を追加し、クラスを宣言して、その initialize メソッドを提供します。

```
require 'yaml'
require_relative 'basic_activity.rb'

# **SubscribeTopicActivity** sends an SMS / email message to the user, asking for
# confirmation.  When this action has been taken, the activity is complete.
class SubscribeTopicActivity < BasicActivity

  def initialize
    super('subscribe_topic_activity')
  end
```

これでアクティビティをセットアップおよび登録するためにコードを配置したので、コードを追加して Amazon SNS トピックを作成します。そのためには、AWS::SNS::Client オブジェクトの create_topic メソッドを使用します。

create_topic メソッドをクラスに追加します。このメソッドは渡された Amazon SNS クライアントオブジェクトを受け取ります。

```
def create_topic(sns_client)
  topic_arn = sns_client.create_topic(:name => 'SWF_Sample_Topic')[:topic_arn]

  if topic_arn != nil
    # For an SMS notification, setting `DisplayName` is *required*. Note that
    # only the *first 10 characters* of the DisplayName will be shown on the
    # SMS message sent to the user, so choose your DisplayName wisely!
    sns_client.set_topic_attributes( {
      :topic_arn => topic_arn,
      :attribute_name => 'DisplayName',
      :attribute_value => 'SWFSample' } )
  else
    @results = {
      :reason => "Couldn't create SNS topic", :detail => "" }.to_yaml
    return nil
  end

  return topic_arn
end
```

トピックの Amazon リソースネーム (ARN) を設定したら、これを Amazon SNS クライアントの set_topic_attributes メソッドで使用して、トピックの DisplayName を設定します。これは、Amazon SNS で SMS メッセージを送信するために必要です。

最後に、do_activity メソッドを定義します。アクティビティがスケジュールされたときに input オプション経由で渡されたデータの収集から開始します。前に説明したように、これは文字列として渡す必要があります。この文字列は「to_yaml」を使用して作成しました。文字列を取得するときは、YAML.load を使用してデータを Ruby オブジェクトに変換します。

これは do_activity の開始であり、ここでは入力データを取得します。

```
def do_activity(task)
  activity_data = {
    :topic_arn => nil,
    :email => { :endpoint => nil, :subscription_arn => nil },
    :sms => { :endpoint => nil, :subscription_arn => nil },
  }

  if task.input != nil
    input = YAML.load(task.input)
    activity_data[:email][:endpoint] = input[:email]
    activity_data[:sms][:endpoint] = input[:sms]
```

```
    else
      @results = { :reason => "Didn't receive any input!", :detail => "" }.to_yaml
      puts("  #{@results.inspect}")
      return false
    end

    # Create an SNS client. This is used to interact with the service. Set the
    # region to $SMS_REGION, which is a region that supports SMS notifications
    # (defined in the file `utils.rb`).
    sns_client = AWS::SNS::Client.new(
      :config => AWS.config.with(:region => $SMS_REGION))
```

入力がなかった場合、多くの操作はないため、アクティビティが失敗します。

ただし、すべて問題ないと仮定すると、引き続き do_activityメソッドを入力し、 で Amazon SNS クライアントを取得し AWS SDK for Ruby、それを create_topicメソッドに渡して Amazon SNS トピックを作成します。

```
    # Create the topic and get the ARN
    activity_data[:topic_arn] = create_topic(sns_client)

    if activity_data[:topic_arn].nil?
      return false
    end
```

ここでいくつか注意が必要なことがあります。

- AWS.config.with を使用して、Amazon SNS クライアントのリージョンを設定します。SMS メッセージを送信するため、utils.rb で宣言した SMS 対応リージョンを使用します。
- トピックの ARN を activity_data マップに保存します。これはワークフローの次のアクティビティに渡されるデータの部分です。

最後に、このアクティビティは、渡されたエンドポイント (E メールと SMS) を使用してユーザーを Amazon SNS トピックにサブスクライブします。ユーザーが両方のエンドポイントを入力する必要はありませんが、少なくとも 1 つが必要です。

```
    # Subscribe the user to the topic, using either or both endpoints.
    [:email, :sms].each do | x |
      ep = activity_data[x][:endpoint]
      # don't try to subscribe an empty endpoint
```

```
          if (ep != nil && ep != "")
            response = sns_client.subscribe( {
              :topic_arn => activity_data[:topic_arn],
              :protocol => x.to_s, :endpoint => ep } )
            activity_data[x][:subscription_arn] = response[:subscription_arn]
          end
        end
```

AWS::SNS::Client.subscribe はトピック ARN、プロトコル (これは明らかに対応エンドポイントの activity_data マップキーとして見せかけたものです)。

最後に、YAML 形式で次のアクティビティの情報を再パッケージし、Amazon SWF に送信できるようにします。

```
        # if at least one subscription arn is set, consider this a success.
        if (activity_data[:email][:subscription_arn] != nil) or (activity_data[:sms]
[:subscription_arn] != nil)
          @results = activity_data.to_yaml
        else
          @results = { :reason => "Couldn't subscribe to SNS topic", :detail =>
 "" }.to_yaml
          puts("  #{@results.inspect}")
          return false
        end
        return true
      end
    end
```

これで subscribe_topic_activity の実装が完了します。次に、wait_for_confirmation_activity を定義します。

WaitForConfirmationActivity の定義

ユーザーは、Amazon SNS トピックに登録したら、サブスクリプションリクエストを確認する必要があります。この場合は、ユーザーが E メールまたは SMS メッセージによってサブスクリプションを確認するのを待ちます。

ユーザーによるサブスクリプションの確認を待機するアクティビティは wait_for_confirmation_activity と呼ばれ、ここで定義します。開始するには、wait_for_confirmation_activity.rb と呼ばれる新しいファイルを作成し、前のアクティビティをセットアップしたようにセットアップします。

```
require 'yaml'
require_relative 'basic_activity.rb'

# **WaitForConfirmationActivity** waits for the user to confirm the SNS
# subscription.  When this action has been taken, the activity is complete. It
# might also time out...
class WaitForConfirmationActivity < BasicActivity

  # Initialize the class
  def initialize
    super('wait_for_confirmation_activity')
  end
```

次に、do_activity メソッドの定義を開始し、subscription_data というローカル変数に入力データを取得します。

```
  def do_activity(task)
    if task.input.nil?
      @results = { :reason => "Didn't receive any input!", :detail => "" }.to_yaml
      return false
    end

    subscription_data = YAML.load(task.input)
```

これでトピック ARN を用意できたので、AWS::SNS::Topic の新しいインスタンスを作成し、これを ARN に渡します。

```
    topic = AWS::SNS::Topic.new(subscription_data[:topic_arn])

    if topic.nil?
      @results = {
        :reason => "Couldn't get SWF topic ARN",
        :detail => "Topic ARN: #{topic.arn}" }.to_yaml
      return false
    end
```

ここで、トピックをチェックし、ユーザーがいずれかのエンドポイントを使用して、サブスクリプションを確認したかどうか調べます。アクティビティを成功と見なすためには、1 つのエンドポイントの確認のみが必要です。

Amazon SNS トピックはそのトピックのサブスクリプションのリストを保持します。サブスクリプションの ARN が PendingConfirmation 以外に設定されているかどうか確認して、ユーザーが特定のサブスクリプションを確認したかどうかをチェックできます。

```
      # loop until we get some indication that a subscription was confirmed.
      subscription_confirmed = false
      while(!subscription_confirmed)
        topic.subscriptions.each do | sub |
          if subscription_data[sub.protocol.to_sym][:endpoint] == sub.endpoint
            # this is one of the endpoints we're interested in. Is it subscribed?
            if sub.arn != 'PendingConfirmation'
              subscription_data[sub.protocol.to_sym][:subscription_arn] = sub.arn
              puts "Topic subscription confirmed for (#{sub.protocol}:
#{sub.endpoint})"
              @results = subscription_data.to_yaml
              return true
            else
              puts "Topic subscription still pending for (#{sub.protocol}:
#{sub.endpoint})"
            end
          end
        end
```

サブスクリプションの ARN を取得する場合、これをアクティビティの結果データに保存し、YAML に変換して、do_activity から true を返します。これにより、アクティビティが正常に完了したことを伝えます。

サブスクリプションが確認されるまでに時間がかかる場合があるため、アクティビティタスクrecord_heartbeatで を呼び出すことがあります。これにより、アクティビティがまだ処理中であることが Amazon SWF に伝えられ、これを使用してアクティビティの進行状況の更新を提供できます (ファイルの処理など何かを実行中は、その進捗状況を報告できます)。

```
        task.record_heartbeat!(
          { :details => "#{topic.num_subscriptions_confirmed} confirmed,
#{topic.num_subscriptions_pending} pending" })
        # sleep a bit.
        sleep(4.0)
      end
```

これで while ループを終了します。成功せずにループを終了した場合、失敗を報告して do_activity メソッドを終了します。

```
      if (subscription_confirmed == false)
        @results = {
          :reason => "No subscriptions could be confirmed",
          :detail => "#{topic.num_subscriptions_confirmed} confirmed,
#{topic.num_subscriptions_pending} pending" }.to_yaml
        return false
      end
    end
  end
```

これで wait_for_confirmation_activity の実装が終わりました。定義するアクティビティは後 1 つのみ (send_result_activity) です。

SendResultActivity の定義

ワークフローがここまで進んだ場合、ユーザーを Amazon SNS トピックに正しくサブスクライブし、ユーザーはサブスクリプションを確認しました。

最後のアクティビティ send_result_activity では、ユーザーがサブスクライブしたトピックと、ユーザーがサブスクリプションを確認したエンドポイントを使用して、ユーザーに成功したトピックサブスクリプションの確認を送信します。

send_result_activity.rb と呼ばれる新しいファイルを作成し、これまでのすべてのアクティビティをセットアップしたようにセットアップします。

```
require 'yaml'
require_relative 'basic_activity.rb'

# **SendResultActivity** sends the result of the activity to the screen, and, if
# the user successfully registered using SNS, to the user using the SNS contact
# information collected.
class SendResultActivity < BasicActivity

  def initialize
    super('send_result_activity')
  end
```

do_activity メソッドは同様に開始し、ワークフローから入力データを取得し、YAML から変換して、トピック ARN を使用して AWS::SNS::Topic インスタンスを作成します。

```
  def do_activity(task)
```

```
      if task.input.nil?
        @results = { :reason => "Didn't receive any input!", :detail => "" }
        return false
      end

      input = YAML.load(task.input)

      # get the topic, so we publish a message to it.
      topic = AWS::SNS::Topic.new(input[:topic_arn])

      if topic.nil?
        @results = {
          :reason => "Couldn't get SWF topic",
          :detail => "Topic ARN: #{topic.arn}" }
        return false
      end
```

トピックの準備ができたら、そのトピックにメッセージを発行します (また、画面にエコーします)。

```
      @results = "Thanks, you've successfully confirmed registration, and your
workflow is complete!"

      # send the message via SNS, and also print it on the screen.
      topic.publish(@results)
      puts(@results)

      return true
    end
  end
```

Amazon SNS トピックへの発行では、そのトピックに対して存在するサブスクライブおよび確認済みのすべてのエンドポイントに提供するメッセージを送信します。したがって、ユーザーが E メールと SMS 番号の両方で確認された場合、ユーザーはエンドポイントごとに 2 つの確認メッセージを受信します。

次のステップ

これで send_result_activity の実装が完了します。ここでアクティビティタスクを処理し、サブスクリプションワークフローのチュートリアルのパート 4: アクティビティタスクポーラーの実装レスポンスでアクティビティタスクを起動できるアクティビティアプリケーションで、これらのアクティビティをまとめます。

サブスクリプションワークフローのチュートリアルのパート 4: アクティビティタスクポーラーの実装

Amazon SWF で、ワークフロー実行を行うためのアクティビティタスクは、アクティビティタスクリストに表示されます。このリストは、ワークフローでアクティビティをスケジュールするときに表示されます。

ここでは、ワークフローでこれらのタスクを処理するための基本的なアクティビティポーラーを実装し、それを使用して、アクティビティを開始するために Amazon SWF がアクティビティタスクリストにタスクを配置したときにアクティビティを起動します。

開始するには、swf_sns_activities.rb という新しいファイルを作成します。これを使用して、以下のことを行います。

- 作成したアクティビティクラスをインスタンス化する。
- 各アクティビティを Amazon SWF に登録する。
- アクティビティをポーリングし、アクティビティタスクリストに名前が表示されたら、各アクティビティに対して do_activity を呼び出す。

swf_sns_activities.rb で、次のステートメントを追加して、定義した各アクティビティクラスを要求する。

```
require_relative 'get_contact_activity.rb'
require_relative 'subscribe_topic_activity.rb'
require_relative 'wait_for_confirmation_activity.rb'
require_relative 'send_result_activity.rb'
```

ここでクラスを作成し、初期化コードを指定します。

```
class ActivitiesPoller

  def initialize(domain, workflowId)
    @domain = domain
    @workflowId = workflowId
    @activities = {}

    # These are the activities we'll run
    activity_list = [
      GetContactActivity,
```

```
      SubscribeTopicActivity,
      WaitForConfirmationActivity,
      SendResultActivity ]

    activity_list.each do | activity_class |
      activity_obj = activity_class.new
      puts "** initialized and registered activity: #{activity_obj.name}"
      # add it to the hash
      @activities[activity_obj.name.to_sym] = activity_obj
    end
  end
```

このコードは、ドメインおよびタスクリストで渡された情報を保存することに加えて、作成した各アクティビティクラスをインスタンス化します。各クラスは関連するアクティビティを登録するため (このコードを確認する必要がある場合は「basic_activity.rb」を参照してください)、これは実行するすべてのアクティビティについて Amazon SWF に知らせるために十分です。

インスタンス化した各アクティビティは、アクティビティ名 (get_contact_activity など) をキーとして使用してマップに保存し、次に定義するアクティビティポーラーコードで簡単に検索できるようにします。

poll_for_activities という新しいメソッドを作成し、ドメインが保持する activity_tasks で poll を呼び出して、アクティビティタスクを取得します。

```
  def poll_for_activities
    @domain.activity_tasks.poll(@workflowId) do | task |
      activity_name = task.activity_type.name
```

タスクの activity_type メンバーからアクティビティ名を取得できます。次に、このタスクと関連付けられたアクティビティ名を使用して、do_activity を実行するクラスを検索し、そのクラスにタスクを渡します (これには、アクティビティに転送される入力タスクが含まれます)。

```
      # find the task on the activities list, and run it.
      if @activities.key?(activity_name.to_sym)
        activity = @activities[activity_name.to_sym]
        puts "** Starting activity task: #{activity_name}"
        if activity.do_activity(task)
          puts "++ Activity task completed: #{activity_name}"
          task.complete!({ :result => activity.results })
          # if this is the final activity, stop polling.
          if activity_name == 'send_result_activity'
```

```
            return true
          end
        else
          puts "-- Activity task failed: #{activity_name}"
          task.fail!(
            { :reason => activity.results[:reason],
              :details => activity.results[:detail] } )
        end
      else
        puts "couldn't find key in @activities list: #{activity_name}"
        puts "contents: #{@activities.keys}"
      end
    end
  end
end
```

コードは do_activity が完了するのを待機したのち、リターンコードに基づいてタスクで complete! または fail! を呼び出します。

> **Note**
>
> このコードは、ミッションを完了し、すべてのアクティビティを起動したため、最終アクティビティが開始されるとポーラーから終了します。独自の Amazon SWF コードでアクティビティを再度実行する場合、アクティビティポーラーの実行を無限に維持できます。

以上で ActivitiesPoller クラスのコードは終わりですが、ユーザーがコマンドラインから実行できるように、ファイルの末尾に少しコードを追加します。

```
if __FILE__ == $0
  if ARGV.count < 1
    puts "You must supply a task-list name to use!"
    exit
  end
  poller = ActivitiesPoller.new(init_domain, ARGV[0])
  poller.poll_for_activities
  puts "All done!"
end
```

ユーザーがコマンドラインからファイルを実行すると (最初の引数としてアクティビティタスクリストを渡す)、このコードはポーラークラスをインスタンス化し、アクティビティのポーリングを開始

します。ポーラーが完了したら (最終のアクティビティの起動後)、メッセージを出力して終了します。

以上がアクティビティポーラーに関する説明です。後はコードを実行し、「サブスクリプションワークフローのチュートリアル: ワークフローの実行」でその動作を確認するだけです。

サブスクリプションワークフローのチュートリアル: ワークフローの実行

ワークフロー、アクティビティ、ワークフローとアクティビティのポーラーの実装を完了したので、ワークフローを実行する準備ができました。

まだ行っていない場合は、チュートリアルのパート 1 AWS セッションの設定のように、 `aws-config.txt` ファイルで AWS アクセスキーを指定する必要があります。

では、コマンドラインに進み、チュートリアルのソースファイルがあるディレクトリに移動します。以下のファイルが必要です。

```
.
|-- aws-config.txt
|-- basic_activity.rb
|-- get_contact_activity.rb
|-- send_result_activity.rb
|-- subscribe_topic_activity.rb
|-- swf_sns_activities.rb
|-- swf_sns_workflow.rb
|-- utils.rb
`-- wait_for_confirmation_activity.rb
```

次のコマンドを使ってワークフローを開始します。

```
ruby swf_sns_workflow.rb
```

これにより、ワークフローが開始され、コピーして別のコマンドラインウィンドウに (または、チュートリアルのソースファイルをコピーしている別のコンピュータがあればそこに) 貼り付けできる行のあるメッセージが出力されます。

```
Amazon SWF Example
------------------
```

```
Start the activity worker, preferably in a separate command-line window, with
the following command:

> ruby swf_sns_activities.rb 87097e76-7c0c-41c7-817b-92527bb0ea85-activities

You can copy & paste it if you like, just don't copy the '>' character.

Press return when you're ready...
```

ワークフローコードは、別のウィンドウでアクティビティポーラーが開始するのをじっと待機します。

新しいコマンドラインウィンドウを開いて、ソースファイルがあるディレクトリに再度移動した後、swf_sns_workflow.rb によりされるコマンドを使用してアクティビティポーラーを開始します。たとえば、上記の出力を受け取った場合、以下を入力 (または貼り付け) します。

```
ruby swf_sns_activities.rb 87097e76-7c0c-41c7-817b-92527bb0ea85-activities
```

アクティビティポーラーを開始したら、アクティビティ登録についての情報の出力が開始されます。

```
** initialized and registered activity: get_contact_activity
** initialized and registered activity: subscribe_topic_activity
** initialized and registered activity: wait_for_confirmation_activity
** initialized and registered activity: send_result_activity
```

これで、元のコマンドラインのウィンドウに戻り、リターンを押してワークフロー実行を開始できます。これにより、ワークフローは登録され、最初のアクティビティがスケジュールされます。

```
Starting workflow execution.
** registered workflow: swf-sns-workflow
** scheduling activity task: get_contact_activity
```

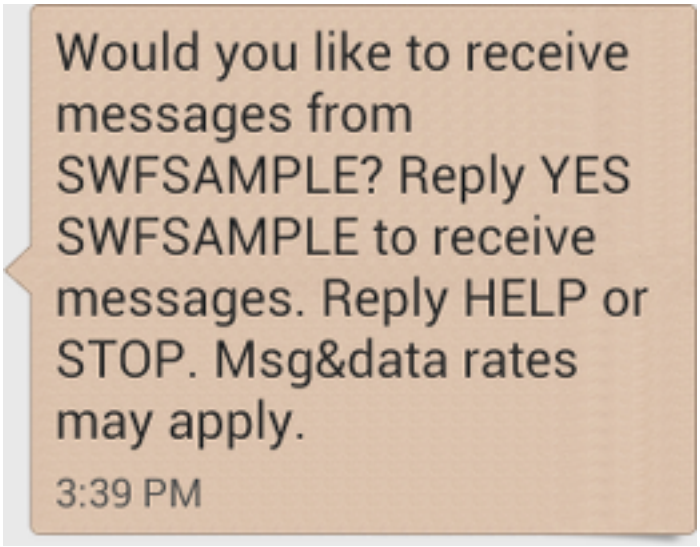
アクティビティポーラーを実行しているもう 1 つのウィンドウに戻ります。最初に実行しているアクティビティの結果が表示され、メールまたは SMS 電話番号を入力するよう求められます。これらのデータのいずれか、または両方を入力し、入力したテキストを確認します。

```
activity task received: <AWS::SimpleWorkflow::ActivityTask>
** Starting activity task: get_contact_activity
```

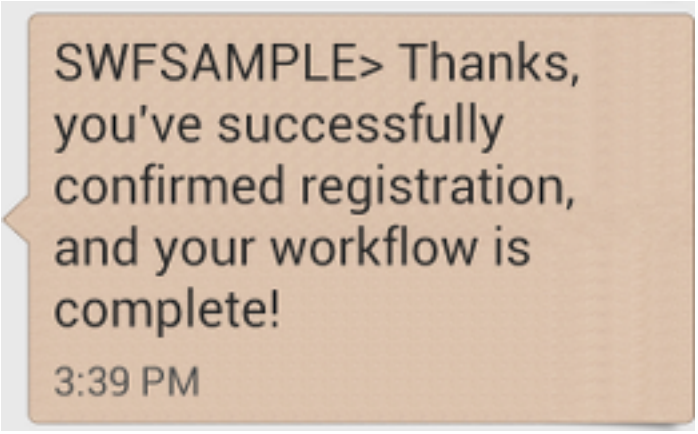
```
Please enter either an email address or SMS message (mobile phone) number to
receive Amazon SNS notifications. You can also enter both to use both address types.

If you enter a phone number, it must be able to receive SMS messages, and must
be 11 digits (such as 12065550101 to represent the number 1-206-555-0101).

Email: me@example.com
Phone: 12065550101

You entered:
  email: me@example.com
  phone: 12065550101

Is this correct? (y/n): y
```

> **Note**
>
> ここにある電話番号は架空のもので、例を示すことのみを目的として使用しています。実際には自分の電話番号とメールアドレスを使用します。

この情報を入力するとすぐに、Amazon SNS からメールまたはからテキストメッセージが届き、トピックのサブスクリプションを確認するよう求められます。SMS 番号を入力した場合は、電話に次のような画面が表示されます。



このメッセージへ YES と返信すると、`send_result_activity` で提供したレスポンスが返されます。



このすべてが起こったときに、コマンドラインのウィンドウで何が起きているか確認しましたか? ワークフローとアクティビティポーラーは、ずっと作業し続けています。

ワークフローポーラーの出力を以下に示します。

```
** scheduling activity task: subscribe_topic_activity
** scheduling activity task: wait_for_confirmation_activity
** scheduling activity task: send_result_activity
!! All activities complete! Sending complete_workflow_execution...
```

アクティビティポーラーの出力を以下に示します。これは、もう 1 つのコマンドラインウィンドウで起きていたことです。

```
++ Activity task completed: get_contact_activity
** Starting activity task: subscribe_topic_activity
++ Activity task completed: subscribe_topic_activity
** Starting activity task: wait_for_confirmation_activity
Topic subscription still pending for (email: me@example.com)
Topic subscription confirmed for (sms: 12065550101)
++ Activity task completed: wait_for_confirmation_activity
** Starting activity task: send_result_activity
Thanks, you've successfully confirmed registration, and your workflow is complete!
++ Activity task completed: send_result_activity
All done!
```

これで、ワークフロー、そしてこのチュートリアルはすべて完了しました。

タイムアウトの動作を確認するため、または、異なるデータを入力するために、ワークフローを再度実行する場合があるかもしれません。一度トピックへのサブスクライブをすると、サブスクリプションを解除するまではサブスクリプションが継続していることを忘れないで下さい。トピックのサブス

クライブを解除する前にワークフローを再実行すると、サブスクリプションがすでに確認されていることが `wait_for_confirmation_activity`に表示されるため、おそらく自動的な成功になります。

Amazon SNS トピックのサブスクライブを解除する方法

- テキストメッセージに否定の返信をします (`STOP` を送信)。
- E メールで受信したサブスクリプション解除のリンクを選択します。

トピックを再度サブスクライブする準備ができました。

次のステップ

このチュートリアルでは多くのことを取り上げていますが、、Amazon SWF AWS SDK for Ruby、または Amazon SNS についてさらに学ぶことができます。詳細および、さらに多くの例については、以下の各項目の公式ドキュメントを参照してください。

- AWS SDK for Ruby ドキュメント
- Amazon Simple Notification Service ドキュメント
- Amazon Simple Workflow Service ドキュメント

Amazon SWF コンソールでの作業

Amazon SWF コンソールには、ワークフロー実行を設定、開始、管理するためのオプションが用意されています。

Amazon SWF コンソールを使用すると、次のことができます。

- ワークフロードメインの登録。
- ワークフロータイプとアクティビティタイプを登録します。
- ワークフロー実行を開始、表示、通知、キャンセル、終了、再起動します。

ドメインの登録

ワークフローは、ワークフローの範囲を制御するドメインと呼ばれる AWS リソースで実行されます。 AWS アカウントは複数のドメインを持つことができ、それぞれに複数のワークフローを含めることができますが、異なるドメインのワークフローは連動できません。

ドメイン登録は、 コンソールで最初に利用できる唯一の機能です。少なくとも 1 つのドメインが登録されたら、ドメインに対して次のアクションを実行できます。

- ワークフローとアクティビティタイプを登録します。
- ワークフロー実行の開始。
- 実行中のワークフロー実行のキャンセル、終了、シグナルの送信。
- クローズしたワークフロー実行の再起動。

ドメインの廃止や廃止などのドメイン管理アクションを実行することもできます。

ドメインを非推奨にした後は、それを使用して新しいワークフロー実行の作成や新しいワークフローの登録できなくなります。ドメインを非推奨にすると、そのドメインに登録されているすべてのアクティビティとワークフローも非推奨化されます。ドメインが非推奨化される前に開始された実行は引き続き実行されます。

以前に廃止されたドメインを非推奨にした後、ドメインを使用してワークフロータイプを登録し、新しいワークフロー実行を開始できます。

これらのドメイン管理アクションの詳細については、「DeprecateDomain」と「UndeprecateDomain」を参照してください。

ワークフロータイプの登録

少なくとも 1 つのドメインを登録した後、Amazon SWF コンソールにワークフロータイプを登録できます。

ワークフロータイプは、目的を実行し、アクティビティを調整するロジックを含むアクティビティタイプのセットです。ワークフロータイプは、複数のコンピューティングデバイス間で非同期的に実行できるアクティビティの実行を調整および管理し、シーケンシャル処理方法と並列処理方法の両方を備えています。

コンソールを使用して Amazon SWF ワークフロータイプを登録するには

1. ワークフローを登録するドメインを開きます。
2. [登録] を選択し、[ワークフローの登録] を選択します。
3. [ワークフローの登録] ページで、[ワークフロー名] と [ワークフローバージョン] を入力します。オプションで、このワークフローを実行する決定タスクのスケジュール設定に使用するデフォルトタスクリストを指定することもできます。
4. (オプション) [詳細オプション] を選択して、ワークフローに関する以下の詳細を指定します。
   - デフォルトタスクの優先度 — ワークフローに割り当てるデフォルトタスクの優先度。
   - デフォルトの実行開始から終了までのタイムアウト — このワークフローを実行するためのデフォルトの最大持続時間。
   - デフォルトのタスク開始から終了までのタイムアウト — このワークフローの決定タスクのデフォルトの最大持続時間。
   - デフォルトの子ポリシー — 子ワークフローの実行に使用するデフォルトポリシー。
   - デフォルト Lambda ロール — このワークフローにアタッチされたデフォルトの IAM ロール。
5. [ワークフローの登録] を選択します。

アクティビティタイプの登録

アクティビティは、ワークフロータイプを調整して実行するタスクです (顧客の注文の確認、クレジットカードへの請求など) 。アクティビティが実行される順序は、ワークフロータイプの調整ロジックによって決まります。

少なくとも 1 つのドメインが登録された後、アクティビティタイプを登録できます。

コンソールを使用して Amazon SWF アクティビティタイプを登録するには

1. アクティビティを登録するドメインを開きます。
2. [登録] を選択し、[アクティビティの登録] を選択します。
3. [アクティビティの登録]ページで、[アクティビティ名] と [アクティビティバージョン] を入力します。オプションで、このアクティビティのタスクのスケジュール設定に使用するデフォルトタスクリストを指定することもできます。
4. (オプション) [詳細オプション] を選択して、アクティビティに関する以下の詳細を指定します。
   - デフォルトタスクの優先度 — アクティビティに割り当てるデフォルトタスクの優先度。
   - タイムアウト開始までのデフォルトタスクスケジュール — このアクティビティのタスクがワーカーに割り当てられるまでに待機できるデフォルトの最大持続時間。
   - デフォルトのタスク開始から終了までのタイムアウト — ワーカーがこのアクティビティのタスクを処理するためにかけることができるデフォルトの最大持続時間。
   - タイムアウト終了までのデフォルトタスクスケジュール — このアクティビティのタスクのデフォルトの最大持続時間。
   - デフォルトのタスクハートビートタイムアウト — このタイプのタスクを処理するワーカーが RecordActivityTaskHeartbeat を呼び出して進捗状況を報告しなければならないデフォルトの最大時間。
5. [アクティビティの登録] を選択します。

ワークフローの開始

Amazon SWF コンソールからワークフロー実行を開始できます。少なくとも 1 つのワークフローを登録するまではワークフロー実行を開始できません。

コンソールを使用してワークフロー実行を開始するには

1. Amazon SWF コンソールを開き、左側のナビゲーションペインで [ドメイン] を選択します。
2. ドメイン名の下にある [ワークフロー] を選択します。
3. [ワークフロー] ページで、実行するワークフローを選択します。
4. [実行のスタート] を選択します。
5. [Start execution] (実行のスタート) ページで、実行内容を名前で識別するための [ワークフロー名] と [実行 ID] を入力します。オプションで、このワークフローの実行用に生成される決定タスクに使用されるタスクリストを指定することもできます。

6. (オプション) [詳細オプション] を選択して、ワークフローの実行に関する以下の詳細を指定します。

   - タスク優先度 — このワークフロー実行に使用するタスク優先度。
   - 実行開始から終了までのタイムアウト — このワークフロー実行の合計時間。
   - タスク開始から終了までのタイムアウト — このワークフロー実行の決定タスクの最大持続時間。
   - 子ポリシー — TerminateWorkflowExecution アクションを呼び出すことによって、またはタイムアウトの期限切れによってこのワークフロー実行が終了された場合に、その子ワークフロー実行に使用するポリシー。
   - Lambda ロール – このワークフロー実行にアタッチする IAM ロール。

7. [実行のスタート] を選択します。

ワークフロー実行の管理

ワークフロー実行は、名前、ステータス、ID、タグでフィルタリングできます。アクティブなワークフロー実行に入力を含むシグナルを送信できます。ワークフローをキャンセルまたは終了する必要がある場合は、Try-cancel オプションを使用できます。キャンセルは、ワークフロー実行を終了するよりもキャンセルが推奨されます。キャンセルすると、ワークフローはクリーンアップタスクを実行してから適切に終了できるためです。

コンソールでは、現在実行中または閉じているワークフロー実行を管理できます。

ワークフロー実行を管理する方法

1. ワークフロー実行を管理するドメインを開きます。
2. [実行を検索] を選択します。
3. [ワークフロー実行] ページで [実行をプロパティで絞り込む] を選択し、[プロパティ] で次のフィルターのいずれかを選択します。

選択	このフィルターの適用方法
ワークフロー	このフィルターを選択すると、特定のワークフローの実行が一覧表示されます。例えば、`fiction-books-order-workflow` の実行情報を表示するには、次の操作を行います。

選択	このフィルターの適用方法
	1. [ワークフロー] を選択します。 2. [オペレーター] で、[Equals] を選択します。 3. [ワークフロー] で、fiction-books-order-workflow を選択します。 4. (オプション) フィルターを削除して実行の検索を新たに開始するには、[Clear filters] (フィルターをクリア) を選択します。
ステータス	このフィルターを選択すると、特定のステータスの実行が一覧表示されます。例えば、ステータスが [失敗] の実行情報を表示するには、次の操作を行います。 1. [ステータス] を選択します。 2. [オペレーター] で、[Equals] を選択します。 3. [ステータス] で [失敗] を選択します。 4. (オプション) フィルターを削除して実行の検索を新たに開始するには、[Clear filters] (フィルターをクリア) を選択します。
実行 ID	ID に基づいてワークフロー実行を表示するには、このフィルターを選択します。例えば、ID が `fiction-books-order-category1` の実行情報を表示するには、次の操作を行います。 1. [実行 ID] を選択します。 2. [オペレーター] で、[Equals] を選択します。 3. [実行 ID] で fiction-books-order-category1 を選択します。 4. (オプション) フィルターを削除して実行の検索を新たに開始するには、[Clear filters] (フィルターをクリア) を選択します。

選択	このフィルターの適用方法
タグ	このフィルターを選択すると、特定のタグの実行が一覧表示されます。例えば、ステータスが `purchaseOrder` の実行情報を表示するには、次の操作を行います。 1. [タグ] を選択します。 2. [オペレーター] で、[Equals] を選択します。 3. [タグ] で purchaseOrder を選択します。 4. (オプション) フィルターを削除して実行の検索を新たに開始するには、[Clear filters] (フィルターをクリア) を選択します。

4. (オプション) 必要なフィルターを適用してワークフロー実行を一覧表示すると、アクティブな実行に対して次の操作を実行できます。

   - シグナル — このオプションを使用して、実行中のワークフロー実行に追加データを送信します。これを実行するには:

     1. 追加データを送信する実行を選択します。
     2. [シグナル] を選択し、[シグナル実行] ダイアログボックスでデータを指定します。
     3. [シグナル] を選択します。

   - キャンセルを試行 — このオプションを使用して、ワークフロー実行のキャンセルを試行します。ワークフロー実行を終了するよりも、キャンセルすることをお勧めします。キャンセルにより、クリーンアップタスクを実行してから適切にクローズする機会がワークフロー実行に与えられます。

     1. キャンセルする実行を選択します。
     2. [キャンセルを試行] を選択します。

   - 終了 — このオプションを使用してワークフローの実行を終了します。ワークフロー実行を終了するよりも、キャンセルする方が望ましいことに留意してください。

     1. 終了する実行を選択します。
     2. [子ポリシー] で、[終了] が選択されていることを確認します。
     3. (オプション) 実行を終了する [理由] と [詳細] を指定します。
     4. [Terminate] (終了) を選択します。

5. (オプション) 再実行 — このオプションを使用して、クローズしたワークフロー実行を再実行します。

1. ワークフロー実行のリストで、再実行する、クローズした実行を選択します。クローズした実行を選択すると、[再実行] ボタンが有効になります。[再実行] を選択します。
2. [実行を再実行] ページで、ワークフローの開始 で説明されているようにワークフロー実行の詳細を指定します。

Amazon SWF の基本的なワークフローの概念

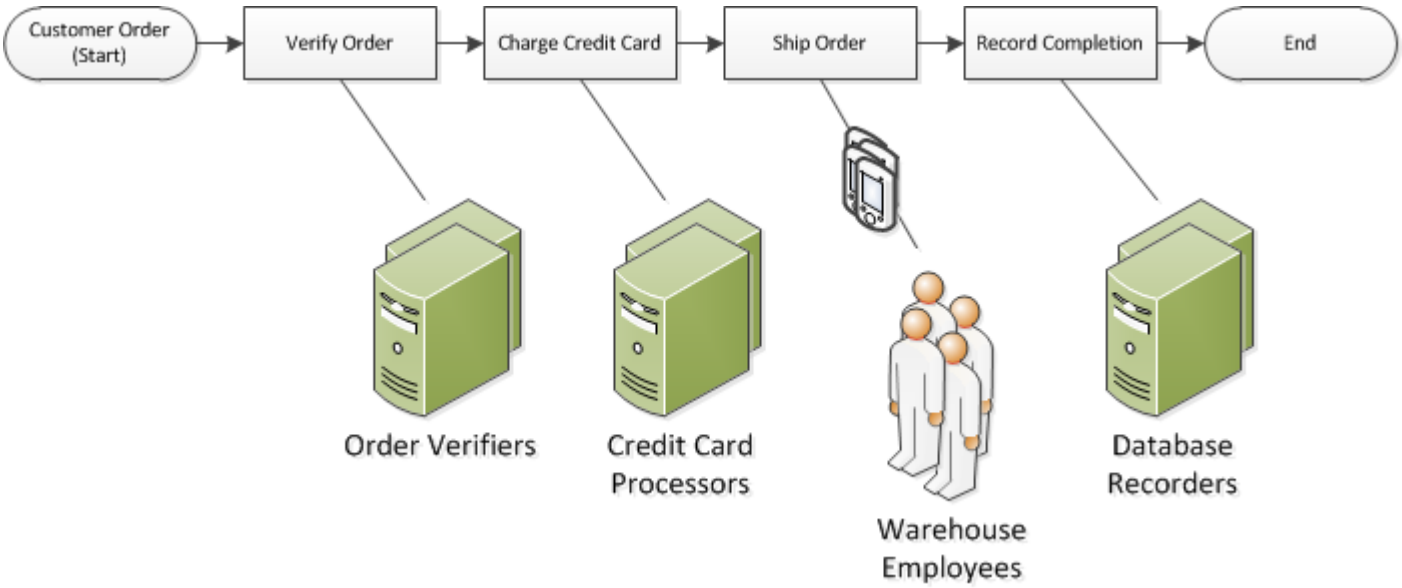
> **Note**
>
> この章の概念では、Amazon Simple Workflow Service の概要を示し、その重要な機能について説明します。例をお探しの場合は、「」を参照してくださいAmazon SWF API を使用する。

Amazon Simple Workflow Service (Amazon SWF) を使用して、分散された非同期アプリケーションをワークフローとして実装できます。ワークフローは複数のコンピューティングデバイス間で非同期的に実行でき、シーケンシャル処理と並列処理の両方を実行できるアクティビティの実行を調整、管理します。

ワークフローを設計するときは、アプリケーションを分析してそのコンポーネントタスクを特定します。Amazon SWF の場合、これらのタスクは、アクティビティを表します。アクティビティが実行される順序は、ワークフローの調整ロジックによって決まります。

e コマースアプリケーションのワークフロー例

次の図は、人と自動プロセスの両方を含む e コマースの注文処理ワークフローを示しています。



e コマースアプリケーションのワークフローは、顧客が注文したときに開始され、次の 4 つのタスクが含まれます。

1. 注文を確認する。
2. 注文が有効である場合、顧客に課金する。
3. 支払いが行われたら、注文品を出荷する。
4. 注文品が出荷されたら、注文の詳細を保存する。

このワークフローのタスクはシーケンシャルです。クレジットカードに課金する前に注文を確認する必要があり、注文品を出荷する前にクレジットカードに正常に課金する必要があります。さらに、注文を記録する前に注文品を出荷する必要があります。それでも、Amazon SWF は分散プロセスをサポートしているため、これらのタスクは別の場所で実行できます。タスクの特性がプログラム的である場合、別のプログラミング言語または別のツールを使用して記述することもできます。

タスクのシーケンシャル処理に加えて、Amazon SWF はタスクの並列処理を使用したワークフローもサポートします。並行タスクは同時に実行され、別のアプリケーションまたは人間のワーカーによって単独に実行することもできます。ワークフローにより、1 つ以上の並行タスクが完了したときの続行方法に関する決定が行われます。

その他の概念

- Amazon SWF でのワークフローの作成
- Amazon SWF でのワークフローの実行
- Amazon SWF のワークフロー履歴
- Amazon SWF のオブジェクト識別子
- Amazon SWF のドメイン
- Amazon SWF のアクター
- Amazon SWF のタスク
- Amazon SWF のタスクリスト
- Amazon SWF でのワークフロー実行の終了
- Amazon SWF ワークフローのライフサイクル
- Amazon SWF でのタスクのポーリング

Amazon SWF でのワークフローの作成

基本的なシーケンシャルワークフローを作成するには、次のステージが必要です。

- ワークフローのモデリング、タイプの登録、アクティビティタイプの登録

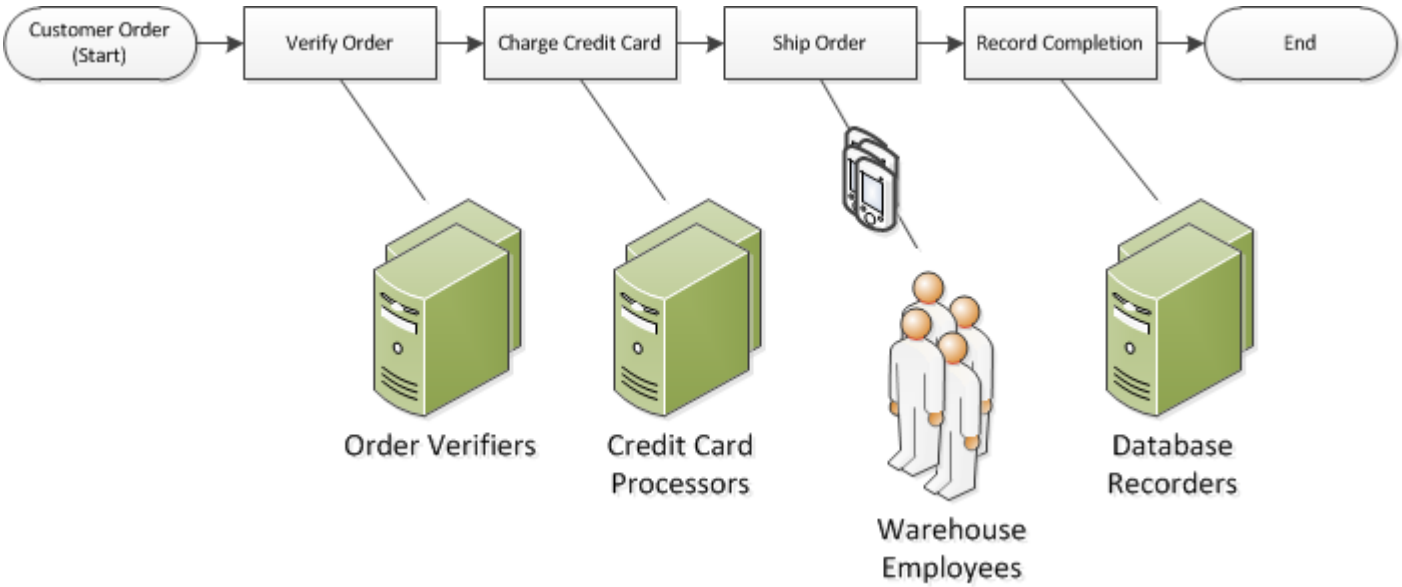
- アクティビティタスクを実行するアクティビティワーカーの開発と起動
- ワークフロー履歴を使用して、次に何をすべきかを判断するディサイダーを開発して起動する
- ワークフロースターター、つまりワークフロー実行を開始するアプリケーションの開発と起動

ワークフローおよびアクティビティのモデル化

Amazon SWF を使用するには、アクティビティとしてアプリケーション内の論理ステップをモデル化します。アクティビティは、ワークフロー内の単一の論理的なステップまたはタスクを表します。たとえば、クレジットカードを認可することは、クレジットカード番号および他の情報を提供すること、および、カードが拒否されたという承認コードやメッセージを受信することを含むアクティビティです。

アクティビティの定義に加えて、判断ポイントを扱う調整ロジックも定義する必要があります。たとえば、調整ロジックは、クレジットカードが承認されたか、または拒否されたかに応じて、異なるフォローアップアクティビティをスケジュールすることがあります。

次の図は、4 つのアクティビティ (注文の確認、クレジットカードに請求、注文を出荷、記録の完了) を含む顧客の注文ワークフローの例を示しています。



Amazon SWF でのワークフローの実行

調整ロジックとアクティビティを設計したら、Amazon SWF を使用してそれらのコンポーネントをワークフローおよびアクティビティタイプとして登録します。登録時に、タイプごとに名前、バージョン、およびデフォルトの設定値を指定します。

Amazon SWF には、登録されたワークフローとアクティビティタイプのみを使用できます。e コマースの例では、CustomerOrder ワークフロータイプ、および VerifyOrder、ChargeCreditCard、ShipOrder、RecordCompletion アクティビティタイプを登録します。

ワークフロータイプの登録後は、何回でも実行できます。ワークフロー実行は、ワークフローの実行中のインスタンスです。

ワークフロー実行は、別のワークフロー実行の後でも、任意のプロセスまたはアプリケーションによって開始できます。e コマースの例では、新しいワークフロー実行が顧客の注文ごとに開始されます。ワークフローを開始するアプリケーションのタイプは、顧客が注文する方法によって異なります。ワークフローは、ウェブサイトやモバイルアプリケーション、または社内アプリケーションを使用してカスタマーサービス担当者が開始できます。

Amazon SWF を使用すると、`workflowId` という識別子をワークフロー実行に関連付けることができるため、既存のビジネス識別子をワークフロー実行に統合することができます。e コマースの例では、各ワークフロー実行を、顧客の請求書番号を使って特定できます。

Amazon SWF は、ユーザーが指定した識別子に加えて、システムが生成する一意の識別子 (`runId`) をワークフローの実行ごとに関連付けます。Amazon SWF では、この識別子を使用して 1 つのワークフロー実行のみを任意の時点で実行できます。同じワークフロータイプの複数のワークフローを実行することはできますが、各ワークフローの実行には別個の `runId` があります。

Amazon SWF のワークフロー履歴

Amazon SWF は、すべてのワークフロー実行の進行状況をワークフロー履歴に記録します。ワークフロー実行の開始以降に発生したすべてのイベントの、詳細で完全、一貫した記録です。

イベントは、スケジュールされている新しいアクティビティや実行中のアクティビティの完了など、ワークフロー実行の状態の個別の変更を表します。ワークフロー履歴には、スケジュールされたアクティビティ、完了したアクティビティ、タスクのタイムアウト、シグナルなど、ワークフロー実行の実行状態を変化させるすべてのイベントが含まれます。

ワークフロー実行の状態を変化させないオペレーションは、通常、ワークフロー履歴には表示されません。たとえば、ワークフロー履歴には、ポーリングの試行や可視性オペレーションの使用は表示されません。

ワークフロー履歴には、以下のような重要な利点があります。

- ワークフロー実行に関するすべての情報はワークフロー履歴に保存されるため、アプリケーションはステートレスである可能性があります。
- ワークフローの実行ごとに、履歴には、スケジュールされたアクティビティ、現在のステータス、およびその結果のレコードが表示されます。ワークフローの実行では、この情報を使用して次のステップを決定します。
- この履歴には、実行中のワークフロー実行をモニタリングし、完了したワークフロー実行を検証するために使用できる詳細な監査証跡が用意されています。

以下は、e コマースのワークフロー履歴の概念図です。

```
Invoice0001

Start Workflow Execution

Schedule Verify Order
Start Verify Order Activity
Complete Verify Order Activity

Schedule Charge Credit Card
Start Charge Credit Card Activity
Complete Charge Credit Card Activity

Schedule Ship Order
Start Ship Order Activity
```

上記の例では、注文が出荷待ちの状態になっています。次の例では、注文が完了しています。ワークフロー履歴は累積的であるため、新しいイベントが追加されます。

```
Invoice0001

Start Workflow Execution

Schedule Verify Order
Start Verify Order Activity
Complete Verify Order Activity

Schedule Charge Credit Card
Start Charge Credit Card Activity
Complete Charge Credit Card Activity
```

```
Schedule Ship Order
Start Ship Order Activity

Complete Ship Order Activity

Schedule Record Order Completion
Start Record Order Completion Activity
Complete Record Order Completion Activity

Close Workflow
```

プログラムでは、ワークフロー実行履歴に関するイベントは、JavaScript Object Notation (JSON) のオブジェクトとして表されます。履歴自体は、これらのオブジェクトの JSON 配列です。各イベントには次のものがあります。

- WorkflowExecutionStarted または ActivityTaskCompleted のような型
- Unix 時間形式のタイムスタンプ
- イベントを一意に識別する ID

さらに、各イベントの型には、その型に適した独自の記述属性セットがあります。たとえば、ActivityTaskCompleted イベントには、アクティビティタスクがスケジュールされた時間およびそのアクティビティタスクが開始された時間に対応するイベントの ID を含む属性と、結果データを保持する属性が含まれています。

GetWorkflowExecutionHistory アクションを使用して、ワークフロー実行履歴の現在の状態のコピーを取得できます。さらに、Amazon SWF とワークフローのディサイダーとのインタラクションの一環として、ディサイダーは履歴のコピーを定期的に受信します。

以下は、JSON 形式でのワークフロー実行履歴の例です。

```
[  {
      "eventId": 11,
      "eventTimestamp": 1326671603.102,
      "eventType": "WorkflowExecutionTimedOut",
      "workflowExecutionTimedOutEventAttributes": {
         "childPolicy": "TERMINATE",
         "timeoutType": "START_TO_CLOSE"
      }
   }, {
      "decisionTaskScheduledEventAttributes": {
         "startToCloseTimeout": "600",
```

```
         "taskList": {
            "name": "specialTaskList"
         }
      },
      "eventId": 10,
      "eventTimestamp": 1326670566.124,
      "eventType": "DecisionTaskScheduled"
   }, {
      "activityTaskTimedOutEventAttributes": {
         "details": "Waiting for confirmation",
         "scheduledEventId": 8,
         "startedEventId": 0,
         "timeoutType": "SCHEDULE_TO_START"
      },
      "eventId": 9,
      "eventTimestamp": 1326670566.124,
      "eventType": "ActivityTaskTimedOut"
   }, {
      "activityTaskScheduledEventAttributes": {
         "activityId": "verification-27",
         "activityType": {
            "name": "activityVerify",
            "version": "1.0"
         },
         "control": "digital music",
         "decisionTaskCompletedEventId": 7,
         "heartbeatTimeout": "120",
         "input": "5634-0056-4367-0923,12/12,437",
         "scheduleToCloseTimeout": "900",
         "scheduleToStartTimeout": "300",
         "startToCloseTimeout": "600",
         "taskList": {
            "name": "specialTaskList"
         }
      },
      "eventId": 8,
      "eventTimestamp": 1326670266.115,
      "eventType": "ActivityTaskScheduled"
   }, {
      "decisionTaskCompletedEventAttributes": {
         "executionContext": "Black Friday",
         "scheduledEventId": 5,
         "startedEventId": 6
      },
```

```
      "eventId": 7,
      "eventTimestamp": 1326670266.103,
      "eventType": "DecisionTaskCompleted"
   }, {
      "decisionTaskStartedEventAttributes": {
         "identity": "Decider01",
         "scheduledEventId": 5
      },
      "eventId": 6,
      "eventTimestamp": 1326670161.497,
      "eventType": "DecisionTaskStarted"
   }, {
      "decisionTaskScheduledEventAttributes": {
         "startToCloseTimeout": "600",
         "taskList": {
            "name": "specialTaskList"
         }
      },
      "eventId": 5,
      "eventTimestamp": 1326668752.66,
      "eventType": "DecisionTaskScheduled"
   }, {
      "decisionTaskTimedOutEventAttributes": {
         "scheduledEventId": 2,
         "startedEventId": 3,
         "timeoutType": "START_TO_CLOSE"
      },
      "eventId": 4,
      "eventTimestamp": 1326668752.66,
      "eventType": "DecisionTaskTimedOut"
   }, {
      "decisionTaskStartedEventAttributes": {
         "identity": "Decider01",
         "scheduledEventId": 2
      },
      "eventId": 3,
      "eventTimestamp": 1326668152.648,
      "eventType": "DecisionTaskStarted"
   }, {
      "decisionTaskScheduledEventAttributes": {
         "startToCloseTimeout": "600",
         "taskList": {
            "name": "specialTaskList"
         }
```

```
      },
      "eventId": 2,
      "eventTimestamp": 1326668003.094,
      "eventType": "DecisionTaskScheduled"
   }
]
```

ワークフローの実行履歴に表示できるさまざまなタイプのイベントの詳細なリストについては、「Amazon Simple Workflow Service API Reference」(Amazon Simple Workflow ServiceAPIリファレンス) の「HistoryEvent data type」(HistoryEvent データ型) を参照してください。

Amazon SWF では、実行が終了してから設定可能な日数の間、すべてのワークフロー実行の完全な履歴が保存されます。ワークフロー履歴の保持期間と呼ばれるこの期間は、ワークフローのドメインを登録するときに指定されます。ドメインについては、このセクションの後半で詳しく説明します。

Amazon SWF のオブジェクト識別子

次のリストに、ワークフロー実行などの Amazon SWF オブジェクト識別子が一意に識別される方法を示します。

- ワークフロータイプ - 登録されたワークフロータイプは、そのドメイン、名前、およびバージョンによって識別されます。ワークフロータイプは、`RegisterWorkflowType` への呼び出しで指定されます。
- アクティビティタイプ - 登録されたアクティビティタイプは、そのドメイン、名前、およびバージョンによって識別されます。アクティビティタイプは、`RegisterActivityType` への呼び出しで指定されます。
- 決定タスクおよびアクティビティタスク - 各決定タスクおよびアクティビティタスクは、一意のタスクトークンによって識別されます。タスクトークンは Amazon SWF によって生成され、`PollForDecisionTask` または `PollForActivityTask` からのレスポンスで、タスクに関する他の情報とともに返されます。トークンはタスクを受信したプロセスによって最も一般的に使用されますが、このプロセスは、別のプロセスにトークンを渡すことができ、それによりタスクの完了または失敗を報告することができます。
- ワークフローの実行 - ワークフローの 1 つの実行は、ドメイン、ワークフロー ID、および実行 ID によって識別されます。最初の 2 つは、StartWorkflowExecution に渡されるパラメータです。`StartWorkflowExecution` によって実行 ID が返されます。

Amazon SWF のドメイン

ワークフローは、 AWS アカウント内の Amazon SWF AWS リソースの範囲を設定する方法を提供するドメインと呼ばれる リソースで実行されます。 Amazon SWF ワークフロータイプやアクティビティタイプなど、ワークフローのすべてのコンポーネントは、ドメインで指定する必要があります。

AWS アカウントは複数のドメインを持つことができ、それぞれに複数のワークフローを含めることができますが、異なるドメインのワークフローは連動できません。

新しいワークフローをセットアップするときは、他のワークフローコンポーネントをセットアップする前に、まだ行っていない場合はドメインを登録する必要があります。

ドメインを登録するときは、ワークフロー履歴の保持期間を指定します。保持期間は、ワークフロー実行が完了した後も Amazon SWF がワークフロー実行に関する情報を引き続き保持する期間です。

ドメイン登録は、 コンソールで最初に利用できる唯一の機能です。少なくとも 1 つのドメインが登録されたら、ドメインに対して次のアクションを実行できます。

- ワークフローとアクティビティタイプを登録します。
- ワークフロー実行の開始。
- 実行中のワークフロー実行のキャンセル、終了、シグナルの送信。
- クローズしたワークフロー実行の再起動。

ドメインの廃止や廃止などのドメイン管理アクションを実行することもできます。

ドメインを非推奨にした後は、それを使用して新しいワークフロー実行の作成や新しいワークフローの登録できなくなります。ドメインを非推奨にすると、そのドメインに登録されているすべてのアクティビティとワークフローも非推奨化されます。ドメインが非推奨化される前に開始された実行は引き続き実行されます。

以前に廃止されたドメインを非推奨にした後、ドメインを使用してワークフロータイプを登録し、新しいワークフロー実行を開始できます。

これらのドメイン管理アクションの詳細については、「DeprecateDomain」と「UndeprecateDomain」を参照してください。

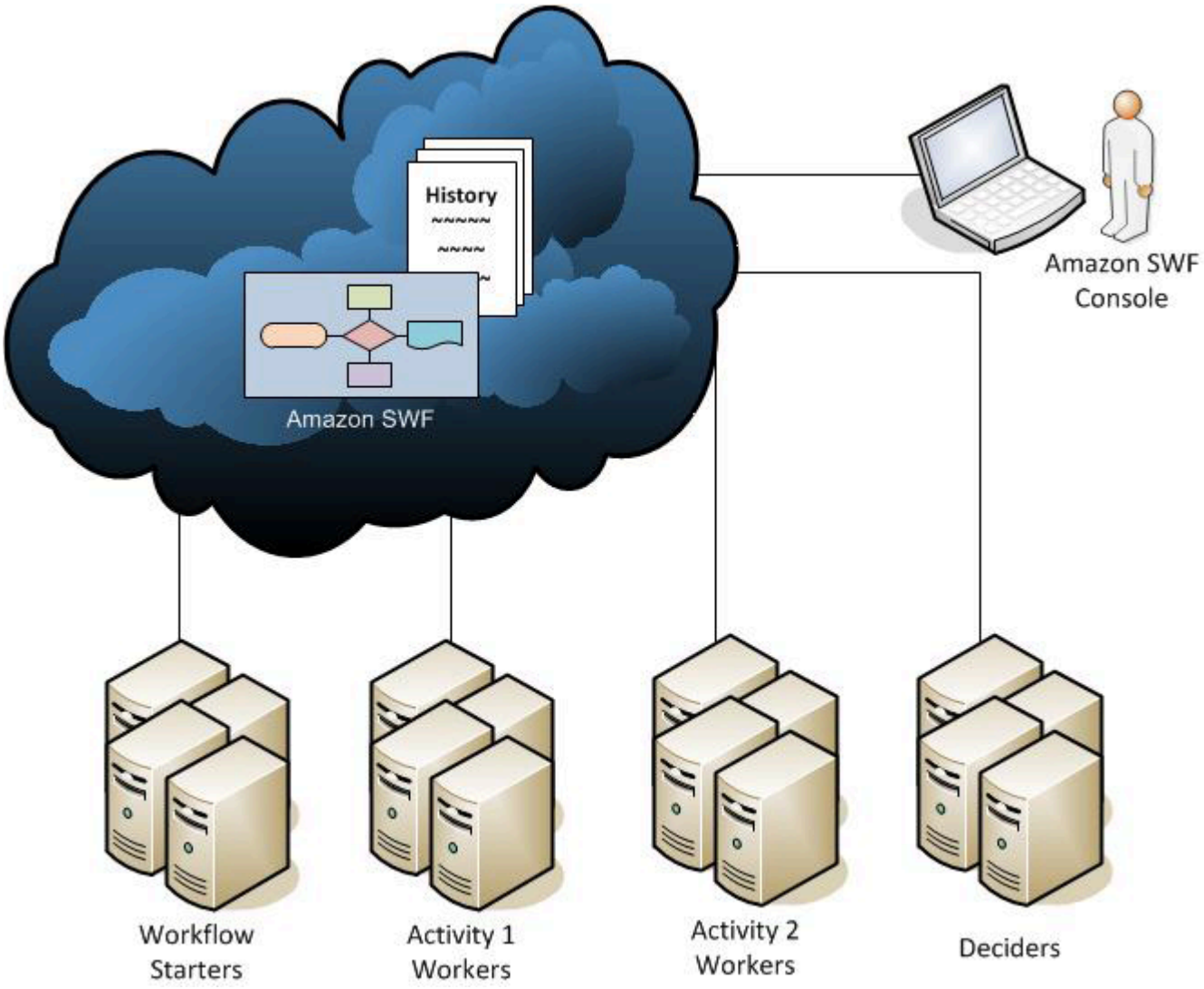
Amazon SWF のアクター

トピック

Amazon SWF のアクターとは?

操作の過程で、Amazon SWF はさまざまな種類のプログラマティックアクターと対話します。アクターはワークフロースターター、ディサイダー、または、アクティビティワーカーになれます。これらのアクターは API を通して Amazon SWF と通信します。任意のプログラミング言語でこれらのアクターを開発できます。

次の図は、Amazon SWF とアクターを含む Amazon SWF アーキテクチャーを示しています。



ワークフロースターター

ワークフロースターターとは、ワークフロー実行を開始できるアプリケーションのことです。e コマースの例では、1 つのワークフロースターターはウェブサイトで、顧客はそこで注文を行います。別のワークフロースターターとしてモバイルアプリケーションやシステムがあり、カスタマーサービス担当者はそれを使用して顧客に代わって注文を行います。

ディサイダー

ディサイダーは、ワークフローの調整ロジックの実装です。ディサイダーは、ワークフロー実行でアクティビティタスクの流れを制御します。ワークフロー実行中に、タスクの完了などの変更が生じると、全体のワークフロー履歴を含む決定タスクがディサイダーに渡されます。ディサイダーは決定タスクを Amazon SWF から受け取ると、ワークフロー実行の履歴を分析し、ワークフロー

実行の次の適切なステップを決定します。ディサイダーは、decisions を使用してこれらのステップを Amazon SWF に通知します。decisions とは、さまざまな次のアクションを表すことができる Amazon SWF のデータ型です。可能な決定のリストについては、「Amazon Simple Workflow Service API Reference」(Amazon Simple Workflow Service API リファレンス) の「決定」を参照してください。

以下に示しているのは JSON 形式の決定の例で、この形式で Amazon SWF へ伝達されます。この決定により新しいアクティビティタスクがスケジュールされます。

```
{
   "decisionType" : "ScheduleActivityTask",
   "scheduleActivityTaskDecisionAttributes" : {
      "activityType" : {
         "name" : "activityVerify",
         "version" : "1.0"
      },
      "activityId" : "verification-27",
      "control" : "digital music",
      "input" : "5634-0056-4367-0923,12/12,437",
      "scheduleToCloseTimeout" : "900",
      "taskList" : {
         "name": "specialTaskList"
      },
      "scheduleToStartTimeout" : "300",
      "startToCloseTimeout" : "600",
      "heartbeatTimeout" : "120"
   }
}
```

ディサイダーは、ワークフロー実行が開始しワークフロー実行で状態変更が発生するたびに、決定タスクを受け取ります。ディサイダーは、決定タスクを受信し、より多くの決定で Amazon SWF に応答することにより、ワークフロー実行が完了したとディサイダーが決めるまで、ワークフロー実行を進行し続けます。その時は、ワークフロー実行をクローズするようにとの決定で応答します。ワークフロー実行がクローズされた後、Amazon SWF はその実行に追加のタスクをスケジュールしません。

e コマースの例では、ディサイダーは各ステップが正しく実行されたかどうかを判断した後、次のステップをスケジュールするか、またはエラー状態を管理します。

ディサイダーは 1 つのコンピュータプロセスまたはスレッドを表します。複数のディサイダーは、同じワークフロータイプのタスクを処理できます。

アクティビティワーカー

アクティビティワーカーは、ワークフローの一部であるアクティビティタスクを実行するプロセスまたはスレッドです。アクティビティタスクは、アプリケーションで識別したタスクの 1 つを表します。

ワークフローでアクティビティタスクを使用するには、Amazon SWF コンソールまたは RegisterActivityType アクションのいずれかを使用して登録する必要があります。

各アクティビティワーカーは、そのアクティビティワーカーに適切な新しいタスクに対して Amazon SWF をポーリングします。特定のタスクは特定のアクティビティワーカーでのみ実行できます。タスクを受け取ったら、アクティビティワーカーはタスクを完了するまで処理し、タスクが完了したことを Amazon SWF にレポートし結果を提供します。その後、アクティビティワーカーは新しいタスクに対してポーリングします。ワークフロー実行に関連付けられたアクティビティワーカーはこのように続き、ワークフロー実行自体が完了するまでタスクを処理します。e コマースの例では、アクティビティワーカーは、クレジットカードの処理者や倉庫の従業員などの人々により使用される独立したプロセスやアプリケーションで、プロセスの中で個々のステップを実行します。

アクティビティワーカーは 1 つのコンピュータプロセス (またはスレッド) を表します。複数のアクティビティワーカーは同じアクティビティタイプのタスクを処理できます。

アクター間のデータ交換

入力データはワークフロー実行が開始するときに提供できます。同様に、入力データは、アクティビティワーカーがアクティビティタスクをスケジュールするときにアクティビティワーカーに提供できます。アクティビティタスクが完了すると、アクティビティワーカーは Amazon SWF に結果を返すことができます。同様に、ディサイダーは、ワークフロー実行が完了するとワークフロー実行の結果をレポートできます。各アクターは、文字列で Amazon SWF にデータを送信する、またはそれからデータを受信することができ、文字列のフォームはユーザーが定義します。データのサイズと機密性により、データを直接渡すか、または、別のシステムやサービス (Amazon S3 または DynamoDB など) に保存されているデータへのポインタを渡すことができます。直接渡されたデータおよび別のデータ保存へのポインタのいずれも、ワークフロー実行の履歴に記録されます。ただし、Amazon SWF は履歴の一部として外部保存からのデータをコピーする、またはキャッシュすることはしません。

Amazon SWF は、入力やタスクの結果を含め、各ワークフロー実行の完全な実行状態を維持するため、すべてのアクターはステートレスにできます。結果として、ワークフロー処理はスケーラビリティに優れています。システム負荷の増加に合わせて、容量を増やすために単にアクターを追加することができます。

Amazon SWF のタスク

Amazon SWF はタスクと呼ばれる作業割り当てを提供することによってアクティビティワーカーとディサイダーを連携させます。Amazon SWF には 3 種類の タスクがあります。

- アクティビティタスク – アクティビティ タスクはアクティビティワーカーに、インベントリの確認やクレジットカードへの請求などの機能を実行するよう指示します。アクティビティタスクには、アクティビティワーカーがその機能を実行するために必要なすべての情報が含まれています。
- Lambda タスク – Lambda タスクはアクティビティタスクに似ていますが、従来の Amazon SWF アクティビティの代わりに Lambda 関数を実行します。Lambda タスクを定義する方法の詳細については、「AWS Lambda Amazon SWF のタスク」を参照してください。
- 決定タスク - 決定 タスクは、ディサイダーが実行する必要のある次のアクティビティを決定できるように、ワークフロー実行の状態が変更されたことをディサイダーに通知します。決定タスクには、現在のワークフロー履歴が含まれます。

Amazon SWF は、ワークフローの開始時や、アクティビティタスクの完了時など、ワークフローの状態が変化するたびに、決定タスクをスケジュールします。各決定タスクには、ワークフロー実行履歴全体のページ分割された表示が含まれています。ディサイダーはワークフローの実行履歴を分析し、ワークフローの実行で次に何が発生するかを指定する一連の決定を使用して Amazon SWF に応答します。基本的に、すべての決定タスクでディサイダーにはワークフローを評価し、Amazon SWF に指示を返す機会があります。

競合する決定が処理されないように、Amazon SWF は各決定タスクを正確に 1 つのディサイダーに割り当て、ワークフロー実行中に一度に 1 つの決定タスクのみをアクティブにします。

次の表に、ワークフローと決定に関連するさまざまな構成要素の関係を示します。

論理設計	登録内容	実行者	受信と実行	生成
ワークフロー	ワークフロータイプ	ディサイダー	決定タスク	決定

アクティビティワーカーがアクティビティタスクを完了すると、そのタスクが完了したことが Amazon SWF に報告されます。それには生成された関連する結果がすべて含まれます。Amazon SWF は、ワークフローの実行履歴を、タスクが完了したことを示すイベントで更新し、更新された履歴を決定者に送信する決定タスクをスケジュールします。

Amazon SWF は、各アクティビティタスクを正確に 1 人のアクティビティワーカーに割り当てます。タスクが割り当てられたら、他のアクティビティワーカーはそのタスクを要求または実行できません。

次の表に、アクティビティに関連するさまざまな構成要素の関係を示します。

論理設計	登録内容	実行者	受信と実行	生成
アクティビティ	アクティビティタイプ	アクティビティワーカー	アクティビティタスク	結果データ

Amazon SWF のタスクリスト

タスクリストを使用すると、ワークフローに関連付けられたさまざまなタスクを整理することができます。タスクリストは、動的クエリと類似したものと考えることができます。タスクが Amazon SWF でスケジュールされた場合、これを配置するキュー (タスクリスト) を指定できます。同様に、タスクに対して Amazon SWF をポーリングする場合、タスクの取得元となるキュー (タスクリスト) を指定します。

タスクリストは、ユースケースでの必要性に応じて、タスクをワーカーにルーティングするための柔軟なメカニズムを提供します。タスクリストは、タスクリストを登録したり、アクションを通じて明示的に作成したりする必要がないという点で動的です。タスクをスケジュールするだけで、既に存在していない場合はタスクリストが作成されます。

アクティビティタスクや決定タスクには別のリストがあります。タスクは常に 1 つのタスクリストでのみスケジュールされ、リスト間で共有されることはありません。さらに、アクティビティやワークフローと同様に、タスクリストは特定の AWS リージョンと Amazon SWF ドメインに限定されます。

トピック

- 決定タスクリスト
- アクティビティタスクリスト
- タスクのルーティング

決定タスクリスト

各ワークフロー実行は、特定の決定タスクリストに関連付けられます。ワークフロータイプ (RegisterWorkflowType アクション) が登録されたら、そのワークフロータイプの実行のデフォルトのタスクリストを指定できます。ワークフロースターターがワークフロー実行 (StartWorkflowExecution アクション) を開始する際に、そのワークフロー実行に対して異なるタスクリストを指定するオプションがあります。

ディサイダーが新しい決定タスク (PollForDecisionTask アクション) をポーリングするときに、ディサイダーは使用する決定タスクリストを指定します。1 つのディサイダーは、呼び出しごとに別のタスクリストを使用し、PollForDecisionTask を複数回呼び出して複数のワークフロー実行に対応できます。各タスクリストは特定のワークフロー実行に固有です。または、ディサイダーは複数のワークフロー実行の決定タスクを提供する 1 つの決定タスクリストをポーリングできます。また、すべてがそのワークフロー実行のタスクリストをポーリングすることによって、1 つのワークフロー実行に対応する複数のディサイダーを持つことができます。

アクティビティタスクリスト

1 つのアクティビティタスクリストに、異なるアクティビティタイプのタスクを含めることができます。タスクはタスクリストに順番にスケジュールされます。Amazon SWF はベストエフォートベースで、順番にリストからタスクを返します。状況によっては、タスクが順番にリストから返されない場合があります。

アクティビティタイプを登録する (RegisterActivityType アクション) ときに、そのアクティビティタイプのデフォルトのタスクリストを指定できます。デフォルトでは、このタイプのアクティビティタスクは指定されたタスクリストでスケジュールされます。ただし、ディサイダーがアクティビティタスクをスケジュールする (ScheduleActivityTask 決定) ときに、ディサイダーはオプションでタスクをスケジュールする別のタスクリストを指定できます。ディサイダーがタスクリストを指定しない場合は、デフォルトのタスクリストが使用されます。その結果、タスクの属性に従って特定のタスクリストにアクティビティタスクを配置できます。たとえば、特定のタスクリストに、特定のクレジットカードタイプのアクティビティタスクのすべてのインスタンスを配置できます。

タスクのルーティング

アクティビティワーカーが新しいタスクをポーリングする (PollForActivityTask アクション) ときに、使用するアクティビティタスクリストを指定できます。その場合、アクティビティワーカーは、そのリストからのみタスクを受け取ります。このようにして、特定のタスクを特定のアクティビティワーカーにのみ割り当てることができます。たとえば、高性能なコンピュータの使用を必要とするタス

クを保持するタスクリストを作成するとします。適切なハードウェアで実行しているアクティビティワーカーのみが、そのタスクリストをポーリングします。別の例として、特定の地域的リージョンのタスクリストを作成するとします。その場合、そのリージョンにデプロイされたワーカーのみが、それらのタスクを選択するようにできます。または、優先順位が高い注文のタスクリストを作成し、常に最初にそのリストを確認するようにできます。

この方法で特定のアクティビティワーカーに特定のタスクを割り当てることを、タスクのルーティングと呼びます。タスクのルーティングはオプションです。アクティビティタスクをスケジュールするときにタスクリストを指定しない場合、タスクはデフォルトのタスクリストに自動的に配置されます。

Amazon SWF でのワークフロー実行の終了

ワークフロー実行を開始すると、その実行は開かれます。開いているワークフロー実行を完了済み、キャンセル済み、失敗、またはタイムアウトとしてクローズできます。新しい実行として続行するか、終了することもできます。ワークフロー実行は、ディサイダー、ワークフローの管理者、または Amazon SWF によってクローズできます。

ワークフローのアクティビティが終了したとディサイダーが判断した場合、RespondDecisionTaskCompleted アクションを使用し、CompleteWorkflowExecution 決定を渡して、ワークフロー実行を完了済みとしてクローズする必要があります。

または、ディサイダーがワークフロー実行をキャンセル済みまたは失敗としてクローズできます。実行をキャンセルするには、ディサイダーは RespondDecisionTaskCompleted アクションを使用して CancelWorkflowExecution 決定を渡す必要があります。

ディサイダーが通常の完了以外の状態になった場合、ワークフロー実行は失敗する必要があります。実行が失敗するためには、ディサイダーは RespondDecisionTaskCompleted アクションを使用して FailWorkflowExecution 決定を渡す必要があります。

Amazon SWF はワークフロー実行をモニタリングし、ユーザーが指定したタイムアウト設定を超えないようにします。ワークフロー実行時間がタイムアウトすると、Amazon SWF は自動的にそれをクローズします。このタイムアウト値の詳細については、「Amazon SWF タイムアウトの種類」セクションを参照してください。

また、ディサイダーは、実行をクローズし、RespondDecisionTaskCompleted アクションを使用して ContinueAsNewWorkflowExecution 決定を渡すことにより、新しい実行として論理的に続行することもできます。これは、時間の経過とともに履歴が大きくなりすぎる場合がある、長時間実行されるワークフロー実行には有効な戦略です。

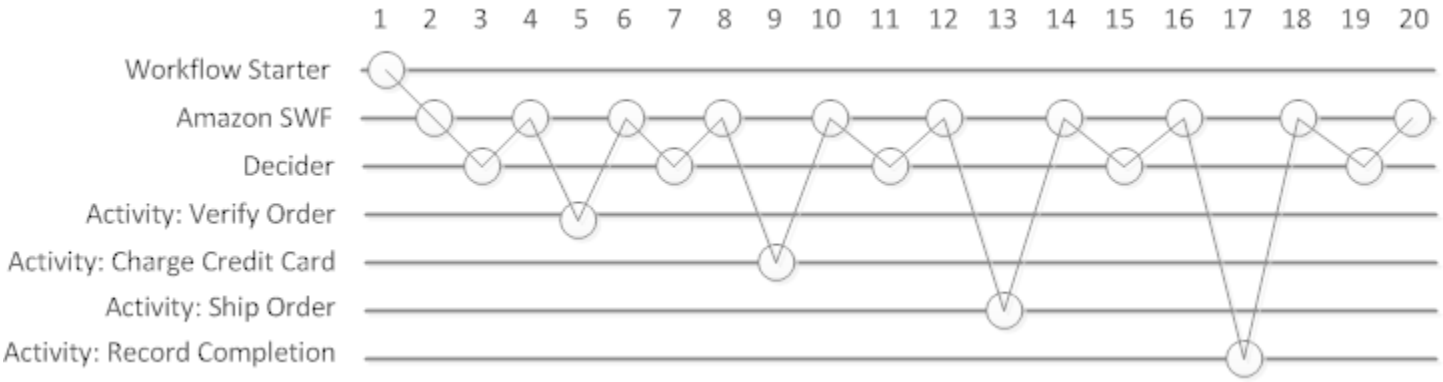
最後に、Amazon SWF コンソールから、または TerminateWorkflowExecution API を使用してプログラムで、ワークフロー実行を直接終了できます。終了により、ワークフロー実行が強制クローズされます。ディサイダーはワークフロー実行のクローズを管理できるため、終了よりもキャンセルが推奨されます。

Amazon SWF は、実行が特定のサービス定義の制限を超えた場合、ワークフローの実行を終了します。Amazon SWF は、親ワークフローが終了し、該当する子ポリシーで子ワークフローも終了する必要があることが示されている場合、子ワークフローを終了します。

Amazon SWF ワークフローのライフサイクル

Amazon SWF は、ワークフロー実行の開始から完了まで適切なタスク (アクティビティタスクまたは決定タスク) を割り当てて、アクターとやり取りします。

次の図は、機能するコンポーネントの観点から、注文処理ワークフロー実行のライフサイクルを示しています。



ワークフロー実行のライフサイクル

次の表では、前述のイメージの各タスクについて説明します。

説明	アクション、決定、またはイベント
1. ワークフロースターターは適切な Amazon SWF アクションを呼び出して、注文のワークフロー実行を開始し、	StartWorkflowExecution アクション

説明	アクション、決定、またはイベント
注文情報を提供します。	
2. Amazon SWF はワークフロー実行リクエストを受信し、最初の決定タスクをスケジュールします。	WorkflowExecutionStarted イベントと DecisionTaskScheduled イベント。
3. ディサイダーは Amazon SWF からタスクを受け取り、履歴を確認し、調整ロジックを適用して、以前にアクティビティが発生していないことを確認します。また、タスクを処理するためにアクティビティワーカーが必要とする情報で [注文の確認] アクティビティをスケジュールする決定を行い、その決定を Amazon SWF に返します。	PollForDecisionTask アクション。RespondDecisionTaskCompleted アクションと ScheduleActivityTask 決定。

説明	アクション、決定、またはイベント
4. Amazon SWF は決定を受け取り、[注文の確認] アクティビティタスクをスケジュールし、アクティビティタスクが完了するかタイムアウトするのを待ちます。	ActivityTaskScheduled イベント
5. [注文の確認] アクティビティを実行できるアクティビティワーカーは、タスクを受け取り、実行して、結果を Amazon SWF に返します。	PollForActivityTask アクションおよび RespondActivityTaskCompleted アクション。
6. Amazon SWF は [注文の確認] アクティビティの結果を受け取り、ワークフロー履歴に追加して、決定タスクをスケジュールします。	ActivityTaskCompleted イベントと DecisionTaskScheduled イベント。

説明	アクション、決定、またはイベント
7. ディサイダーは Amazon SWF からタスクを受け取り、履歴を確認し、調整ロジックを適用します。また、ChargeCreditCard アクティビティタスクを処理するためにアクティビティワーカーが必要とする情報でそのアクティビティをスケジュールする決定を行い、結果を Amazon SWF に返します。	PollForDecisionTask アクション。RespondDecisionTaskCompleted アクションと ScheduleActivityTask 決定。
8. Amazon SWF は決定を受け取り、ChargeCreditCard アクティビティタスクをスケジュールし、そのタスクが完了するかタイムアウトするのを待ちます。	DecisionTaskCompleted イベントと ActivityTaskScheduled イベント。
9. ChargeCreditCard アクティビティを実行できるアクティビティワーカーは、タスクを受け取り、実行して、結果を Amazon SWF に返します。	PollForActivityTask アクションと RespondActivityTaskCompleted アクション。

説明	アクション、決定、またはイベント
10. Amazon SWF は ChargeCreditCard アクティビティタスクの結果を受け取り、ワークフロー履歴に追加して、決定タスクをスケジュールします。	ActivityTaskCompleted イベントと DecisionTaskScheduled イベント。
11. ディサイダーは Amazon SWF からタスクを受け取り、履歴を確認し、調整ロジックを適用します。また、ShipOrder アクティビティタスクを実行するためにアクティビティワーカーが必要とする情報でそのアクティビティをスケジュールする決定を行い、結果を Amazon SWF に返します。	PollForDecisionTask アクション RespondDecisionTaskCompleted と ScheduleActivityTask 決定。
12. Amazon SWF は決定を受け取り、ShipOrder アクティビティタスクをスケジュールし、そのタスクが完了するかタイムアウトするのを待ちます。	DecisionTaskCompleted イベントと ActivityTaskScheduled イベント。

説明	アクション、決定、またはイベント
13. ShipOrder アクティビティを実行できるアクティビティワーカーは、タスクを受け取り、実行して、結果を Amazon SWF に返します。	PollForActivityTask アクションおよび RespondActivityTaskCompleted アクション。
14. Amazon SWF は ShipOrder アクティビティタスクの結果を受け取り、ワークフロー履歴に追加して、決定タスクをスケジュールします。	ActivityTaskCompleted イベントと DecisionTaskScheduled イベント。
15. ディサイダーは Amazon SWF からタスクを受け取り、履歴を確認し、調整ロジックを適用します。また、RecordCompletion アクティビティタスクを実行するためにアクティビティワーカーが必要とする情報でそのアクティビティをスケジュールする決定を行い、結果を Amazon SWF に返します。	PollForDecisionTask アクション。RespondDecisionTaskCompleted アクションと ScheduleActivityTask 決定。

説明	アクション、決定、またはイベント
16. Amazon SWF は決定を受け取り、RecordCompletion アクティビティタスクをスケジュールし、そのタスクが完了するかタイムアウトするのを待ちます。	DecisionTaskCompleted イベントと ActivityTaskScheduled イベント。
17. RecordCompletion アクティビティを実行できるアクティビティワーカーは、タスクを受け取り、実行して、結果を Amazon SWF に返します。	PollForActivityTask アクションおよび RespondActivityTaskCompleted アクション。
18。Amazon SWF は RecordCompletion アクティビティタスクの結果を受け取り、ワークフロー履歴に追加して、決定タスクをスケジュールします。	ActivityTaskCompleted イベントと DecisionTaskScheduled イベント。

説明	アクション、決定、またはイベント
19. ディサイダーが Amazon SWF からタスクを受け取り、履歴を確認し、調整ロジックを適用します。次に、ワークフロー実行をクローズする決定を行い、結果とともに決定を Amazon SWF に返します。	PollForDecisionTask アクション。RespondDecisionTaskCompleted アクションと CompleteWorkflowExecution 決定。
20。Amazon SWF はワークフロー実行をクローズし、今後のリファレンス用に履歴をアーカイブします。	WorkflowExecutionCompleted イベント。

Amazon SWF でのタスクのポーリング

ディサイダーとアクティビティワーカーは、ロングポーリングを使用して Amazon SWF と通信します。ディサイダーやアクティブワーカーは、Amazon SWF と定期的に通信してタスクを引き受けられることを Amazon SWF に知らせ、タスクの取得元のタスクリストを指定します。

指定したタスクリストのタスクが利用可能な場合は、直ちに Amazon SWF からレスポンスで返されます。利用可能なタスクがない場合は、Amazon SWF が TCP 接続を最大 60 秒間保持し、その時間内にタスクが利用可能になった場合は同じ接続中に返すことができます。60 秒以内にタスクが利用可能にならなかった場合は、空のレスポンスが返され、接続をクローズします (空のレスポンスは、taskToken が空の文字列になっているタスク構造です)。これが発生した場合、ディサイダーやアクティビティワーカーは再度ポーリングを行う必要があります。

ロングポーリングは、大量のタスク処理に適しています。ディサイダーやアクティビティワーカーは独自にキャパシティーを管理できます。キャパシティーは、ディサイダーとアクティビティワーカーがファイアウォール内にある場合に簡単に使用できます。

詳細については、「決定タスクのポーリング」および「アクティビティタスクのポーリング」を参照してください。

Amazon SWF の高度なワークフローの概念

「???」セクションの e コマースの例は、簡略化されたワークフローシナリオを表しています。実際には、ワークフロー実行の一部として、ワークフローで同時タスクを実行する (クレジットカードの認証中に注文確認メールを送信する)、主要なイベントを記録する (すべての品目をまとめる)、注文を変更で更新する (品目を追加または削除する)、その他の高度な決定を行う可能性があります。このセクションでは、ワークフローの構築に使用できる高度なワークフローの概念について説明します。

高度な概念

- バージョニング
- シグナル
- Amazon SWF の子ワークフロー
- Amazon SWF のマーカー
- Amazon SWF のタグ
- Amazon SWF による排他的な選択の実装
- Amazon SWF のタイマー
- Amazon SWF でのアクティビティタスクのキャンセル

バージョニング

ビジネスニーズにより、同じワークフローやアクティビティの異なる実装やバリエーションを同時に実行する必要がある場合が多くあります。たとえば、別の実装が実稼働中に、ワークフローの新しい実装をテストできるようにしたい場合があります。また、基本実装とプレミアム実装など、2 つの異なる機能セットを持つ 2 つの異なる実装を運用したい場合があります。バージョニングにより、要件を満たすあらゆる目的のために、ワークフローとアクティビティの複数の実装を同時に実行することができます。

ワークフローおよびアクティビティタイプには、登録時に指定されたバージョンがそれらに関連付けられています。バージョンは自由形式の文字列で、独自のバージョンスキームを選択できます。登録されているタイプの新しいバージョンを作成するには、同じ名前と異なるバージョンで登録する必要があります。前に説明した Amazon SWF のタスクリスト は、バージョニングの実装にさらに役立ちます。既に特定のタイプの長時間実行中のワークフロー実行が進行中で、新機能を追加するなどワークフローの変更が必要な状況を考えてみます。アクティビティタイプとワーカーの新しいバージョンと、新しいディサイダーを作成して、新機能を実装できます。次に、別のセットのタスクリストを使

用して、新しいワークフローバージョンの実行を開始できます。このようにして、相互に影響を与えることなく、異なるバージョンのワークフローを同時に実行することができます。

シグナル

シグナルにより、実行中のワークフロー実行に情報を挿入できます。シナリオによっては、実行中のワークフロー実行に情報を追加して、何かが変わったことを知らせたり、外部イベントについて通知したりする場合があります。任意のプロセスで、オープン状態のワークフロー実行にシグナルを送信できます。たとえば、1 つのワークフロー実行で他のワークフロー実行にシグナルを送信できます。

> **Note**
>
> オープン状態ではないワークフロー実行にシグナルを送信しようとすると、`SignalWorkflowExecution` は `UnknownResourceFault` で失敗します。

シグナルを使用するには、シグナルに渡すシグナル名とデータを定義します (ある場合)。次に、履歴でシグナルイベント (WorkflowExecutionSignaled) を認識し、適切に処理するようにディサイダーをプログラムします。プロセスがワークフローの実行を処理する場合、ターゲットとなるワークフロー実行の識別子である、シグナル名、およびシグナルデータを指定する Amazon SWF を呼び出します (SignalWorkflowExecution アクションを使用するか、ディサイダーの場合は SignalExternalWorkflowExecution 決定を使用します)。次に、Amazon SWF はシグナルを受信し、ターゲットワークフロー実行の履歴に記録し、そのシグナルに対する決定タスクをスケジュールします。ディサイダーが決定タスクを受け取るときに、ワークフロー実行履歴内でシグナルも受け取ります。次に、ディサイダーは、シグナルとそのデータに基づいて適切なアクションを実行します。

場合によっては、シグナルを待機することがあります。例えば、ユーザーはシグナルを送信して注文をキャンセルできますが、これが可能であるのは注文後 1 時間以内に限られています。Amazon SWF には、ディサイダーがサービスからのシグナルを待つことを可能にするプリミティブはありません。一時停止の機能は、ディサイダー自体で実装する必要があります。ディサイダーは、一時停止するには `StartTimer` 決定を使用してタイマーを開始します。それにより、決定タスクのポーリングを継続するためにディサイダーがシグナルを待機する期間が指定されます。ディサイダーが決定タスクを受信した場合、履歴でシグナルが受信されたかどうか、およびタイマーが起動したかどうか確認する必要があります。シグナルが受信された場合、ディサイダーはタイマーをキャンセルします。ただし、代わりにタイマーが起動した場合、指定した時間内にシグナルが到着しなかったことを意味します。まとめると、特定のシグナルを待機するには、次の操作を行います。

1. ディサイダーが待機する期間のタイマーを作成します。

2. 決定タスクを受け取ったら、履歴で、シグナルが到着したかどうか、およびタイマーが起動したか確認します。
3. シグナルが到着した場合は、CancelTimer 決定を使用してタイマーをキャンセルし、シグナルを処理します。タイミングに応じて、履歴には TimerFired および WorkflowExecutionSignaled の両方のイベントが含まれる場合があります。このような場合、履歴でのイベントの相対的な順序により、最初に発生したイベントを判断できます。
4. シグナルを受信する前にタイマーが起動した場合、ディサイダーはシグナルの待機をタイムアウトしました。実行を失敗するか、ユースケースに適している他のロジックを実行できます。

ワークフローをキャンセルするケース (注文自体が顧客によってキャンセルされたなど) では、ワークフローにシグナルを送信するのではなく、RequestCancelWorkflowExecution アクションを使用します。

シグナルの一部のアプリケーションには、以下が含まれます。

- シグナルが受信されるまで (たとえば、インベントリの出荷まで) ワークフロー実行を一時停止する。
- ディサイダーによる決定のロジックに影響する情報をワークフロー実行に提供する。これは、外部イベントによって影響を受けるワークフロー (マーケットの終了後に株式売却の完了を試みるなど) に有用です。
- 変更の発生が予期される場合に、ワークフロー実行を更新する (注文後、出荷前に注文数量が変更されるなど)。

次の例では、注文をキャンセルするためにワークフロー実行にシグナルが送信されます。

```
https://swf.us-east-1.amazonaws.com
SignalWorkflowExecution
{"domain": "867530901",
 "workflowId": "20110927-T-1",
 "runId": "f5ebbac6-941c-4342-ad69-dfd2f8be6689",
 "signalName": "CancelOrder",
 "input": "order 3553"}
```

ワークフロー実行がシグナルを受信した場合、Amazon SWF は次のような正常な HTTP レスポンスを返します。Amazon SWF はシグナルの処理をディサイダーに通知する決定タスクを生成します。

```
HTTP/1.1 200 OK
```

```
Content-Length: 0
Content-Type: application/json
x-amzn-RequestId: bf78ae15-3f0c-11e1-9914-a356b6ea8bdf
```

Amazon SWF の子ワークフロー

複雑なワークフローは、子ワークフローを使用して、より小さく管理しやすい、再利用できる可能性のあるコンポーネントに分割できます。子ワークフローは、別の (親) ワークフロー実行により開始されるワークフロー実行です。子ワークフローを開始するには、親ワークフローのディサイダーが `StartChildWorkflowExecution` の決定を使用します。この決定で指定された入力データは、履歴を通じて子ワークフローが使用できます。

`StartChildWorkflowExecution` 決定の属性では、子ポリシー も指定します。つまり、子ワークフロー実行より前に親ワークフロー実行が終了する状況を Amazon SWF がどのように処理するかを指定します。次の 3 つの可能な値があります。

- TERMINATE: Amazon SWF は子実行を終了します。
- REQUEST_CANCEL: Amazon SWF は子ワークフロー実行履歴に `WorkflowExecutionCancelRequested` イベントを配置して、子実行のキャンセルを試みます。
- ABANDON: Amazon SWF はアクションを実行しません。子実行は実行を続行します。

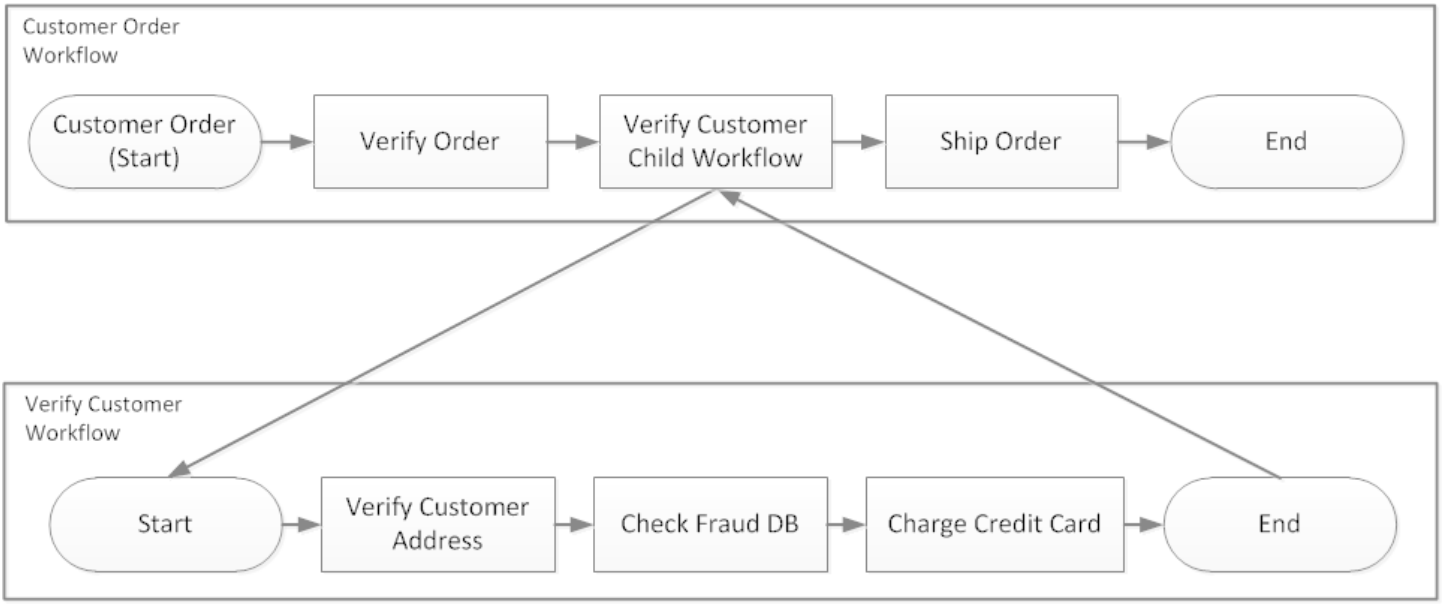
子ワークフロー実行が開始されると、通常の実行のように動作します。この処理が完了すると、Amazon SWF は親ワークフロー実行のワークフロー履歴で、結果と併せて完了を記録します。子ワークフローの例には次のようなものがあります。

- 別のウェブサイトでワークフローによって使用されるクレジットカード処理の子ワークフロー
- 顧客の E メールアドレス、オプトアウトリスト、E メールの送信、E メールがバウンスまたは失敗しなかったことの確認を行う、E メールの子ワークフロー。
- 接続、セットアップ、トランザクション、および検証を組み合わせるデータベースストレージおよび取得の子ワークフロー。
- 構築、パッケージ化、および検証を組み合わせるソースコードのコンパイルの子ワークフロー。

e コマースの例では、クレジットカードへの課金アクティビティを子ワークフローにする場合があります。これを行うには、新しい [顧客の確認] ワークフローを登録し、[顧客住所の確認] および [不正 DB の確認] アクティビティを登録して、タスクの調整ロジックを定義できま

す。次に、[顧客の注文] ワークフローのディサイダーは、このワークフロータイプを指定する StartChildWorkflowExecution 決定をスケジュールして、[顧客の確認] 子ワークフローを開始できます。

次の図は、新しい [顧客の確認] 子ワークフローを含む顧客の注文ワークフローを示しています。このワークフローでは顧客の住所および不正データベースについて確認し、クレジットカードに課金します。



複数のワークフローが、同じワークフロータイプを使用して子ワークフロー実行を作成できます。たとえば、[顧客の確認] 子ワークフローは、組織の他の部分でも使用できます。子ワークフローのイベントは独自のワークフロー履歴に含まれ、親のワークフロー履歴には含まれません。

子ワークフローはディサイダーによって開始されるワークフロー実行であるため、通常のスタンドアロンワークフロー実行として開始することもできます。

Amazon SWF のマーカー

ユースケースに固有のワークフロー実行のワークフロー履歴に情報を記録したい場合があります。マーカーにより、カスタムまたはシナリオ固有の目的に使用できる情報を、ワークフロー実行履歴に記録できます。

マーカーを使用するには、ディサイダーは RecordMarker 決定を使用し、マーカーに名前を付け、希望のデータを決定に添付し、RespondDecisionTaskCompleted アクションを使用して Amazon SWF に通知します。Amazon SWF はリクエストを受信し、ワークフロー履歴にマーカーを記録し

て、リクエストのその他の決定を有効にします。この時点から、ディサイダーはワークフロー履歴でマーカーを表示し、プログラムする任意の方法でそれを使用できます。

マーカーを記録しても、それだけでは決定タスクは開始されません。ワークフロー実行が止まらないようにするため、ワークフロー実行を続行する何かが発生する必要があります。たとえば、ディサイダーが別のアクティビティタスクをスケジュールする、ワークフロー実行がシグナルを受信する、以前にスケジュールされたアクティビティタスクが完了するなどです。

マーカーの例には次のようなものがあります。

- 再帰的なワークフローでループの数を数えるカウンタ。
- アクティビティの結果に基づく、ワークフロー実行の進行状況。
- 以前のワークフローイベントからまとめた情報。

e コマースの例として、インベントリを毎日確認し、毎回マーカーの数を増分するアクティビティを追加する場合があります。次に、その数が 5 を超えた場合に、履歴全体を確認することなく顧客に E メールを送信するか、上司に通知する決定ロジックを追加できます。

次の例では、ディサイダーが決定タスクを完了し、RecordMarker の決定を含む RespondDecisionTaskCompleted アクションで応答します。

```
https://swf.us-east-1.amazonaws.com
RespondDecisionTaskCompleted
{
  "taskToken":"12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "decisions":[{
          "decisionType":"RecordMarker",
          "recordMarkerDecisionAttributes":{
              "markerName":"customer elected special shipping offer"
          }
      },
  ]
}
```

Amazon SWF がマーカーを正常に記録すると、次のような成功 HTTP レスポンスを返します。

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: application/json
x-amzn-RequestId: 6c0373ce-074c-11e1-9083-8318c48dee96
```

Amazon SWF のタグ

Amazon SWF では、ワークフロー実行のタグ付けをサポートしています。これはリソースが多数ある場合に特に便利です。

Amazon SWF は最大 5 つまでのタグ付けとワークフロー実行をサポートします。各タグは自由形式の文字列で、最大 256 文字を使用できます。タグを使用する場合は、ワークフロー実行を開始するときに割り当てる必要があります。開始後にワークフロー実行にタグを追加することはできません。また、ワークフロー実行に割り当てられたタグを編集または削除することもできません。

IAM では、タグに基づく Amazon SWF ドメインのアクセスの制御をサポートしています。タグに基づいてアクセスを制御するには、IAM ポリシーの条件要素でリソースタグに関する情報を指定します。

タグの管理

AWS SDKs を使用するか、Amazon SWF API と直接やり取りして、Amazon Simple Workflow Service タグを管理します。API を使用すると、ドメインの登録時にタグを追加する、既存のドメインでタグを一覧表示する、および既存のドメインでタグを追加または削除することができます。

> **Note**
>
> リソースあたりのタグは 50 個に制限されています。「Amazon SWF の全般アカウントクォータ」を参照してください。

- `RegisterDomain`
- `ListTagsForResource`
- `TagResource`
- `UntagResource`

詳細については、Amazon SWF API を使用する および「Amazon Simple Workflow Service API Reference」(Amazon Simple Workflow Service API リファレンス) を参照してください。

ワークフロー実行のタグ付け

Amazon SWF を使用すると、タグをワークフロー実行に関連付け、これらのタグに基づいてワークフロー実行をクエリできます。可視性オペレーションを使用すると、listi をフィルタリングできま

す。実行に割り当てるタグを慎重に選択することで、タグを使用して意味のあるリストを提供できます。

たとえば、複数のフルフィルメントセンターを運営しているとします。タグを使用すると、特定のフルフィルメントセンターで発生するプロセスを一覧表示できます。または、顧客がさまざまなタイプのメディアファイルを変換する場合、タグは動画、オーディオ、および画像ファイルを変換するときに異なるプロセスを示している可能性があります。

StartWorkflowExecution アクション、StartChildWorkflowExecution 決定、ContinueAsNewWorkflowExecution 決定を使用して実行を開始すると、最大 5 つのタグをワークフロー実行に関連付けることができます。可視性アクションを使用してワークフロー実行を一覧表示またはカウントする場合、タグに基づいて結果をフィルタリングできます。

タグ付けを使用するには

1. タグ付け戦略を考案します。ビジネス要件について考え、適切なタグのリストを作成します。どの実行がどのタグを取得するかを決定します。実行には最大 5 つのタグを割り当てることができますが、タグライブラリには任意の数のタグを含めることができます。各タグは 256 文字までの任意の文字列値にすることができるため、タグにより、ほぼすべてのビジネスコンセプトを記述できます。
2. 作成時に最大 5 つのタグを使用して実行をタグ付けします。
3. 特定のタグでタグ付けされた実行をリストまたはカウントするには、tagFilter パラメータを ListOpenWorkflowExecutions、ListClosedWorkflowExecutions、CountOpenWorkflowExe および CountClosedWorkflowExecutions アクションで指定します。このアクションでは、指定されたタグに基づいて実行をフィルタリングします。

タグをワークフロー実行に関連付けると、タグはその実行に永続的に関連付けられ、削除することはできません。

ListWorkflowExecutions で tagFilter パラメータに指定できるタグは 1 つのみです。また、タグマッチングでは大文字と小文字が区別され、完全一致の結果のみが返されます。

次のように、タグ付けされた 2 つの実行を既に設定しているとします。

実行名	割り当てられたタグ
Execution-One	Consumer, 2011-February

実行名	割り当てられたタグ
Execution-Two	Wholesale, 2011-March

Consumer タグの `ListOpenWorkflowExecutions` によって返された実行のリストをフィルタリングできます。`oldestDate` および `latestDate` の値は Unix 時間の値として指定されます。

```
https://swf.us-east-1.amazonaws.com
  RespondDecisionTaskCompleted
  {
    "domain":"867530901",
    "startTimeFilter":{
        "oldestDate":1262332800,
        "latestDate":1325348400
    },
    "tagFilter":{
      "tag":"Consumer"
      }
  }
```

タグを使用してドメインへのアクセスを制御する

IAM で Amazon SWF ドメインに関連付けられているタグを参照することで、Amazon Simple Workflow Service ドメインへのアクセスを制御できます。

たとえば、 キーと 値を持つタグを含む Amazon SWF ドメイン`environment`を`production`次の条件で制限できます。

```
"Condition": {
    "StringEquals": {"aws:ResourceTag/environment": "production"}
}
```

詳細については、以下を参照してください。

- IAM タグを使用したアクセスの制御
- タグベースのポリシー

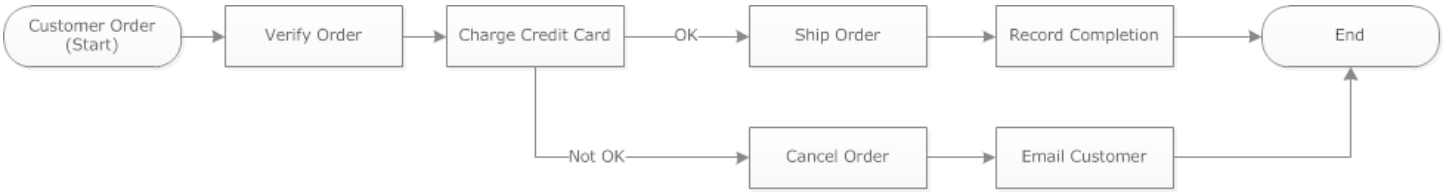
Amazon SWF による排他的な選択の実装

一部のシナリオでは、前のアクティビティの結果に基づいて、異なるアクティビティのセットをスケジュールする場合があります。排他的な選択パターンを使用すると、アプリケーションの複雑な要件を満たす柔軟なワークフローを作成できます。

Amazon SWF には、特定の排他的な選択アクションはありません。排他的な選択を実装するには、ディサイダーロジックを記述して、以前のアクティビティの結果に基づいて決定を行う必要があります。排他的な選択のアプリケーションには、以下のような例があります。

- 前のアクティビティの結果が失敗の場合にクリーンアップアクティビティを実行する
- 顧客が基本プランを購入したかアドバンストプランを購入したかに基づいて、異なるアクティビティをスケジュールする
- 顧客の注文履歴に基づいて異なる顧客認証アクティビティを実行する

e コマースの例では、クレジットカードの課金結果に基づいて注文品を発送するかキャンセルする排他的な選択を使用できます。次の図では、クレジットカードに正しく課金された場合に、ディサイダーが [注文品の発送と完了の記録] アクティビティタスクをスケジュールします。それ以外の場合、[注文のキャンセルおよび顧客への E メール送信] アクティビティタスクをスケジュールします。



クレジットカードに正常に課金された場合、ディサイダーは `ShipOrder` アクティビティをスケジュールします。それ以外の場合、ディサイダーは `CancelOrder` アクティビティをスケジュールします。

この場合、履歴を解釈し、クレジットカードが正常に課金されたかどうかを確認するようにディサイダーをプログラムします。これを行うため、次のようなロジックがあるとします。

```
IF lastEvent = "WorkflowExecutionStarted"
 addToDecisions ScheduleActivityTask(ActivityType = "VerifyOrderActivity")

ELSIF lastEvent = "ActivityTaskCompleted"
     AND ActivityType = "VerifyOrderActivity"
 addToDecisions ScheduleActivityTask(ActivityType = "ChargeCreditCardActivity")
```

```
#Successful Credit Card Charge Activities
ELSIF lastEvent = "ActivityTaskCompleted"
      AND ActivityType = "ChargeCreditCardActivity"
  addToDecisions ScheduleActivityTask(ActivityType = "ShipOrderActivity")

ELSIF lastEvent = "ActivityTaskCompleted"
     AND ActivityType = "ShipOrderActivity"
  addToDecisions ScheduleActivityTask(ActivityType = "RecordOrderCompletionActivity")

ELSIF lastEvent = "ActivityTaskCompleted"
     AND ActivityType = "RecordOrderCompletionActivity"
 addToDecisions CompleteWorkflowExecution

#Unsuccessful Credit Card Charge Activities
ELSIF lastEvent = "ActivityTaskFailed"
      AND ActivityType = "ChargeCreditCardActivity"
  addToDecisions ScheduleActivityTask(ActivityType = "CancelOrderActivity")

ELSIF lastEvent = "ActivityTaskCompleted"
     AND ActivityType = "CancelOrderActivity"
 addToDecisions ScheduleActivityTask(ActivityType = "EmailCustomerActivity")

ELSIF lastEvent = "ActivityTaskCompleted"
     AND ActivityType = "EmailCustomerActivity"
 addToDecisions CompleteWorkflowExecution

ENDIF
```

クレジットカードに正常に課金された場合、ディサイダーは RespondDecisionTaskCompleted で応答して ShipOrder アクティビティをスケジュールします。

```
https://swf.us-east-1.amazonaws.com
RespondDecisionTaskCompleted
{
  "taskToken": "12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "decisions":[
      {
          "decisionType":"ScheduleActivityTask",
          "scheduleActivityTaskDecisionAttributes":{
              "control":"OPTIONAL_DATA_FOR_DECIDER",
              "activityType":{
                  "name":"ShipOrder",
```

```
                "version":"2.4"
            },
            "activityId":"3e2e6e55-e7c4-fee-deed-aa815722b7be",
            "scheduleToCloseTimeout":"3600",
            "taskList":{
                "name":"SHIPPING"
            },
            "scheduleToStartTimeout":"600",
            "startToCloseTimeout":"3600",
            "heartbeatTimeout":"300",
            "input": "123 Main Street, Anytown, United States"
        }
    }
  ]
}
```

クレジットカードに正常に課金されなかった場合、ディサイダーは RespondDecisionTaskCompleted で応答して CancelOrder アクティビティをスケジュールします。

```
https://swf.us-east-1.amazonaws.com
RespondDecisionTaskCompleted
{
  "taskToken": "12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "decisions":[
      {
          "decisionType":"ScheduleActivityTask",
          "scheduleActivityTaskDecisionAttributes":{
              "control":"OPTIONAL_DATA_FOR_DECIDER",
              "activityType":{
                  "name":"CancelOrder",
                  "version":"2.4"
              },
              "activityId":"3e2e6e55-e7c4-fee-deed-aa815722b7be",
              "scheduleToCloseTimeout":"3600",
              "taskList":{
                  "name":"CANCELLATIONS"
              },
              "scheduleToStartTimeout":"600",
              "startToCloseTimeout":"3600",
              "heartbeatTimeout":"300",
              "input": "Out of Stock"
          }
```

```
      }
  ]
}
```

Amazon SWF が RespondDecisionTaskCompleted アクションのデータを検証できる場合、Amazon SWF は次のような正常な HTTP レスポンスを返します。

```
HTTP/1.1 200 OK
Content-Length: 11
Content-Type: application/json
x-amzn-RequestId: 93cec6f7-0747-11e1-b533-79b402604df1
```

Amazon SWF のタイマー

タイマーを使用すると、一定の時間が経過したときにディサイダーに通知できます。

ディサイダーには、決定タスクに応答するときに StartTimer 決定で応答するオプションがあります。この決定では、タイマーが起動するまでの時間を指定します。指定した時間が経過すると、Amazon SWF は TimerFired イベントをワークフロー実行履歴に追加し、決定タスクをスケジュールします。次に、ディサイダーはこの情報を使用して、さらに決定を通知できます。タイマーの 1 つの一般的なアプリケーションとして、アクティビティタスクの実行を遅延します。たとえば、顧客が品目の遅延配送を希望する場合があります。

Amazon SWF でのアクティビティタスクのキャンセル

アクティビティタスクのキャンセルは、実行する必要がなくなったアクティビティを終了するようにディサイダーに通知します。Amazon SWF は協調的なキャンセルメカニズムを使用し、実行中のアクティビティタスクを強制的に中断することはありません。キャンセルのリクエストを処理するようにアクティビティワーカーをプログラムする必要があります。

ディサイダーは、決定タスクを処理中にアクティビティタスクをキャンセルする決定ができます。アクティビティタスクをキャンセルするには、ディサイダーは RespondDecisionTaskCompleted アクションを RequestCancelActivityTask 決定とともに使用します。

アクティビティワーカーによってアクティビティタスクがまだ取得されていない場合、サービスはタスクをキャンセルします。アクティビティワーカーがいつでもタスクを取得できる競合状態が発生する可能性があることに注意してください。タスクがアクティビティワーカーに関連付けられている場合、アクティビティワーカーにはタスクのキャンセルがリクエストされます。

この例では、注文をキャンセルするためにワークフロー実行にシグナルが送信されます。

```
https://swf.us-east-1.amazonaws.com
SignalWorkflowExecution
{"domain": "867530901",
 "workflowId": "20110927-T-1",
 "runId": "9ba33198-4b18-4792-9c15-7181fb3a8852",
 "signalName": "CancelOrder",
 "input": "order 3553"}
```

ワークフロー実行がシグナルを受信した場合、Amazon SWF は次のような正常な HTTP レスポンスを返します。Amazon SWF はシグナルの処理をディサイダーに通知する決定タスクを生成します。

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: application/json
x-amzn-RequestId: 6c0373ce-074c-11e1-9083-8318c48dee96
```

ディサイダーが決定タスクを処理し、履歴のシグナルを確認すると、ディサイダーは ShipOrderActivity0001 アクティビティ ID がある未処理のアクティビティのキャンセルを試みます。アクティビティタスクのスケジュールイベントから、ワークフロー履歴でアクティビティ ID が提供されます。

```
https://swf.us-east-1.amazonaws.com
RespondDecisionTaskCompleted
{
  "taskToken":"12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "decisions":[{
          "decisionType":"RequestCancelActivityTask",
          "RequestCancelActivityTaskDecisionAttributes":{
              "ActivityID":"ShipOrderActivity0001"
          }
      }
  ]
}
```

Amazon SWF がキャンセルのリクエストを正常に受け取ると、次のような成功 HTTP レスポンスを返します。

```
HTTP/1.1 200 OK
Content-Length: 0
```

```
Content-Type: application/json
x-amzn-RequestId: 6c0373ce-074c-11e1-9083-8318c48dee96
```

キャンセルの試行は ActivityTaskCancelRequested イベントとして履歴に記録されます。

ActivityTaskCanceled イベントによって示されるようにタスクが正常にキャンセルされた場合、ワークフロー実行のクローズなど、タスクのキャンセルに続く適切なステップを実行するようにディサイダーをプログラムします。

タスクがキャンセルされずに完了、失敗、タイムアウトするなど、アクティビティタスクをキャンセルできなかった場合、ディサイダーはアクティビティの結果を受け入れるか、ユースケースで必要となるクリーンアップまたは軽減を実行する必要があります。

アクティビティタスクがアクティビティワーカーによって取得された場合、キャンセルのリクエストは、タスクのハートビートメカニズムを通して送信されます。アクティビティワーカーは定期的に RecordActivityTaskHeartbeat を使用して、タスクがまだ進行中であることを Amazon SWF に報告できます。

アクティビティワーカーにハートビートは必要ありませんが、長時間実行されるタスクには推奨されます。タスクのキャンセルでは、定期的なハートビートを記録する必要があり、ワーカーがハートビートしない場合、タスクはキャンセルできません。

ディサイダーがタスクのキャンセルをリクエストする場合、Amazon SWF は cancelRequest オブジェクトの値を true に設定します。cancelRequest オブジェクトは、RecordActivityTaskHeartbeat に応じてサービスによって返される ActivityTaskStatus オブジェクトの一部です。

Amazon SWF によって、キャンセルがリクエストされたアクティビティタスクが正常に完了できなくなることはありません。キャンセルリクエストを処理する方法は、アクティビティによって決まります。要件に応じて、アクティビティタスクをキャンセルするか、キャンセルリクエストを無視するようにアクティビティワーカーをプログラムします。

アクティビティワーカーで、アクティビティタスクのワークがキャンセルされたことを示すには、RespondActivityTaskCanceled で応答するようにプログラムします。アクティビティワーカーでタスクを完了する場合は、標準の RespondActivityTaskCompleted で応答するようにプログラムします。

Amazon SWF が RespondActivityTaskCompleted または RespondActivityTaskCanceled リクエストを受け取ると、ワークフロー実行履歴を更新し、ディサイダーに通知するように決定タスクをスケジュールします。

決定タスクを処理し、追加の決定を返すようにディサイダーをプログラムします。アクティビティタスクが正常にキャンセルされた場合、続行するために必要なタスクを実行するか、ワークフロー実行をクローズするようにディサイダーをプログラムします。アクティビティタスクが正常にキャンセルされない場合は、結果を受け入れる、結果を無視する、または必要なクリーンアップをスケジュールするようにディサイダーをプログラムします。

Amazon Simple Workflow Service のセキュリティ

このセクションでは、Amazon Simple Workflow Service のセキュリティと認証について説明します。

トピック

- Amazon Simple Workflow Service におけるデータ保護
- Amazon Simple Workflow Service での Identity and Access Management
- ログ記録とモニタリング
- Amazon Simple Workflow Service のコンプライアンス検証
- Amazon Simple Workflow Service の耐障害性
- Amazon Simple Workflow Service でのインフラストラクチャセキュリティ
- Amazon Simple Workflow Service での設定と脆弱性の分析

Amazon SWF は IAM を使用して、他の AWS サービスやリソースへのアクセスを制御します。IAM の仕組みの概要については、「IAM ユーザーガイド」の「アクセス管理の概要」を参照してください。セキュリティ認証情報の概要については、「Amazon Web Services 全般のリファレンス」の「AWS セキュリティ認証情報」を参照してください。

Amazon Simple Workflow Service におけるデータ保護

責任 AWS 共有モデル、Amazon Simple Workflow Service でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「 AWS のサービス 」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された AWS 責任共有モデルおよび GDPR のブログ記事を参照してください。

データ保護の目的で、認証情報を保護し AWS アカウント 、 AWS IAM アイデンティティセンターまたは AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-3 検証済み暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報を、タグ、または [名前] フィールドなどの自由形式のテキストフィールドに含めないことを強くお勧めします。これは、コンソール、API、または SDK を使用して Amazon SWF AWS CLIまたは他の AWS のサービス を使用する場合も同様です。 AWS SDKs タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

Amazon Simple Workflow Service での暗号化

保管中の暗号化

Amazon SWF は、保管中のデータを常に暗号化します。Amazon Simple Workflow Service のデータは、透過的なサーバー側の暗号化を使用して保存時に暗号化されます。これは、機密データの保護における負担と複雑な作業を減らすのに役立ちます。保管時に暗号化することで、セキュリティを重視したアプリケーションを構築して、暗号化のコンプライアンスと規制の要件を満たすことができます。

転送中の暗号化

Amazon SWF とその他のサービス間を通過するすべてのデータは、Transport layer Security (TLS) を使用して暗号化されています。

Amazon Simple Workflow Service での Identity and Access Management

Amazon SWF にアクセスするには、 がリクエストの認証 AWS に使用できる認証情報が必要です。これらの認証情報には、他の AWS リソースからイベントデータを取得するなど、 AWS リソースにアクセスするためのアクセス許可が必要です。次のセクションでは、AWS Identity and Access Management (IAM) と Amazon SWF を使用して、リソースにアクセスできるユーザーを制御することで、リソースをセキュリティで保護する方法について詳しく説明します。

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインインを許可) し、誰に Amazon SWF リソースの使用を承認する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

- Amazon SWF IAM ポリシー
- API の要約
- タグベースのポリシー
- Amazon SWF 用の Amazon VPC エンドポイント
- Amazon Simple Workflow Service アイデンティティとアクセスのトラブルシューティング

オーディエンス

AWS Identity and Access Management (IAM) の使用方法は、ロールによって異なります。

- サービスユーザー - 機能にアクセスできない場合は、管理者にアクセス許可をリクエストします (「Amazon Simple Workflow Service アイデンティティとアクセスのトラブルシューティング」を参照)。
- サービス管理者 - ユーザーアクセスを決定し、アクセス許可リクエストを送信します (「Amazon Simple Workflow Service で IAM を使用する方法」を参照)
- IAM 管理者 - アクセスを管理するためのポリシーを作成します (「Amazon Simple Workflow Service のアイデンティティベースのポリシーの例」を参照)

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。、IAM ユーザー AWS アカウントのルートユーザー、または IAM ロールを引き受けることで認証される必要があります。

AWS IAM アイデンティティセンター (IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーティッド ID としてサインインできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「AWS アカウントにサインインする方法」を参照してください。

プログラムによるアクセスの場合、 は SDK と CLI AWS を提供してリクエストを暗号化して署名します。詳細については、「IAM ユーザーガイド」の「API リクエストに対するAWS 署名バージョン 4」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、すべての AWS のサービス および リソースへの完全なアクセス権を持つ AWS アカウント ルートユーザーと呼ばれる 1 つのサインインアイデンティティから始

めます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザー認証情報を必要とするタスクについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーテッドアイデンティティ

ベストプラクティスとして、人間のユーザーが一時的な認証情報 AWS のサービス を使用して にアクセスするには、ID プロバイダーとのフェデレーションを使用する必要があります。

フェデレーション ID は、エンタープライズディレクトリ、ウェブ ID プロバイダー、または ID Directory Service ソースの認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーテッドアイデンティティは、一時的な認証情報を提供するロールを引き受けます。

アクセスを一元管理する場合は、 AWS IAM アイデンティティセンターをお勧めします。詳細については、「AWS IAM アイデンティティセンター ユーザーガイド」の「IAM アイデンティティセンターとは」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、特定の個人やアプリケーションに対する特定のアクセス許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用することをお勧めします。詳細については、IAM ユーザーガイドの「ID プロバイダーとのフェデレーションを使用して にアクセスすることを人間のユーザーに要求する AWS」を参照してください。

IAM グループは、IAM ユーザーの集合を指定し、大量のユーザーに対するアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、特定のアクセス許可を持つアイデンティであり、一時的な認証情報を提供します。ユーザーから IAM ロール (コンソール) に切り替えるか、 または API オペレーションを呼び出すことで、ロールを引き受けることができます。 AWS CLI AWS 詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーのアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行するアプリケーションに役立ちます。詳細については、IAM ユーザーガイド の IAM でのクロスアカウントリソースアクセス を参照してください。

ポリシーを使用したアクセスの管理

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは、ID またはリソースに関連付けられたときにアクセス許可を定義します。 は、プリンシパルがリクエストを行うときにこれらのポリシー AWS を評価します。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

管理者は、ポリシーを使用して、どのプリンシパルがどのリソースに対して、どのような条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールにアクセス許可はありません。IAM 管理者は IAM ポリシーを作成してロールに追加し、このロールをユーザーが引き受けられるようにします。IAM ポリシーは、オペレーションの実行方法を問わず、アクセス許可を定義します。

アイデンティティベースのポリシー

アイデンティティベースのポリシーは、アイデンティティ (ユーザー、グループ、またはロール) にアタッチできる JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティがどのリソースに対してどのような条件下でどのようなアクションを実行できるかを制御します。アイデンティティベースポリシーの作成方法については、IAM ユーザーガイド の カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する を参照してください。

アイデンティティベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーとインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。例としては、IAM ロール信頼ポリシーや Amazon S3 バケットポリシーなどがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

その他のポリシータイプ

AWS は、より一般的なポリシータイプによって付与されるアクセス許可の最大数を設定できる追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできるアクセス許可の数の上限を設定します。詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可境界」を参照してください。
- サービスコントロールポリシー (SCP) - AWS Organizations内の組織または組織単位の最大のアクセス許可を指定します。詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー」を参照してください。
- リソースコントロールポリシー (RCP) – は、アカウント内のリソースで利用できる最大数のアクセス許可を定義します。詳細については、「AWS Organizations ユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡される高度なポリシーです。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成されるアクセス許可を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

アクセスコントロール

リクエストを認証するために有効な認証情報を持つことができますが、アクセス許可を持っていなければ Amazon SWF リソースを作成またはアクセスすることはできません。例えば、Amazon SWF ルールに関連付けられた AWS Lambda、、Amazon Simple Notification Service (Amazon SNS)、および Amazon Simple Queue Service (Amazon SQS) ターゲットを呼び出すためのアクセス許可が必要です。

以下のセクションでは、Amazon SWF のアクセス許可を管理する方法について説明します。最初に概要のセクションを読むことをお勧めします。

- 基本的な原則
- Amazon SWF IAM ポリシー

- Amazon SWF のポリシーの記述

Amazon SWF のポリシーアクション

ポリシーアクションのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

JSON ポリシーの `Action` 要素にはポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。このアクションは関連付けられたオペレーションを実行するためのアクセス許可を付与するポリシーで使用されます。

Amazon SWF アクションのリストを確認するには、「サービス認可リファレンス」の「Amazon Simple Workflow Service で定義されるリソース」を参照してください。

Amazon SWF のポリシーアクションは、アクションの前に次のプレフィックスを使用します。

```
swf
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
      "swf:action1",
      "swf:action2"
         ]
```

Amazon SWF のアイデンティティベースポリシーの例を確認するには、「Amazon Simple Workflow Service のアイデンティティベースのポリシーの例」を参照してください。

Amazon SWF のポリシーリソース

ポリシーリソースのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Resource` JSON ポリシー要素はアクションが適用されるオブジェクトを指定します。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。リソースレベ

ルのアクセス許可をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

Amazon SWF リソースタイプとその ARNs「Amazon Simple Workflow Service で定義されるアクション」を参照してください。 どのアクションで各リソースの ARN を指定できるかについては、「Amazon Simple Workflow Service で定義されるリソース」を参照してください。

Amazon SWF のアイデンティティベースポリシーの例を確認するには、「Amazon Simple Workflow Service のアイデンティティベースのポリシーの例」を参照してください。

Amazon SWF のポリシー条件キー

サービス固有のポリシー条件キーのサポート: あり

管理者は JSON AWS ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースに対してどのような条件下でアクションを実行できるかということです。

`Condition` 要素は、定義された基準に基づいてステートメントが実行される時期を指定します。イコールや未満などの条件演算子を使用して条件式を作成して、ポリシーの条件とリクエスト内の値を一致させることができます。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「グローバル条件コンテキストキー」を参照してください。

Amazon SWF 条件キーのリストを確認するには、「サービス認可リファレンス」の「Amazon Simple Workflow Service の条件キー」を参照してください。条件キーを使用できるアクションとリソースについては、「Amazon Simple Workflow Service で定義されるリソース」を参照してください。

Amazon SWF のアイデンティティベースポリシーの例を確認するには、「Amazon Simple Workflow Service のアイデンティティベースのポリシーの例」を参照してください。

Amazon SWF での ACL

ACL のサポート: なし

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするためのアクセス許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon SWF での ABAC

ABAC (ポリシー内のタグ) のサポート: 一部

属性ベースのアクセスコントロール (ABAC) は、タグと呼ばれる属性に基づいてアクセス許可を定義する認可戦略です。IAM エンティティと AWS リソースにタグをアタッチし、プリンシパルのタグがリソースのタグと一致するときにオペレーションを許可するように ABAC ポリシーを設計できます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値はありです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、「IAM ユーザーガイド」の「ABAC 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順を説明するチュートリアルについては、「IAM ユーザーガイド」の「属性ベースのアクセスコントロール (ABAC) を使用する」を参照してください。

Amazon SWF での一時的な認証情報の使用

一時的な認証情報のサポート: あり

一時的な認証情報は、 AWS リソースへの短期的なアクセスを提供し、フェデレーションまたはスイッチロールの使用時に自動的に作成されます。長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成 AWS することをお勧めします。詳細については、「IAM ユーザーガイド」の「IAM の一時的な認証情報」および「AWS のサービス と IAM との連携」を参照してください。

Amazon SWF のクロスサービスプリンシパル許可

転送アクセスセッション (FAS) のサポート: あり

転送アクセスセッション (FAS) は、 を呼び出すプリンシパルのアクセス許可と AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストをリクエストする を使用します。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

Amazon SWF のサービスロール

サービスロールのサポート: あり

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、IAM ユーザーガイドの AWS のサービスに許可を委任するロールを作成するを参照してください。

> **Warning**
>
> サービスロールの許可を変更すると、Amazon SWF の機能が破損する可能性があります。Amazon SWF が指示する場合以外は、サービスロールを編集しないでください。

Amazon SWF のサービスにリンクされたロール

サービスにリンクされたロールのサポート: なし

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールのアクセス許可を表示できますが、編集することはできません。

サービスにリンクされたロールの作成または管理の詳細については、「IAM と提携するAWS のサービス」を参照してください。表の「サービスリンクロール」列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

Amazon SWF のアイデンティティベースの ポリシー

アイデンティティベースのポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 許可ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。JSON ポリシーで使用できるすべての要素について学ぶには、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

Amazon SWF のアイデンティティベースのポリシー例

Amazon SWF のアイデンティティベースポリシーの例を確認するには、「Amazon Simple Workflow Service のアイデンティティベースのポリシーの例」を参照してください。

Amazon SWF 内のリソースベースのポリシー

リソースベースのポリシーのサポート: なし

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーで、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、全体のアカウント、または別のアカウントの IAM エンティティを、リソースベースのポリシーのプリンシパルとして指定します。詳細については、IAM ユーザーガイドのIAM でのクロスアカウントリソースアクセスを参照してください。

Amazon Simple Workflow Service で IAM を使用する方法

IAM を使用して Amazon SWF へのアクセスを管理する前に、Amazon SWF で使用できる IAM 機能について理解しておく必要があります。

Amazon Simple Workflow Service で使用できる IAM の機能

IAM 機能	Amazon SWF のサポート
アイデンティティベースのポリシー	あり
リソースベースのポリシー	なし
ポリシーアクション	あり

IAM 機能	Amazon SWF のサポート
ポリシーリソース	はい
ポリシー条件キー (サービス固有)	はい
ACL	なし
ABAC (ポリシー内のタグ)	部分的
一時認証情報	あり
プリンシパルアクセス権限	あり
サービスロール	あり
サービスリンクロール	いいえ

Amazon SWF およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要については、「IAM ユーザーガイド」のAWS 「IAM と連携する のサービス」を参照してください。

Amazon Simple Workflow Service のアイデンティティベースのポリシーの例

デフォルトでは、ユーザーとロールには Amazon SWF リソースを作成または変更するアクセス許可がありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。

これらのサンプルの JSON ポリシードキュメントを使用して IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーを作成する (コンソール)」を参照してください。

各リソースタイプの ARN の形式など、Amazon SWF で定義されるアクションとリソースタイプの詳細については、「サービス認可リファレンス」の「Amazon Simple Workflow Service のアクション、リソース、および条件キー」を参照してください。 ARNs

トピック

- ポリシーに関するベストプラクティス
- Amazon SWF コンソールの使用

- 自分の権限の表示をユーザーに許可する

ポリシーに関するベストプラクティス

ID ベースのポリシーは、ユーザーのアカウント内で誰かが Amazon SWF リソースを作成、アクセス、または削除できるどうかを決定します。これらのアクションでは、 AWS アカウントに費用が発生する場合があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可の付与を開始するには、多くの一般的なユースケースにアクセス許可を付与するAWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義することで、アクセス許可をさらに減らすことをお勧めします。詳細については、IAM ユーザーガイド の AWS マネージドポリシー または ジョブ機能のAWS マネージドポリシー を参照してください。
- 最小特権を適用する – IAM ポリシーでアクセス許可を設定する場合は、タスクの実行に必要な許可のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権アクセス許可とも呼ばれています。IAM を使用して許可を適用する方法の詳細については、IAM ユーザーガイド の IAM でのポリシーとアクセス許可 を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。たとえば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、サービスアクションが などの特定の を通じて使用されている場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます CloudFormation。詳細については、IAM ユーザーガイド の IAM JSON ポリシー要素:条件 を参照してください。
- IAM アクセスアナライザー を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM アクセスアナライザー は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、IAM ユーザーガイド の IAM Access Analyzer でポリシーを検証する を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、MFA をオンにしてセキュリティを強化します。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、IAM ユーザーガイド の MFA を使用した安全な API アクセス を参照してください。

IAM でのベストプラクティスの詳細については、IAM ユーザーガイド の IAM でのセキュリティのベストプラクティス を参照してください。

Amazon SWF コンソールの使用

Amazon Simple Workflow Service コンソールにアクセスするには、最小限の権限のセットが必要です。これらのアクセス許可により、 の Amazon SWF リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成すると、そのポリシーを持つエンティティ (ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

ユーザーとロールが引き続き Amazon SWF コンソールを使用できるようにするには、エンティティに Amazon SWF *ConsoleAccess*または *ReadOnly* AWS 管理ポリシーもアタッチします。詳細については、「IAM ユーザーガイド」の「ユーザーへのアクセス許可の追加」を参照してください。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
```

```
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

基本的な原則

Amazon SWF アクセスコントロールは、主に次の 2 種類の権限に基づいています。

- リソースアクセス許可: ユーザーがアクセスできる Amazon SWF リソース。

  ドメインに対してのみリソースアクセス許可を表すことができます。
- API のアクセス許可: ユーザーが呼び出すことができる Amazon SWF アクション。

最も簡単な方法は、完全なアカウントアクセスを許可する、任意のドメインで任意の Amazon SWF アクションを呼び出すか、アクセスを完全に拒否することです。ただし IAM では、アクセス許可への細分化されたより便利なアプローチをサポートします。例えば、次のことができます。

- 制限なしで、特定のドメイン内でのみ、任意の Amazon SWF アクションを呼び出すことができます。このようなポリシーを使用すると、開発中のワークフローアプリケーションで任意のアクションを使用できるようになりますが、「サンドボックス」ドメインのみが使用できます。
- ユーザーが任意のドメインにアクセスできるようにしますが、API の使用方法を制限します。このようなポリシーを使用すると、「監査人」アプリケーションにより任意のドメイン内の API を呼び出し、読み取りアクセスのみを許可できます。
- 特定のドメインでは、限定された一連のアクションのみを呼び出すことができます。このようなポリシーを使用すると、ワークフロースターターが指定されたドメインで `StartWorkflowExecution` アクションのみを呼び出すことができます。

Amazon SWF アクセスコントロールは、以下の原則に基づいています。

- アクセスコントロールの判断は、IAM ポリシーにのみ基づいています。すべてのポリシーの監査と操作は IAM によって行われます。
- アクセスコントロールモデルでは、「デフォルトによる拒否」ポリシーが使用されます。明示的に許可されていないすべてのアクセスは拒否されます。
- 適切な IAM ポリシーをワークフローのアクターにアタッチすることによって、Amazon SWF リソースへのアクセスを制御します。
- ドメインに対してのみリソース権限を表すことができます。
- 1 つ以上のパラメータに条件を適用することによって、一部のアクションの使用をさらに制限することができます。
- RespondDecisionTaskCompleted を使用する許可を与えた場合、そのアクションに含まれる決定のリストに対するアクセス許可を表すことができます。

  それぞれの決定には、通常の API 呼び出しと同じように、1 つ以上のパラメータがあります。ポリシーを可能な限り読みやすくするために、いくつかのパラメータに条件を適用するなど、実際の API 呼び出しのように、決定にアクセス許可を表すことができます。これらのタイプのアクセス許可は、擬似 API のアクセス許可と呼ばれます。

条件を使用して、制約を受ける、通常の API パラメータと擬似 API パラメータの概要については、API の要約 を参照してください。

Amazon SWF IAM ポリシー

IAM ポリシーには、1 つ以上の `Statement` 要素が含まれており、各要素には、そのポリシーを定義する一連の要素が含まれます。要素の完全な一覧とポリシーの作成方法に関する一般的な説明については、「The Access Policy Language」(アクセスポリシー言語) を参照してください。Amazon SWF のアクセス制御は、次の要素に基づいています。

効果

(必須) ステートメントの効果: `deny` または `allow`。

> **Note**
>
> 明示的にアクセスを許可する必要があります。IAM は、デフォルトでアクセスを拒否します。

[リソース]

(必須) ステートメントが適用される、ユーザーが操作できる AWS サービスのエンティティであるリソース。

ドメインに対してのみリソースアクセス許可を表すことができます。たとえば、ポリシーでは、アカウント内の特定のドメインのみにアクセスを許可することができます。ドメインのアクセス許可を表すには、`Resource` を「arn:aws:swf:*Region*:*AccountID*:/domain/*DomainName*」の形式のドメインの Amazon Resource Name (ARN) に設定します。*Region* は AWS リージョン、*AccountID* はダッシュのないアカウント ID、*DomainName* はドメイン名です。

Action

(必須) 次の形式 (*serviceId*:*action*) を使用して参照する、ステートメントが適用されるアクション。Amazon SWFの場合、*serviceID* を `swf` に設定します。たとえば、`swf:StartWorkflowExecution` は StartWorkflowExecution アクションを参照し、ワークフローを開始できるユーザーを制御するために使用されます。

RespondDecisionTaskCompleted を使用する権限を与えた場合は、`Action` を使用して疑似 API のアクセス許可を表すことで、含まれる決定リストへのアクセスを制御することもできます。IAM はデフォルトでアクセスを拒否するため、ディサイダーの決定が明示的に許可されない場合、受け入れられません。* 値を使用してすべての決定を許可することができます。

Condition

許可された値を制限するアクションのパラメータの 1 つ以上に制約を表現します。

Amazon SWF アクションには多くの場合、幅広い範囲があり、IAM 条件を使用することで削減できます。たとえば、PollForActivityTask アクションがアクセスできるタスクリストを制限するには、`Condition` を含め、`swf:taskList.name` キーを使用して許容リストを指定します。

以下のエンティティに対して制約を表すことができます。

- ワークフロータイプ。名前とバージョンに別のキーがあります。
- アクティビティのタイプ。名前とバージョンに別のキーがあります。
- タスクリスト。
- タグ。一部のアクションに対して複数のタグを指定できます。その場合、各タグには別のキーがあります。

> **Note**
>
> Amazon SWF の場合、値はすべて文字列です。そのため、パラメータを指定した文字列に制限する StringEquals などの文字列演算子を使用してパラメータを制約します。ただし、StringEquals などの通常の文字列比較演算子では、すべての要求にパラメータを含める必要があります。パラメータを明示的に含めず、タイプ登録時に指定されたデフォルトのタスクリストなどのデフォルト値がない場合、アクセスは拒否されます。条件をオプションとして扱うと便利です。関連するパラメータを必ずしも必要とせずにアクションを呼び出すことができます。たとえば、ディサイダーが一連の RespondDecisionTaskCompleted 決定を指定することもできますが、特定の呼び出しに対して 1 つだけ指定することもできます。その場合、StringEqualsIfExists 演算子を使用して適切なパラメータを制約します。これは、パラメータが条件を満たす場合にアクセスを許可しますが、パラメータがない場合はアクセスを拒否しません。

制約付きのパラメータと関連するキーの完全なリストについては、API の要約 を参照してください。

次のセクションでは、Amazon SWF ポリシーを構築する方法の例を示します。詳細については、「String Conditions」(文字列の条件) を参照してください。

Amazon SWF のポリシーの記述

ワークフローは、アクティビティ、ディサイダーなどの複数のアクターで構成されます。適切な IAM ポリシーをアタッチすることで、各アクターのアクセスを制御できます。

次のアクションでは、アクターにすべてのリージョンでフルアカウントアクセスが付与されます。

- アクション: `swf:*`
- リソース: `arn:aws:swf:*:123456789012:/domain/*`

ワイルドカードを使用すると、単一の値で複数のリソース、アクション、またはリージョンを表すことができます。

- Resource 値の最初のワイルドカード (*) は、リソースのアクセス許可がすべてのリージョンに適用されることを示します。

アクセス権限を単一のリージョンに制限するには、ワイルドカードを us-east-1 などの適切なリージョン文字列に置き換えます。

- `Resource` 値の 2 番目のワイルドカード (*) は、アクターが指定されたリージョン内の任意のアカウントのドメインにアクセスできるようにします。
- `Action` 値のワイルドカード (*) を使用すると、アクターは、任意の Amazon SWF アクションを呼び出すことができます。

ワイルドカードを使用する方法の詳細については、「Element Descriptions」(要素の説明) を参照してください。

ドメインのアクセス許可

部門のワークフローを特定のドメインに制限するには、特定の部門に対してのみ、アクターが任意のアクションを呼び出すことを許可するアクセス許可を付与できます。

アクターに複数のドメインへのアクセスを許可するには、各ドメインのアクセス許可をステートメントのリストとして表現します。

- アクション: `swf:*`
- リソース: `arn:aws:swf:*:123456789012:/domain/department1`
- リソース: `arn:aws:swf:*:123456789012:/domain/department2`

および department1department2ドメインで任意の Amazon SWF アクションを使用することをアクターに許可できます。ワイルドカードを使用して複数のドメインを表すこともできます。

API のアクセス許可と制約

アクターが使用できるアクションを制御するには、 `Action`要素でアクションを指定します。

次のアクションでは、アクターは を呼び出しStartWorkflowExecutionてワークフローを開始できます。他のアクションを使用することはできません。

- アクション: `swf:StartWorkflowExecution`

条件

オプションで、 `Condition`要素を使用してアクションの許容パラメータ値を制約できます。

アクターが開始できるワークフローを制限するには、次のように 1 つ以上のStartWorkflowExecutionパラメータ値を制約します。

```
"Condition" : {
   "StringEquals" : {
      "swf:workflowType.name" : "workflow1",
      "swf:workflowType.version" : "version2"
    }
}
```

前述の制約を持つアクターは のみを実行することができversion2workflow1、両方のパラメータをリクエストに含める必要があります。

次のように、StringEqualsIfExists 演算子を使用して、リクエストにパラメータを含めることなくパラメータを制約できます。

```
"Condition" : {
   "StringEqualsIfExists" : { "swf:taskList.name" : "task_list_name" }
}
```

前のポリシーを持つアクターは、ワークフロー実行を開始するときに、オプションでタスクリストを指定できます。

いくつかのアクションに対してタグのリストを制約することができます。各タグには個別のキーがあるため、 swf:tagList.member.0を使用してリスト内の最初のタグを制限し、swf:tagList.member.1 を使用してリスト内の 2 番目のタグを制限します。次に、最大 5 つまで制限します。

タグリストの制約方法には注意が必要です。たとえば、次の条件は推奨されません。

以下の条件は、オプションで some_ok_tagまたは を指定できるため、推奨されませんanother_ok_tag。ただし、Condition はタグリストの最初の要素のみを制限します。条件がswf:tagList.member.1、 などに条件を適用しないためswf:tagList.member.2、リストには任意の値を持つ追加の要素を含めることができます。

```
// Example to illustrate an insecure Condition
"Condition" : {
   "StringEqualsIfExists" : {
      "swf:tagList.member.0" : "some_ok_tag", "another_ok_tag"
   }
```

```
}
```

前の問題に対処する 1 つの方法は、タグリストの使用を禁止することです。

次のポリシーでは、リストに 1 つの要素のみ持つように要求することで、some_ok_tag または another_ok_tag のみが許可されるようにします。

```
"Condition" : {
   "StringEqualsIfExists" : {
      "swf:tagList.member.0" : "some_ok_tag", "another_ok_tag"
    },
    "Null" : { "swf:tagList.member.1" : "true" }
}
```

疑似 API のアクセス許可と制約

で使用できる決定を制限するにはRespondDecisionTaskCompleted、まずアクターに の呼び出しを許可する必要がありますRespondDecisionTaskCompleted。次に、次のように、通常の API と同じ構文を使用して、適切な擬似 API メンバーのアクセス許可を表現します。

- ステートメント 1

  リソース: arn:aws:swf:*:123456789012:/domain/*

  アクション: swf:RespondDecisionTaskCompleted
- ステートメント 2

  リソース: *

  アクション: swf:ScheduleActivityTask

  条件: "StringEquals" : { "swf:activityType.name" : "SomeActivityType" }

1 つ目は、アクターが を呼び出すことStatementを許可しますRespondDecisionTaskCompleted。2 番目のステートメントでは、アクターはScheduleActivityTask決定を使用して Amazon SWF にアクティビティタスクのスケジュールを指示できます。すべての決定を可能にするには、「swf:ScheduleActivityTask」を「swf:*」に置き換えます。

条件演算子を使用すると、通常の API と同様にパラメータを制約できます。前の例のStringEquals演算子Conditionは、 RespondDecisionTaskCompleted がアクティビティ

のSomeActivityTypeアクティビティタスクをスケジュールすることを許可し、そのタスクをスケジュールする必要があります。RespondDecisionTaskCompleted がパラメータ値を使用することを許可したいが、そうする必要がない場合は、代わりに StringEqualsIfExists 演算子を使用することができます。

AWS マネージドポリシー: SimpleWorkflowFullAccess

SimpleWorkflowFullAccess ポリシーを IAM アイデンティティにアタッチできます。

このポリシーは、Amazon SWF 設定サービスへのフルアクセスを提供します。

IAM ポリシーに関するサービスモデルの制限

IAM ポリシーを作成するときは、サービスモデルの制限を検討する必要があります。無効な Amazon SWF リクエストを表す構文的に有効な IAM ポリシーを作成することは可能です。アクセスコントロールに関して許容されるリクエストでも、無効なリクエストであるために失敗する可能性があります。

例えば、Amazon SWF サービスモデルでは、 パラメータtypeFilterと tagFilterパラメータを同じListOpenWorkflowExecutionsリクエストで使用することはできません。次の条件では、スローによってサービスが拒否する呼び出しを無効なリクエストValidationExceptionとして許可します。

```
"Condition" : {
   "StringEquals" : {
      "swf:typeFilter.name" : "workflow_name",
      "swf:typeFilter.version" : "workflow_version",
      "swf:tagFilter.tag" : "some_tag"
    }
}
```

API の要約

このセクションでは、IAM ポリシーを使用して、アクターが各 API および疑似 API を使用して Amazon SWF リソースにアクセスする方法を制御する方法について簡単に説明します。

- RegisterDomain および ListDomains 以外のすべてのアクションでは、ドメインリソースへのアクセス権限を表すことによって、アカウントのドメインの一部またはすべてへのアクセスを許可または拒否できます。

- 通常の API の任意のメンバーに対して、RespondDecisionTaskCompleted を呼び出す許可を与えた場合は、疑似 API の任意のメンバーに対してアクセス権限を許可または拒否できます。
- 条件を使用して、一部のパラメータの許容値を制限することができます。

次のセクションでは、通常 API および擬似 API の各メンバーに制約され、関連するキーを提供するパラメータをリストアップし、ドメインアクセスを制御する方法の制限を書き留めます。

通常の API

このセクションでは、通常の API メンバーをリストアップし、制約される可能性のあるパラメータと関連するキーについて簡単に説明します。また、ドメインへのアクセスを制御する方法に関する制限事項についても説明します。

CountClosedWorkflowExecutions

- tagFilter.tag - 文字列制約です。キーは swf:tagFilter.tag です。
- typeFilter.name - 文字列制約です。キーは、swf:typeFilter.name です。
- typeFilter.version - 文字列制約です。キーは、swf:typeFilter.version です。

> **Note**
>
> CountClosedWorkflowExecutions では、typeFilter と tagFilter を相互に排他的にする必要があります。

CountOpenWorkflowExecutions

- tagFilter.tag - 文字列制約です。キーは swf:tagFilter.tag です。
- typeFilter.name - 文字列制約です。キーは、swf:typeFilter.name です。
- typeFilter.version - 文字列制約です。キーは、swf:typeFilter.version です。

> **Note**
>
> CountOpenWorkflowExecutions では、typeFilter と tagFilter を相互に排他的にする必要があります。

CountPendingActivityTasks

- taskList.name - 文字列制約です。キーは、swf:taskList.name です。

CountPendingDecisionTasks

- taskList.name - 文字列制約です。キーは、swf:taskList.name です。

DeleteActivityType

- activityType.name - 文字列制約です。キーは、swf:activityType.name です。
- activityType.version - 文字列制約です。キーは、swf:activityType.version です。

DeprecateActivityType

- activityType.name - 文字列制約です。キーは、swf:activityType.name です。
- activityType.version - 文字列制約です。キーは、swf:activityType.version です。

DeprecateDomain

- このアクションのパラメータを制約することはできません。

DeleteWorkflowType

- workflowType.name - 文字列制約です。キーは、swf:workflowType.name です。
- workflowType.version - 文字列制約です。キーは、swf:workflowType.version です。

DeprecateWorkflowType

- workflowType.name - 文字列制約です。キーは、swf:workflowType.name です。
- workflowType.version - 文字列制約です。キーは、swf:workflowType.version です。

DescribeActivityType

- activityType.name - 文字列制約です。キーは、swf:activityType.name です。
- activityType.version - 文字列制約です。キーは、swf:activityType.version です。

DescribeDomain

- このアクションのパラメータを制約することはできません。

DescribeWorkflowExecution

- このアクションのパラメータを制約することはできません。

DescribeWorkflowType

- workflowType.name - 文字列制約です。キーは、swf:workflowType.name です。
- workflowType.version - 文字列制約です。キーは、swf:workflowType.version です。

GetWorkflowExecutionHistory

- このアクションのパラメータを制約することはできません。

ListActivityTypes

- このアクションのパラメータを制約することはできません。

ListClosedWorkflowExecutions

- tagFilter.tag - 文字列制約です。キーは swf:tagFilter.tag です。
- typeFilter.name - 文字列制約です。キーは、swf:typeFilter.name です。
- typeFilter.version - 文字列制約です。キーは、swf:typeFilter.version です。

> **Note**
>
> ListClosedWorkflowExecutions では、typeFilter と tagFilter を相互に排他的にする必要があります。

ListDomains

- このアクションのパラメータを制約することはできません。

ListOpenWorkflowExecutions

- tagFilter.tag - 文字列制約です。キーは swf:tagFilter.tag です。
- typeFilter.name - 文字列制約です。キーは、swf:typeFilter.name です。
- typeFilter.version - 文字列制約です。キーは、swf:typeFilter.version です。

> **Note**
>
> ListOpenWorkflowExecutions では、typeFilter と tagFilter を相互に排他的にする必要があります。

ListWorkflowTypes

- このアクションのパラメータを制約することはできません。

PollForActivityTask

- taskList.name - 文字列制約です。キーは、swf:taskList.name です。

PollForDecisionTask

- taskList.name - 文字列制約です。キーは、swf:taskList.name です。

RecordActivityTaskHeartbeat

- このアクションのパラメータを制約することはできません。

RegisterActivityType

- defaultTaskList.name - 文字列制約です。キーは、swf:defaultTaskList.name です。
- name - 文字列制約です。キーは、swf:name です。
- version - 文字列制約です。キーは、swf:version です。

RegisterDomain

- name - 登録されたドメインの名前は、このアクションのリソースとして利用できます。

RegisterWorkflowType

- defaultTaskList.name - 文字列制約です。キーは、swf:defaultTaskList.name です。
- name - 文字列制約です。キーは、swf:name です。
- version - 文字列制約です。キーは、swf:version です。

RequestCancelWorkflowExecution

- このアクションのパラメータを制約することはできません。

RespondActivityTaskCanceled

- このアクションのパラメータを制約することはできません。

RespondActivityTaskCompleted

- このアクションのパラメータを制約することはできません。

RespondActivityTaskFailed

- このアクションのパラメータを制約することはできません。

RespondDecisionTaskCompleted

- decisions.member.N - 擬似 API のアクセス権限を使用して間接的に制限されます。詳細については、「擬似 API」を参照してください。

SignalWorkflowExecution

- このアクションのパラメータを制約することはできません。

StartWorkflowExecution

- tagList.member.0 - 文字列制約です。キーは swf:tagList.member.0 です。

- tagList.member.1 - 文字列制約です。キーは swf:tagList.member.1 です。
- tagList.member.2 - 文字列制約です。キーは swf:tagList.member.2 です。
- tagList.member.3 - 文字列制約です。キーは swf:tagList.member.3 です。
- tagList.member.4 - 文字列制約です。キーは swf:tagList.member.4 です。
- taskList.name - 文字列制約です。キーは、swf:taskList.name です。
- workflowType.name - 文字列制約です。キーは、swf:workflowType.name です。
- workflowType.version - 文字列制約です。キーは、swf:workflowType.version です。

> **Note**
>
> 5 つ以上のタグを制約することはできません。

TerminateWorkflowExecution

- このアクションのパラメータを制約することはできません。

擬似 API

このセクションでは、RespondDecisionTaskCompleted に含まれる決定を表す擬似 API のメンバーをリストします。RespondDecisionTaskCompleted の使用を許可した場合、ポリシーは通常の API と同じ方法でこの API のメンバーのアクセス許可を表すことができます。1 つ以上のパラメータに条件を設定することによって、擬似 API の一部のメンバーをさらに制限することができます。このセクションでは、疑似 API メンバーをリストアップし、制約される可能性のあるパラメータと関連するキーについて簡単に説明します。

> **Note**
>
> aws:SourceIP、aws:UserAgent、aws:SecureTransport キーは擬似 API には使用できません。意図したセキュリティポリシーでこれらのキーで疑似 API へのアクセスを制御する必要がある場合は、RespondDecisionTaskCompleted アクションでそれらのキーを使用できます。

CancelTimer

- このアクションのパラメータを制約することはできません。

CancelWorkflowExecution

- このアクションのパラメータを制約することはできません。

CompleteWorkflowExecution

- このアクションのパラメータを制約することはできません。

ContinueAsNewWorkflowExecution

- tagList.member.0 - 文字列制約です。キーは swf:tagList.member.0 です。
- tagList.member.1 - 文字列制約です。キーは swf:tagList.member.1 です。
- tagList.member.2 - 文字列制約です。キーは swf:tagList.member.2 です。
- tagList.member.3 - 文字列制約です。キーは swf:tagList.member.3 です。
- tagList.member.4 - 文字列制約です。キーは swf:tagList.member.4 です。
- taskList.name - 文字列制約です。キーは、swf:taskList.name です。
- workflowTypeVersion - 文字列制約です。キーは、swf:workflowTypeVersion です。

> **Note**
>
> 5 つ以上のタグを制約することはできません。

FailWorkflowExecution

- このアクションのパラメータを制約することはできません。

RecordMarker

- このアクションのパラメータを制約することはできません。

RequestCancelActivityTask

- このアクションのパラメータを制約することはできません。

RequestCancelExternalWorkflowExecution

- このアクションのパラメータを制約することはできません。

ScheduleActivityTask

- activityType.name - 文字列制約です。キーは、swf:activityType.name です。
- activityType.version - 文字列制約です。キーは、swf:activityType.version です。
- taskList.name - 文字列制約です。キーは、swf:taskList.name です。

SignalExternalWorkflowExecution

- このアクションのパラメータを制約することはできません。

StartChildWorkflowExecution

- tagList.member.0 - 文字列制約です。キーは swf:tagList.member.0 です。
- tagList.member.1 - 文字列制約です。キーは swf:tagList.member.1 です。
- tagList.member.2 - 文字列制約です。キーは swf:tagList.member.2 です。
- tagList.member.3 - 文字列制約です。キーは swf:tagList.member.3 です。
- tagList.member.4 - 文字列制約です。キーは swf:tagList.member.4 です。
- taskList.name - 文字列制約です。キーは、swf:taskList.name です。
- workflowType.name - 文字列制約です。キーは、swf:workflowType.name です。
- workflowType.version - 文字列制約です。キーは、swf:workflowType.version です。

> **Note**
>
> 5 つ以上のタグを制約することはできません。

StartTimer

- このアクションのパラメータを制約することはできません。

タグベースのポリシー

Amazon SWF はタグベースのポリシーをサポートしています。たとえば、キー と値 を持つタグを含む Amazon SWF ドメイン`environment`を`production`次の条件で制限できます。

```
"Condition": {
    "StringEquals": {"aws:ResourceTag/environment": "production"}
}
```

タグ付けの詳細については、以下を参照してください。

- Amazon SWF のタグ
- IAM タグを使用したアクセスの制御

Amazon SWF 用の Amazon VPC エンドポイント

> **Note**
>
> AWS PrivateLink サポートは現在、 AWS トップシークレット - 東部、 AWS シークレットリージョン、中国リージョンでのみ利用できます。

Amazon Virtual Private Cloud (Amazon VPC) を使用して AWS リソースをホストする場合は、Amazon VPC と Amazon Simple Workflow Service ワークフロー間の接続を確立できます。パブリックインターネットと交差せずに、この Amazon SWF ワークフローとの接続を使用できます。

Amazon VPC では、カスタム仮想ネットワークで AWS リソースを起動できます。VPC を使用して、IP アドレス範囲、サブネット、ルートテーブル、ネットワークゲートウェイなどのネットワーク設定を制御できます。Amazon VPC の詳細については、「Amazon VPC ユーザーガイド」を参照してください。

Amazon VPCを Amazon SWF に接続するには、まずインターフェイス VPC エンドポイントを定義する必要があります。これにより、VPC を他の AWS のサービスに接続できるようになります。このエンドポイントを使用すると、インターネットゲートウェイやネットワークアドレス変換 (NAT) インスタンス、または VPN 接続などを必要とせずに、信頼性の高いスケーラブルな方法で接続できるようになります。詳細については、「Amazon VPC ユーザーガイド」の「インターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。

エンドポイントを作成する

VPC で Amazon SWF エンドポイントを作成するには AWS マネジメントコンソール、、 AWS Command Line Interface (AWS CLI) 、 AWS SDK、Amazon SWF API、または を使用します CloudFormation。

Amazon VPC コンソールまたは AWS CLIを使用して、エンドポイントを作成および設定する方法については、「Amazon VPC ユーザーガイド」の「インターフェイスエンドポイントの作成」を参照してください。

> **Note**
>
> エンドポイントを作成するとき、VPC の接続先のサービスとして Amazon SWF を指定します。Amazon VPC コンソール上のサービス名は AWS リージョンによって異なります。たとえば、 AWS トップシークレット - 東部リージョンでは、Amazon SWF のサービス名は com.amazonaws.us-iso-east-1.swf です。

を使用してエンドポイントを作成および設定する方法については CloudFormation、「 CloudFormation ユーザーガイド」のAWS::EC2::VPCEndpoint リソース」を参照してください。

Amazon VPC エンドポイントポリシー

Amazon SWF への接続アクセスを制御するには、Amazon VPC エンドポイントの作成中に AWS Identity and Access Management (IAM) エンドポイントポリシーをアタッチします。複数のエンドポイントポリシーをアタッチすることで、複雑な IAM ルールを作成できます。詳細については、以下を参照してください。

- Amazon SWF の Amazon Virtual Private Cloud エンドポイントポリシー
- VPC エンドポイントによるサービスのアクセス制御

Amazon SWF の Amazon Virtual Private Cloud エンドポイントポリシー

以下を指定して、Amazon SWF の Amazon VPC エンドポイントポリシーを作成できます。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。
- このアクションを実行できるリソース。

次の例では、ポリシーに特定の IAM ロールを追加します。

```
"Principal": {
   "AWS": "arn:aws:iam::123456789012:role/MyRole"
}
```

- エンドポイントポリシーの作成方法の詳細については、「VPC エンドポイントによるサービスへのアクセスの制御」を参照してください。
- IAM を使用して AWS および Amazon SWF リソースへのアクセスを制御する方法については、「」を参照してくださいAmazon Simple Workflow Service での Identity and Access Management。

Amazon Simple Workflow Service アイデンティティとアクセスのトラブルシューティング

Amazon SWF と IAM の使用に伴って発生する可能性がある一般的な問題の診断や修復には、次の情報を利用してください。

トピック

- Amazon SWF でアクションを実行する認可がありません
- iam:PassRole を実行する権限がありません
- 自分の 以外のユーザーに Amazon SWF リソース AWS アカウント へのアクセスを許可したい

Amazon SWF でアクションを実行する認可がありません

アクションを実行する権限がないというエラーが表示された場合は、そのアクションを実行できるようにポリシーを更新する必要があります。

以下のエラー例は、mateojackson ユーザーがコンソールを使用して架空の *my-example-widget* リソースに関する詳細情報を表示しようとしているが、架空の swf:*GetWidget* 許可がないという場合に発生します。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 swf:GetWidget on resource: my-example-widget
```

この場合、Mateo のポリシーでは、*my-example-widget* アクションを使用して swf:*GetWidget* リソースへのアクセスを許可するように更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がありません

`iam:PassRole` アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して Amazon SWF にロールを渡せるようにする必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、既存のロールをそのサービスに渡すことができます。そのためには、サービスにロールを渡すアクセス許可が必要です。

次の例のエラーは、`marymajor` という IAM ユーザーがコンソールを使用して Amazon SWF でアクションを実行しようとする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与されたアクセス許可が必要です。Mary には、ロールをサービスに渡すアクセス許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに `iam:PassRole` アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の 以外のユーザーに Amazon SWF リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- Amazon SWF がこれらの機能をサポートしているかどうかを確認するには、「Amazon Simple Workflow Service で IAM を使用する方法」を参照してください。

- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、「IAM ユーザーガイド」の「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、IAM ユーザーガイド の 外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可 を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

ログ記録とモニタリング

このセクションでは、Amazon SWFのログ作成およびモニタリングについて説明します。

トピック

- CloudWatch の Amazon SWF メトリクス
- を使用した CloudWatch の Amazon SWF メトリクスの表示 AWS マネジメントコンソール
- を使用した API コールの記録 AWS CloudTrail
- EventBridge for Amazon SWF 実行状況の変更
- Amazon Simple Workflow Service AWS User Notifications での の使用

CloudWatch の Amazon SWF メトリクス

Amazon SWF では、ワークフローおよびアクティビティを追跡し、選択したしきい値にアラームを設定するために使用できる CloudWatch のメトリクスを提供するようになりました。を使用してメトリクスを表示できます AWS マネジメントコンソール。詳細については、「を使用した CloudWatch の Amazon SWF メトリクスの表示 AWS マネジメントコンソール」を参照してください。

トピック

- Amazon SWF メトリクスの単位の報告
- API と決定イベントのメトリクス
- Amazon SWF のメトリクス

- Amazon SWF 非 ASCII リソース名と CloudWatch ディメンション

Amazon SWF メトリクスの単位の報告

時間間隔を報告するメトリクス

CloudWatch の Amazon SWF メトリクスの一部は時間間隔であり、常にミリ秒単位で測定されます。CloudWatch の単位は `Time` として報告されます。これらのメトリクスは通常、ワークフローおよびアクティビティのタイムアウトを設定できるワークフロー内の実行段階に対応しており、それぞれに似た名前が付いています。

たとえば、`DecisionTaskStartToCloseTime` メトリクスは、ディシジョンタスクが実行開始されてから完了するまでの時間を測定します。この同じ期間に対し、`DecisionTaskStartToCloseTimeout` 値を設定できます。

これらの各ワークフローステージの図と、ワークフローおよびアクティビティのライフサイクルで各ワークフローステージがいつ発生するかについては、「Amazon SWF タイムアウトの種類」を参照してください。

カウントを報告するメトリクス

CloudWatch の Amazon SWF メトリクスの一部は、結果を カウント として報告します。例えば、`WorkflowsCanceled` は、結果を 1 または 0 としてレコードしますが、これはワークフローがキャンセルされたかどうかを表します。値が 0 の場合、メトリクスが報告されなかったのではなく、メトリクスに記述された条件が発生しなかったことを示しているだけです。

CloudWatch で `Count` を報告する CloudWatch の Amazon SWF メトリックスの一部は、1 秒あたりのカウント数です。例えば、CloudWatch で `Count` として報告される `ProvisionedRefillRate` は、`Count` の 1 秒あたりのリクエスト数を表します。

カウントを返すメトリクスでは、最小値および最大値は常に 0 または 1 ですが、平均は 0 ~ 1 の範囲の値になります。

API と決定イベントのメトリクス

CloudWatch の API および決定イベントをモニタリングして、使用量と容量のインサイトを得ることができます。「Amazon SWF の基本的なワークフローの概念」セクションの「deciders」(ディサイダー)、および「Amazon Simple Workflow Service API Reference」(Amazon Simple Workflow Service API リファレンス) の「Decision」(決定) のトピックを参照してください。

また、Amazon SWF のスロットリング制限に近づいているときに、これらの制限をモニタリングしてアラームを受けることができます。これらの制約の説明とデフォルト設定については、「Amazon SWF スロットリングのクォータ」を確認してください。これらの制限は、不適切なワークフローがシステムリソースを過度に消費することを防ぐためのものです。この制限の上限緩和を申請する方法については、「???」を参照してください。

ベストプラクティスとして、API または決定イベントの容量の 60% 前後に CloudWatch アラームを設定します。これにより、Amazon SWF のスロットリングを有効にする前に、ワークフローを調整するか、サービスの上限緩和を申請できます。呼び出しのバースト性により、サービスの上限に近づいているときに通知する複数のアラームを設定できます。

- トラフィックに非常に大きなスパイクがある場合は、`ProvisionedBucketSize` の制限の 60% にアラームを設定します。
- 呼び出しのレートが比較的安定している場合は、関連 API および決定イベントの `ProvisionedRefillRate` 制限の 60% にアラームを設定します。

Amazon SWF のメトリクス

Amazon SWF では、次のメトリックを使用できます。

メトリクス	説明
`DecisionTaskScheduleToStartTime`	決定タスクを予定した時刻からワーカーがそれを取得し、開始した時刻までの時間間隔 (ミリ秒)。 CloudWatch 単位: `Time` ディメンション: `Domain`, `WorkflowTypeName`, `WorkflowTypeVersion` 有効な統計: `Average`, `Minimum`, `Maximum`
`DecisionTaskStartToCloseTime`	決定タスクの開始時刻からクローズ時刻までの時間間隔 (ミリ秒)。 CloudWatch 単位: `Time` ディメンション: `Domain`, `WorkflowTypeName`, `WorkflowTypeVersion`

メトリクス	説明
	有効な統計: Average, Minimum, Maximum
DecisionTasksCompleted	完了した決定タスクの数。 CloudWatch 単位: Count ディメンション: Domain, WorkflowTypeName, WorkflowTypeVersion 有効な統計: Sum
PendingTasks	特定のタスクリストでの保留中のタスクの数 (1 分間隔)。 CloudWatch 単位: Count ディメンション: Domain, TaskListName 有効な統計: Sum
StartedDecisionTasksTimedOutOnClose	開始したが、クローズ時にタイムアウトした決定タスクの数。 CloudWatch 単位: Count ディメンション: Domain, WorkflowTypeName, WorkflowTypeVersion 有効な統計: Sum
WorkflowStartToCloseTime	ワークフローの開始時刻からクローズ時刻までの時間間隔 (ミリ秒)。 CloudWatch 単位: Time ディメンション: Domain, WorkflowTypeName, WorkflowTypeVersion 有効な統計: Average, Minimum, Maximum

メトリクス	説明
WorkflowsCanceled	取り消されたワークフローの数。 CloudWatch 単位: Count ディメンション: Domain, WorkflowTypeName, WorkflowTypeVersion 有効な統計: Sum
WorkflowsCompleted	完了したワークフローの数。 CloudWatch 単位: Count ディメンション: Domain, WorkflowTypeName, WorkflowTypeVersion 有効な統計: Sum
WorkflowsContinuedAsNew	新規として継続したワークフローの数。 CloudWatch 単位: Count ディメンション: Domain, WorkflowTypeName, WorkflowTypeVersion 有効な統計: Sum
WorkflowsFailed	失敗したワークフローの数。 CloudWatch 単位: Count ディメンション: Domain, WorkflowTypeName, WorkflowTypeVersion 有効な統計: Sum

メトリクス	説明
WorkflowsTerminated	終了されたワークフローの数。 CloudWatch 単位: Count ディメンション: Cause, Domain, WorkflowTypeName, WorkflowTypeVersion 有効な統計: Sum
WorkflowsTimedOut	何らかの理由でタイムアウトしたワークフローの数。 CloudWatch 単位: Count ディメンション: Domain, WorkflowTypeName, WorkflowTypeVersion 有効な統計: Sum
ActivityTaskScheduleToCloseTime	アクティビティのスケジュール時刻からクローズ時刻までの時間間隔 (ミリ秒)。 CloudWatch 単位: Time ディメンション: Domain, ActivityTypeName, ActivityTypeVersion 有効な統計: Average, Minimum, Maximum
ActivityTaskScheduleToStartTime	アクティビティタスクのスケジュール時刻から開始時刻までの時間間隔 (ミリ秒)。 CloudWatch 単位: Time ディメンション: Domain, ActivityTypeName, ActivityTypeVersion 有効な統計: Average, Minimum, Maximum

メトリクス	説明
ActivityTaskStartToCloseTime	アクティビティタスクの開始時刻からクローズ時刻までの時間間隔 (ミリ秒)。 CloudWatch 単位: Time ディメンション: Domain, ActivityTypeName, ActivityTypeVersion 有効な統計: Average, Minimum, Maximum
ActivityTasksCanceled	取り消されたアクティビティタスクの数。 CloudWatch 単位: Count ディメンション: Domain, ActivityTypeName, ActivityTypeVersion 有効な統計: Sum
ActivityTasksCompleted	完了したアクティビティタスクの数。 CloudWatch 単位: Count ディメンション: Domain, ActivityTypeName, ActivityTypeVersion 有効な統計: Sum
ActivityTasksFailed	失敗したアクティビティタスクの数。 CloudWatch 単位: Count ディメンション: Domain, ActivityTypeName, ActivityTypeVersion 有効な統計: Sum

メトリクス	説明
ScheduledActivityTasksTimedOutOnClose	スケジュールされたが、クローズ時にタイムアウトしたアクティビティタスクの数。 CloudWatch 単位: Count ディメンション: Domain, ActivityTypeName, ActivityTypeVersion 有効な統計: Sum
ScheduledActivityTasksTimedOutOnStart	スケジュールされたが、開始時にタイムアウトしたアクティビティタスクの数。 CloudWatch 単位: Count ディメンション: Domain, ActivityTypeName, ActivityTypeVersion 有効な統計: Sum
StartedActivityTasksTimedOutOnClose	開始されたが、クローズ時にタイムアウトしたアクティビティタスクの数。 CloudWatch 単位: Count ディメンション: Domain, ActivityTypeName, ActivityTypeVersion 有効な統計: Sum
StartedActivityTasksTimedOutOnHeartbeat	開始されたが、ハートビートタイムアウトのためタイムアウトしたアクティビティタスクの数。 CloudWatch 単位: Count ディメンション: Domain, ActivityTypeName, ActivityTypeVersion 有効な統計: Sum

メトリクス	説明
ThrottledEvents	スロットルされたリクエストの数。 CloudWatch 単位: Count ディメンション: APIName, DecisionName, ThrottlingScope 有効な統計: Sum
ProvisionedBucketSize	1 秒あたりに利用できるリクエストの数。 ディメンション: APIName, DecisionName 有効な統計: Minimum
ConsumedCapacity	1 秒あたりのリクエストの数。 CloudWatch 単位: Count ディメンション: APIName, DecisionName 有効な統計: Sum
ConsumedLimit	消費された一般限度額。 ディメンション: GeneralLimitType
ProvisionedRefillRate	バケットで許可される 1 秒あたりのリクエストの数。 ディメンション: APIName, DecisionName 有効な統計: Minimum
ProvisionedLimit	アカウントにプロビジョニングされた一般限度額。 ディメンション: GeneralLimitType

ディメンション	説明
Domain	ワークフローまたはアクティビティが実行中の Amazon SWF ドメインにデータをフィルタリングします。
ActivityTypeName	アクティビティタイプの名前にデータをフィルタリングします。
ActivityTypeVersion	アクティビティタイプのバージョンにデータをフィルタリングします。
WorkflowTypeName	このワークフロー実行のワークフロータイプの名前にデータをフィルタリングします。
WorkflowTypeVersion	このワークフロー実行のワークフロータイプのバージョンにデータをフィルタリングします。
APIName	指定された API 名の API にデータをフィルタリングします。
DecisionName	指定された決定名にデータをフィルタリングします。
TaskListName	指定されたタスクリスト名にデータをフィルタリングします。
TaskListClassification	タスクリストの分類にデータをフィルタリングします。値は、決定タスクリストの場合は「D」、アクティビティタスクリストの場合は「A」です。
ThrottlingScope	指定されたスロットリングスコープにデータをフィルタリングします。値は、アカウントレベルのクォータを超える場合は「アカウント」、ワークフローレベルのクォータを超える場合は「ワークフロー」です。

Amazon SWF 非 ASCII リソース名と CloudWatch ディメンション

Amazon SWF では、TaskList や DomainName などのリソース名に非 ASCII 文字を使用できます。ただし、CloudWatch メトリクスのディメンション値に含めることができるのは、印刷可能な ASCII 文字だけです。Amazon SWF が CloudWatch 要件 と互換性のあるディメンション値を使用するようにするために、これらの要件を満たしていない Amazon SWF リソース名は変換され、次のようにチェックサムが追加されます

- 非 ASCII 文字はすべて ? に置き換えられます。
- 入力文字列または変換された文字列は、必要に応じて切り捨てられます。これにより、チェックサムが追加されたときに、新しい文字列の長さが CloudWatch の最大値を超えないようにします。
- ASCII 以外の文字は に変換されるため?、変換前に異なっていた一部の CloudWatch メトリクスディメンション値は、変換後に同じように見える場合があります。それらを区別しやすくするために、リソース名には、アンダースコア (_) の後に、元のリソース名の SHA256 チェックサムの最初の 16 文字が追加されます。

変換の例:

- `test àpple` は `test ?pple_82cc5b8e3a771d12` に変換されます
- `àòà` は `???_2fec5edbb2c05c22` に変換されます。
- TaskList の名前 `àpplé` と `âpplè` は両方とも `?ppl?` に変換され、同一になります。チェックサムを追加すると、それぞれ `?ppl?_f39a36df9d85a69d`、`?ppl?_da3efb4f11dd0f7f` という異なる値が返されます。

> **Tip**
>
> 独自の SHA256 チェックサムを生成できます。例えば、`shasum` コマンドラインツールを使用するには、次のようにします。
> echo -n "<the original resource name>" | shasum -a 256 | cut -c1-16

を使用した CloudWatch の Amazon SWF メトリクスの表示 AWS マネジメントコンソール

Amazon CloudWatch は、Amazon SWF ワークフローとアクティビティの表示可能な数多くのメトリクスを提供します。Amazon SWF ワークフロー実行のメトリクスの表示とアラームの設定は、AWS マネジメントコンソール を使用して行うことができます。続行するには、コンソールにログインしている必要があります。

利用できる各メトリクスの説明については、「CloudWatch の Amazon SWF メトリクス」を参照してください。

トピック

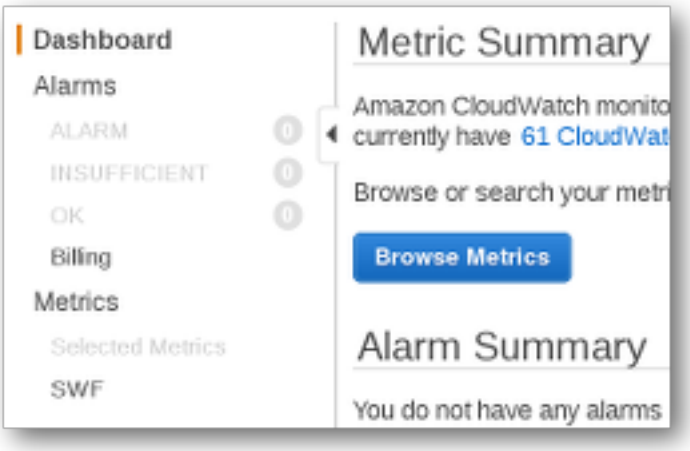
- メトリクスの表示

- アラームの設定

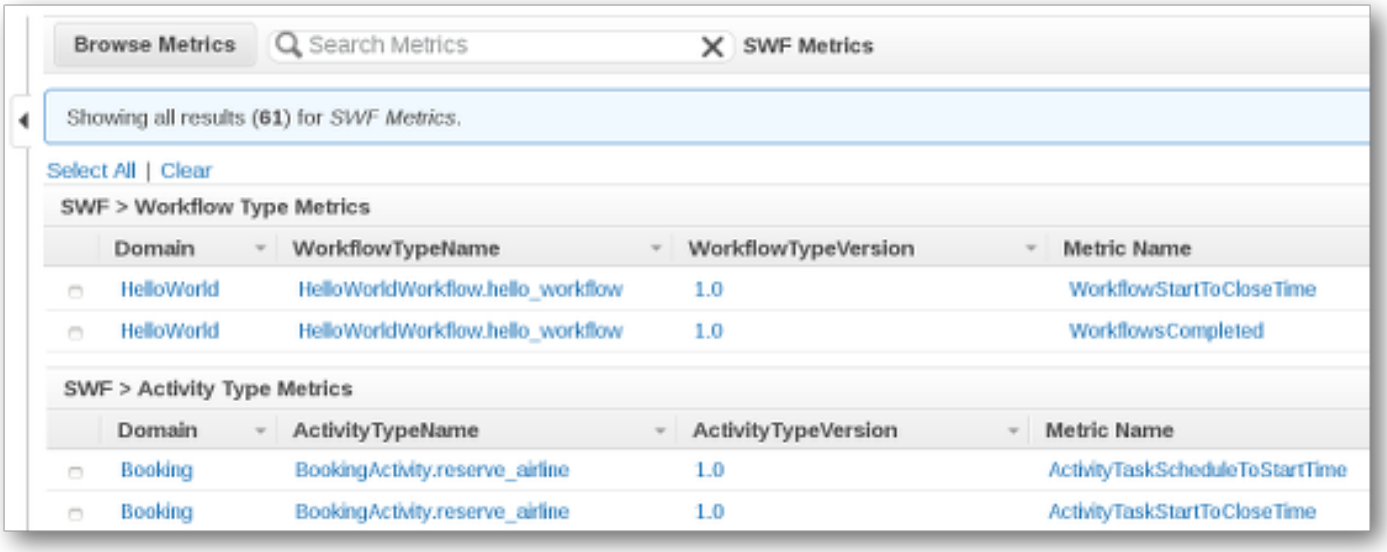
メトリクスの表示

Amazon SWF のメトリクスを表示するには

1. にサインイン AWS マネジメントコンソール し、https://console.aws.amazon.com/cloudwatch/ で CloudWatch コンソールを開きます。
2. ナビゲーションペインの [Metrics] (メトリクス) で、[SWF] を選択します。



ワークフロー実行を最近行っている場合、[Workflow Type Metrics] (ワークフロータイプのメトリクス) および [Activity Type Metrics] (アクティビティタイプのメトリクス) の 2 つのメトリクスのリストが表示されます。



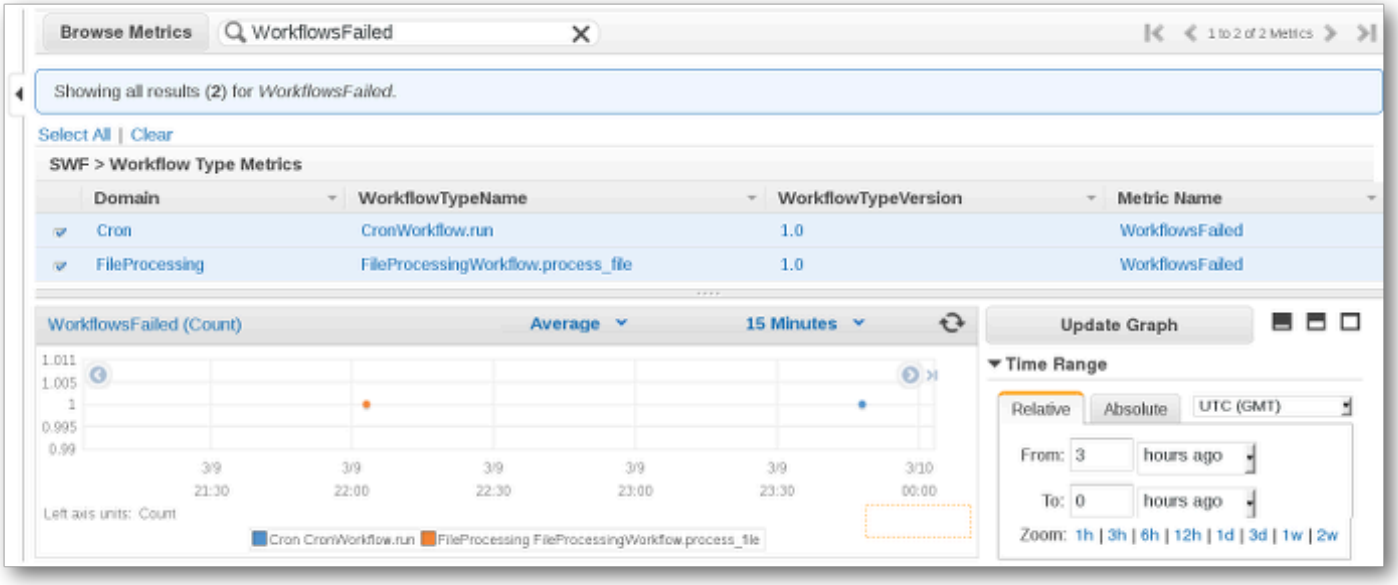
> **Note**
>
> 最初は、[Workflow Type Metrics] (ワークフロータイプのメトリクス) のみが表示される場合があります。[Activity Type Metrics] (アクティビティタイプのメトリクス) は同じビューで表示されますが、表示するには下へのスクロールが必要になる場合があります。

一度に最大 50 の最新のメトリクスが、ワークフローメトリクスおよびアクティビティメトリクスに分けられて表示されます。

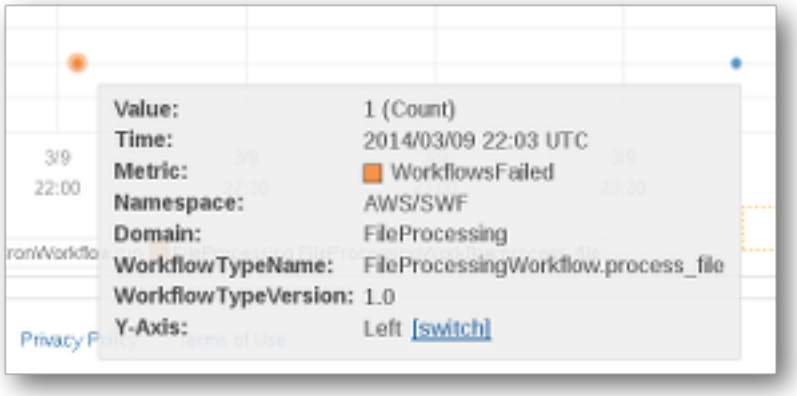
リストの上記の各列のインタラクティブな見出しを使用し、提供されたディメンションのいずれかを使ってメトリクスをソートできます。ワークフローの場合、ディメンションは、Domain (ドメイン)、WorkflowTypeName、WorkflowTypeVersion、および Metric Name (メトリクス名) です。アクティビティの場合、ディメンションは、Domain (ドメイン)、ActivityTypeName、ActivityTypeVersion、および Metric Name (メトリクス名) です。

さまざまな種類のメトリクスについては、「CloudWatch の Amazon SWF メトリクス」で説明しています。

リストのメトリクス列の横にあるボックスを選択して、メトリクスのグラフを表示でき、グラフ表示の右側にある [Time Range] (時間の範囲) コントロールを使用してグラフのパラメータを変更できます。



グラフの任意のポイントに関する詳細については、グラフのポイントにカーソルを置きます。ポイントのディメンションの詳細が表示されます。



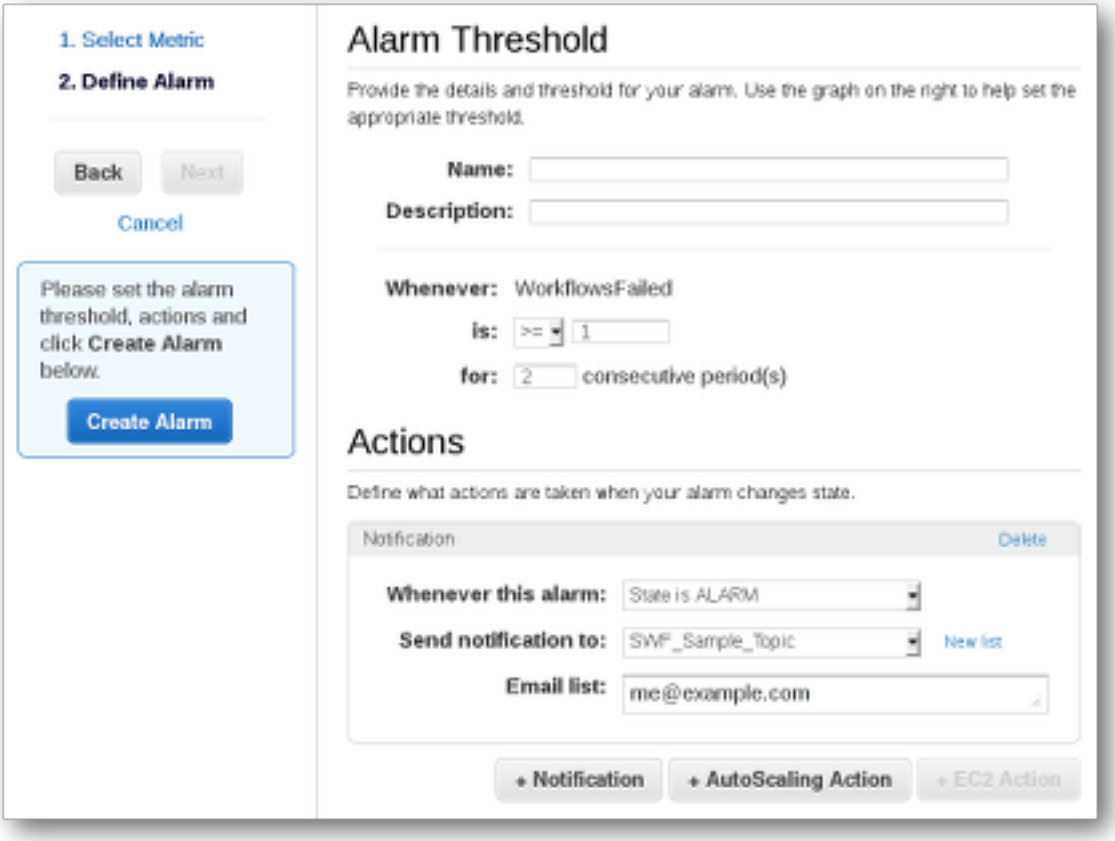
CloudWatch メトリクスの使用方法の詳細については、「Amazon CloudWatch ユーザーガイド」の「Viewing, Graphing, and Publishing Metrics」(メトリクスの表示/グラフ化/発行) を参照してください。

アラームの設定

CloudWatch アラームを使用して、アラームのしきい値に達したときに通知するなどのアクションを実行できます。たとえば、`WorkflowsFailed` メトリクスが特定のしきい値を超えたときに SNS トピックに通知を送信したり、E メールを送信したりするようにアラームを設定できます。

メトリクスに対してアラームを設定するには

1. ボックスを選択して 1 つのメトリクスを選択します。
2. グラフの右にある [Tools] (ツール) コントロールで、[Create Alarm] (アラームの作成) を選択します。
3. [Define Alarm] (アラームの定義) 画面で、アラームのしきい値、期間パラメータ、および実行するアクションを入力します。



CloudWatch アラームの設定と使用の詳細については、「Amazon CloudWatch User Guide」(Amazon Cloud Watch ユーザーガイド) の「Creating Amazon CloudWatch Alarms」(Amazon CloudWatch アラームの作成) を参照してください。

を使用した API コールの記録 AWS CloudTrail

Amazon Simple Workflow Service は、ユーザーAWS CloudTrail、ロール、または によって実行されたアクションを記録するサービスである と統合されています AWS のサービス。CloudTrail は、Amazon SWF へのすべての API コールをイベントとしてキャプチャします。キャプチャされた呼び出しには、Amazon SWF コンソールからの呼び出しと、Amazon SWF API オペレーションへの

コード呼び出しが含まれます。CloudTrail で収集された情報を使用して、Amazon SWF に対するリクエスト、リクエスト元の IP アドレス、リクエスト日時などの詳細を確認できます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- ルートユーザーまたはユーザー認証情報のどちらを使用してリクエストが送信されたか。
- リクエストが IAM Identity Center ユーザーに代わって行われたかどうか。
- リクエストがロールまたはフェデレーションユーザーのテンポラリなセキュリティ認証情報を使用して行われたかどうか。
- リクエストが、別の AWS のサービスによって送信されたかどうか。

CloudTrail は、アカウントを作成する AWS アカウント と でアクティブになり、CloudTrail イベント履歴に自動的にアクセスできます。CloudTrail の [イベント履歴] では、 AWS リージョンで過去 90 日間に記録された 管理イベントの表示、検索、およびダウンロードが可能で、変更不可能な記録を確認できます。詳細については、「AWS CloudTrail ユーザーガイド」の「CloudTrail イベント履歴の使用」を参照してください。[イベント履歴] の閲覧には CloudTrail の料金はかかりません。

AWS アカウント 過去 90 日間のイベントの継続的な記録については、証跡または CloudTrail Lake イベントデータストアを作成します。

CloudTrail 証跡

証跡により、CloudTrail はログファイルを Amazon S3 バケットに配信できます。を使用して作成されたすべての証跡 AWS マネジメントコンソール はマルチリージョンです。 AWS CLIを使用する際は、単一リージョンまたは複数リージョンの証跡を作成できます。アカウント AWS リージョン 内のすべての でアクティビティをキャプチャするため、マルチリージョン証跡を作成することをお勧めします。単一リージョンの証跡を作成する場合、証跡の AWS リージョンに記録されたイベントのみを表示できます。証跡の詳細については、「AWS CloudTrail ユーザーガイド」の「AWS アカウントの証跡の作成」および「組織の証跡の作成」を参照してください。

証跡を作成すると、進行中の管理イベントのコピーを 1 つ無料で CloudTrail から Amazon S3 バケットに配信できますが、Amazon S3 ストレージには料金がかかります。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。Amazon S3 の料金に関する詳細については、「Amazon S3 の料金」を参照してください。

CloudTrail Lake イベントデータストア

[CloudTrail Lake] を使用すると、イベントに対して SQL ベースのクエリを実行できます。CloudTrail Lake は、行ベースの JSON 形式の既存のイベントを Apache ORC 形式に変換します。ORC は、データを高速に取得するために最適化された単票ストレージ形式です。イベントは、イベントデータストアに集約されます。イベントデータストアは、高度なイベントセレクタを適用することによって選択する条件に基づいた、イベントのイミュータブルなコレクションです。どのイベントが存続し、クエリに使用できるかは、イベントデータストアに適用するセレクタが制御します。CloudTrail Lake の詳細については、 AWS CloudTrail ユーザーガイドのAWS CloudTrail 「Lake の使用」を参照してください。

CloudTrail Lake のイベントデータストアとクエリにはコストがかかります。イベントデータストアを作成する際に、イベントデータストアに使用する料金オプションを選択します。料金オプションによって、イベントの取り込みと保存にかかる料金、および、そのイベントデータストアのデフォルトと最長の保持期間が決まります。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。

CloudTrail のデータイベント

データイベントでは、リソース上またはリソース内で実行されるリソースオペレーション (Amazon S3 オブジェクトの読み取りまたは書き込みなど) についての情報が得られます。これらのイベントは、データプレーンオペレーションとも呼ばれます。データイベントは、多くの場合、高ボリュームのアクティビティです。デフォルトでは、CloudTrail はデータイベントをログ記録しません。CloudTrail [イベント履歴] にはデータイベントは記録されません。

追加の変更がイベントデータに適用されます。CloudTrail の料金の詳細については、「AWS CloudTrail の料金」を参照してください。

CloudTrail コンソール、または CloudTrail CloudTrail API オペレーションを使用して AWS CLI、Amazon SWF リソースタイプのデータイベントを記録できます。データイベントをログに記録する方法の詳細については、「AWS CloudTrail ユーザーガイド」の「AWS マネジメントコンソールを使用したデータイベントのログ記録」および「AWS Command Line Interfaceを使用したデータイベントのログ記録」を参照してください。

次の表に、データイベントを記録できる Amazon SWF リソースタイプを示します。[データイベントタイプ] 列には、CloudTrail コンソールの [データイベントタイプ] リストから選択する値が表示されます。resources.type 値列には、 AWS CLI または CloudTrail APIs を使用して高度なイベントセレクタを設定するときに指定する `resources.type`値が表示されます。CloudTrail に記録されたデータ API 列には、リソース タイプの CloudTrail にログ記録された API コールが表示されます。

eventName、readOnly、および resources.ARN フィールドでフィルタリングして、自分にとって重要なイベントのみをログに記録するように高度なイベントセレクタを設定できます。オブジェクトの詳細については、「AWS CloudTrail API リファレンス」の「AdvancedFieldSelector」を参照してください。

データイベントタイプ	resources.type 値	CloudTrail にログ記録されたデータ API
SWF ドメイン	AWS::SWF::Domain	ワークフローイベント • CountClosedWorkflowExecutions • CountOpenWorkflowExecutions • DescribeWorkflowExecution • ListClosedWorkflowExecutions • ListOpenWorkflowExecutions • GetWorkflowExecutionHistory • RequestCancelWorkflowExecution • SignalWorkflowExecution • StartWorkflowExecution • TerminateWorkflowExecution タスクイベント • CountPendingActivityTasks • PollForDecisionTask • PollForActivityTask • RecordActivityTaskHeartbeat

データイベントタイプ	resources.type 値	CloudTrail にログ記録されたデータ API
		• RespondActivityTaskCanceled • RespondActivityTaskCompleted • RespondActivityTaskFailed • RespondDecisionTaskCompleted 決定イベント • CancelTimer • CancelWorkflowExecution • CompleteWorkflowExecution • ContinueAsNewWorkflowExecution • FailWorkflowExecution • RecordMarker • RequestCancelActivityTask • RequestCancelExternalWorkflowExecution • ScheduleActivityTask • ScheduleLambdaFunction • SignalExternalWorkflowExecution • StartChildWorkflowExecution • StartTimer

> **CloudTrail イベントと RespondDecisionTaskCompleted**
>
> RespondDecisionTaskCompleted アクションは、リクエストペイロードの決定のリストを取得します。完了した呼び出しは N+1 CloudTrail データイベントを出力します。1 つは決定ごとに、もう 1 つは API コール自体に対して出力されます。データイベントと API イベントはすべて同じリクエスト ID を持ちます。

CloudTrail でイベントを管理する

管理イベントは、 のリソースで実行される管理オペレーションに関する情報を提供します AWS アカウント。これらのイベントは、コントロールプレーンオペレーションとも呼ばれます。CloudTrail は、デフォルトで管理イベントをログ記録します。

Amazon Simple Workflow Service は、次のコントロールプレーンオペレーションを管理イベントとして CloudTrail に記録します。

ドメインイベント

- RegisterDomain
- DescribeDomain
- ListDomains
- DeprecateDomain
- UndeprecateDomain

アクティビティイベント

- RegisterActivityType
- DescribeActivityType
- ListActivityTypes
- DeprecateActivityType
- UndeprecateActivityType
- DeleteActivityType

WorkflowType イベント

- RegisterWorkflowType
- DescribeWorkflowType
- ListWorkflowTypes
- DeprecateWorkflowType
- UndeprecateWorkflowType
- DeleteWorkflowType

タグイベント

- TagResource
- UntagResource
- ListTagsforResource

イベントの例

各イベントは任意の送信元からの単一のリクエストを表し、リクエストされた API オペレーション、オペレーションの日時、リクエストパラメータなどに関する情報を含みます。CloudTrail ログファイルは、パブリック API コールの順序付けられたスタックトレースではないため、イベントは特定の順序で表示されません。

次の例は、CountClosedWorkflowExecutions オペレーションを示す CloudTrail イベントを示しています。


```
{
    "eventVersion": "1.09",
    "userIdentity": {
      "type": "AssumedRole",
      "principalId": "1234567890abcdef02345:admin",
      "arn": "arn:aws:sts::111122223333:assumed-role/Admin/admin",
      "accountId": "111122223333",
      "accessKeyId": "abcdef01234567890abc",
      "sessionContext": {
        "sessionIssuer": {
          "type": "Role",
          "principalId": "1234567890abcdef02345",
          "arn": "arn:aws:iam::111122223333:role/Admin",
          "accountId": "111122223333",
          "userName": "Admin"
```

```
        },
        "attributes": {
          "creationDate": "2023-11-23T16:37:38Z",
          "mfaAuthenticated": "false"
        }
      }
    },
    "eventTime": "2023-11-23T17:52:46Z",
    "eventSource": "swf.amazonaws.com",
    "eventName": "CountClosedWorkflowExecutions",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "198.51.100.42",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.42",
    "requestParameters": {
      "domain": "nsg-domain",
      "closeTimeFilter": {
        "oldestDate": "Nov 23, 2023 5:52:46 PM",
        "latestDate": "Nov 23, 2023 5:52:46 PM"
      }
    },
    "responseElements": null,
    "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
    "readOnly": true,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::SWF::Domain",
        "ARN": "arn:aws:swf:us-east-1:111122223333:/domain/nsg-domain"
      }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data",
    "tlsDetails": {
      "clientProvidedHostHeader": "swf.example.amazondomains.com"
    }
  }
```


CloudTrail レコードの内容については、「AWS CloudTrail ユーザーガイド」の「CloudTrail record contents」を参照してください。

EventBridge for Amazon SWF 実行状況の変更

Amazon EventBridge を使用して、 AWS リソースの状態変更やイベントに応答します。Amazon SWF がイベントを発行すると、常にアカウントのデフォルトの EventBridge イベントバスに移動します。イベントのルールを作成し、それをデフォルトのイベントバスに関連付け、EventBridge がルールに一致するイベントを受信したときに実行するターゲットアクションを指定できます。これにより、`GetWorkflowExecutionHistory` API を使用して継続的にポーリングすることなく、ワークフローをモニタリングできます。ワークフロー実行の変更に基づいて、EventBridge ターゲットを使用して AWS Lambda 関数を呼び出したり、Amazon Simple Notification Service (Amazon SNS) トピックにメッセージを発行したりできます。

実行ステータス変更イベントの全内容を確認するには、`DescribeWorkflowExecution` を使用します。

詳細については、「Amazon EventBridge User Guide」(Amazon EventBridge ユーザーガイド) を参照してください。

EventBridge イベント

履歴イベントタイプには、実行状態の変更が含まれます。各イベントの `detail` セクションには、少なくとも次のパラメータが含まれています。

- `eventId`: GetWorkflowExecutionHistory によって表示されるイベント ID。
- `workflowExecutionDetail`: イベントが出力されたときのワークフローの状態。
- `eventType`: 履歴イベントタイプ。以下のいずれかを指定します。
  - `ActivityTaskCanceled`
  - `ActivityTaskFailed`
  - `ActivityTaskTimedOut`
  - `WorkflowExecutionCanceled`
  - `WorkflowExecutionCompleted`
  - `WorkflowExecutionFailed`
  - `WorkflowExecutionStarted`
  - `WorkflowExecutionTerminated`
  - `WorkflowExecutionTimedOut`
  - `WorkflowExecutionContinuedAsNew`
  - `CancelTimerFailed`

- CancelWorkflowExecutionFailed
- ChildWorkflowExecutionFailed
- ChildWorkflowExecutionTimedOut
- CompleteWorkflowExecutionFailed
- ContinueAsNewWorkflowExecutionFailed
- DecisionTaskTimedOut
- FailWorkflowExecutionFailed
- RecordMarkerFailed
- RequestCancelActivityTaskFailed
- RequestCancelExternalWorkflowExecutionFailed
- ScheduleActivityTaskFailed
- SignalExternalWorkflowExecutionFailed
- StartActivityTaskFailed
- StartChildWorkflowExecutionFailed
- StartTimerFailed
- TimerCanceled
- LambdaFunctionFailed
- LambdaFunctionTimedOut
- StartLambdaFunctionFailed
- ScheduleLambdaFunctionFailed

Amazon SWF イベントの例

以下は、Amazon SWF が EventBridge にイベントを送信する例です。

トピック

いずれの場合も、イベントデータの detail セクションに、DescribeWorkflowExecution API と同じ情報が含まれています。executionStatus フィールドは、イベントが送信されたときの実行の状態 (OPEN または CLOSED) を示します。

実行のスタート


```
{
  "version": "0",
  "id": "444444444444",
  "detail-type": "Simple Workflow Execution State Change",
  "source": "aws.swf",
  "account": "444444444444",
  "time": "2020-05-08T15:57:38Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:swf:us-east-1:444444444444:/domain/SimpleWorkflowUserSimulator"
  ],
  "detail": {
    "eventId": 1,
    "eventType": "WorkflowExecutionStarted",
    "workflowExecutionDetail": {
      "executionInfo": {
        "execution": {
          "workflowId": "123456789012",
          "runId": "AKIAIOSFODNN7EXAMPLE"
        },
        "workflowType": {
          "name": "SimpleWorkflowUserSimulator",
          "version": "myWorkflow"
        },
        "startTimestamp": 1588953458484,
        "closeTimestamp": null,
        "executionStatus": "OPEN",
        "closeStatus": null,
        "parent": null,
        "parentExecutionArn": null,
        "tagList": null,
        "cancelRequested": false
      },
      "executionConfiguration": {
        "taskStartToCloseTimeout": "60",
        "executionStartToCloseTimeout": "1000",
        "taskList": {
```

```
          "name": "444444444444"
        },
        "taskPriority": null,
        "childPolicy": "ABANDON",
        "lambdaRole": "arn:aws:iam::444444444444:role/BasicSWFLambdaExecution"
      },
      "openCounts": {
        "openActivityTasks": 0,
        "openDecisionTasks": 1,
        "openTimers": 0,
        "openChildWorkflowExecutions": 0,
        "openLambdaFunctions": 0
      },
      "latestActivityTaskTimestamp": null,
    }
  }
}
```

実行の完了

```
{
  "version": "0",
  "id": "1111-2222-3333",
  "detail-type": "Simple Workflow Execution State Change",
  "source": "aws.swf",
  "account": "444455556666",
  "time": "2020-05-08T15:57:39Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:swf:us-east-1:444455556666:/domain/SimpleWorkflowUserSimulator"
  ],
  "detail": {
    "eventId": 35,
    "eventType": "WorkflowExecutionCompleted",
    "workflowExecutionDetail": {
      "executionInfo": {
        "execution": {
          "workflowId": "1234-5678-9012",
          "runId": "777788889999"
        },
        "workflowType": {
          "name": "SimpleWorkflowUserSimulator",
          "version": "myWorkflow"
```

```
        },
        "startTimestamp": 1588953458820,
        "closeTimestamp": 1588953459448,
        "executionStatus": "CLOSED",
        "closeStatus": "COMPLETED",
        "parent": null,
        "parentExecutionArn": null,
        "tagList": null,
        "cancelRequested": false
      },
      "executionConfiguration": {
        "taskStartToCloseTimeout": "60",
        "executionStartToCloseTimeout": "1000",
        "taskList": {
          "name": "1111-1111-1111"
        },
        "taskPriority": null,
        "childPolicy": "ABANDON",
        "lambdaRole": "arn:aws:iam::444455556666:role/BasicSWFLambdaExecution"
      },
      "openCounts": {
        "openActivityTasks": 0,
        "openDecisionTasks": 0,
        "openTimers": 0,
        "openChildWorkflowExecutions": 0,
        "openLambdaFunctions": 0
      },
      "latestActivityTaskTimestamp": 1588953459402,
    }
  }
}
```

実行の失敗

```
{
  "version": "0",
  "id": "1111-2222-3333",
  "detail-type": "Simple Workflow Execution State Change",
  "source": "aws.swf",
  "account": "444455556666",
  "time": "2020-05-08T15:57:38Z",
  "region": "us-east-1",
  "resources": [
```

```
    "arn:aws:swf:us-east-1:444455556666:/domain/SimpleWorkflowUserSimulator"
  ],
  "detail": {
    "eventId": 11,
    "eventType": "WorkflowExecutionFailed",
    "workflowExecutionDetail": {
      "executionInfo": {
        "execution": {
          "workflowId": "1234-5678-9012",
          "runId": "777788889999"
        },
        "workflowType": {
          "name": "SimpleWorkflowUserSimulator",
          "version": "myWorkflow"
        },
        "startTimestamp": 1588953158481,
        "closeTimestamp": 1588953458560,
        "executionStatus": "CLOSED",
        "closeStatus": "FAILED",
        "parent": null,
        "parentExecutionArn": null,
        "tagList": null,
        "cancelRequested": false
      },
      "executionConfiguration": {
        "taskStartToCloseTimeout": "60",
        "executionStartToCloseTimeout": "1000",
        "taskList": {
          "name": "1111-1111-1111"
        },
        "taskPriority": null,
        "childPolicy": "ABANDON",
        "lambdaRole": "arn:aws:iam::444455556666:role/BasicSWFLambdaExecution"
      },
      "openCounts": {
        "openActivityTasks": 0,
        "openDecisionTasks": 0,
        "openTimers": 0,
        "openChildWorkflowExecutions": 0,
        "openLambdaFunctions": 0
      },
      "latestActivityTaskTimestamp": null,
    }
  }
```

```
}
```

実行のタイムアウト

```
{
  "version": "0",
  "id": "1111-2222-3333",
  "detail-type": "Simple Workflow Execution State Change",
  "source": "aws.swf",
  "account": "444455556666",
  "time": "2020-05-05T17:26:30Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:swf:us-east-1:444455556666:/domain/SimpleWorkflowUserSimulator"
  ],
  "detail": {
    "eventId": 6,
    "eventType": "WorkflowExecutionTimedOut",
    "workflowExecutionDetail": {
      "executionInfo": {
        "execution": {
          "workflowId": "1234-5678-9012",
          "runId": "777788889999"
        },
        "workflowType": {
          "name": "SimpleWorkflowUserSimulator",
          "version": "myWorkflow"
        },
        "startTimestamp": 1588698073748,
        "closeTimestamp": 1588699590745,
        "executionStatus": "CLOSED",
        "closeStatus": "TIMED_OUT",
        "parent": null,
        "parentExecutionArn": null,
        "tagList": null,
        "cancelRequested": false
      },
      "executionConfiguration": {
        "taskStartToCloseTimeout": "60",
        "executionStartToCloseTimeout": "1000",
        "taskList": {
          "name": "1111-1111-1111"
        },
```

```
        "taskPriority": null,
        "childPolicy": "ABANDON",
        "lambdaRole": "arn:aws:iam::444455556666:role/BasicSWFLambdaExecution"
      },
      "openCounts": {
        "openActivityTasks": 1,
        "openDecisionTasks": 0,
        "openTimers": 0,
        "openChildWorkflowExecutions": 0,
        "openLambdaFunctions": 0
      },
      "latestActivityTaskTimestamp": 1588699585802,
    }
  }
}
```

実行の終了

```
{
  "version": "0",
  "id": "1111-2222-3333",
  "detail-type": "Simple Workflow Execution State Change",
  "source": "aws.swf",
  "account": "444455556666",
  "time": "2020-05-08T22:37:26Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:swf:us-east-1:444455556666:/domain/canary"
  ],
  "detail": {
    "eventId": 48,
    "eventType": "WorkflowExecutionTerminated",
    "workflowExecutionDetail": {
      "executionInfo": {
        "execution": {
          "workflowId": "1234-5678-9012",
          "runId": "777788889999"
        },
        "workflowType": {
          "name": "1111-1111-1111",
          "version": "1.3"
        },
        "startTimestamp": 1588977445279,
```

```
        "closeTimestamp": 1588977446062,
        "executionStatus": "CLOSED",
        "closeStatus": "TERMINATED",
        "parent": null,
        "parentExecutionArn": null,
        "tagList": null,
        "cancelRequested": false
      },
      "executionConfiguration": {
        "taskStartToCloseTimeout": "60",
        "executionStartToCloseTimeout": "120",
        "taskList": {
          "name": "1111-1111-1111-2222-2222-2222"
        },
        "taskPriority": null,
        "childPolicy": "TERMINATE",
        "lambdaRole": null
      },
      "openCounts": {
        "openActivityTasks": 0,
        "openDecisionTasks": 1,
        "openTimers": 0,
        "openChildWorkflowExecutions": 0,
        "openLambdaFunctions": 0
      },
      "latestActivityTaskTimestamp": 1588977445882,
    }
  }
}
```

Amazon Simple Workflow Service AWS User Notifications での の使用

AWS User Notifications を使用して配信チャネルを設定し、Amazon Simple Workflow Service イベントに関する通知を受け取ることができます。指定したルールにイベントが一致すると、通知を受け取ります。イベントの通知は、Eメール、チャットアプリケーション内の Amazon Q Developerのチャット通知、AWS Console Mobile Applicationのプッシュ通知などの複数のチャネルで受け取ることができます。また、コンソール通知センターの通知を確認することもできます。 User Notifications は集約をサポートしているため、特定のイベント中に受け取る通知の数を減らすことができます。

Amazon Simple Workflow Service のコンプライアンス検証

Amazon Simple Workflow Service のセキュリティとコンプライアンスは、 AWS のさまざまなコンプライアンスプログラムのパートとして、第三者の監査機関によって評価されます。これらのプログラムには、SOC、PCI、FedRAMP、HIPAA などがあります。

特定のコンプライアンスプログラムの対象となる AWS サービスのリストについては、「コンプライアンスAWS プログラムによる対象範囲内のサービスコンプライアンス」を参照してください。一般的な情報については、AWS 「 Compliance ProgramsAssurance」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、AWS 「Artifact でのレポートのダウンロード」を参照してください。

Amazon SWF を使用する際のお客様のコンプライアンス責任は、お客様のデータの機密性や貴社のコンプライアンス目的、適用される法律および規制によって決まります。 AWS では、コンプライアンスに役立つ以下のリソースを提供しています。

- セキュリティとコンプライアンスのクイックスタートガイド – これらのデプロイガイドでは、アーキテクチャ上の考慮事項について説明し、セキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイする手順について説明します AWS。
- 「Architecting for HIPAA Security and Compliance」ホワイトペーパー – このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。
- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- 「 デベロッパーガイド」の「ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が内部プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub CSPM – この AWS サービスは、 内のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立ちます。

Amazon Simple Workflow Service の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。 AWS リージョンは、低レイテンシー、高スループット、高度に冗長なネットワークで接続された、物理的に分離および分離された複数のアベイラビリティーゾーンを提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーする

アプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS「グローバルインフラストラクチャ」を参照してください。

Amazon SWF には、 AWS グローバルインフラストラクチャに加えて、データの耐障害性とバックアップのニーズをサポートするのに役立つ機能がいくつか用意されています。

Amazon Simple Workflow Service でのインフラストラクチャセキュリティ

マネージドサービスとして、 は AWS グローバルネットワークセキュリティで保護されています。AWS セキュリティサービスと がインフラストラクチャ AWS を保護する方法については、AWS「 クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスを使用して環境を AWS 設計するには、「Security Pillar AWS Well-Architected Framework」の「Infrastructure Protection」を参照してください。

AWS が発行した API コールを使用して、ネットワーク経由で にアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは Java 7 以降など、ほとんどの最新システムでサポートされています。

これらの API オペレーションは任意のネットワークの場所から呼び出すことができますが、Amazon SWF ではリソースベースのアクセスポリシーがサポートされているため、ソース IP アドレスに基づく制限を含めることができます。また、Amazon SWF ポリシーを使用して、特定の Amazon Virtual Private Cloud (Amazon VPC) エンドポイントまたは特定の VPC からのアクセスを管理することもできます。これにより、実質的に、ネットワーク内の特定の VPC からのみ、特定の Amazon SWF リソースへの AWS ネットワークアクセスが分離されます。

Amazon Simple Workflow Service での設定と脆弱性の分析

設定と IT コントロールは、 AWS とお客様の間の責任共有です。詳細については、 AWS「 責任共有モデル」を参照してください。

Amazon Simple Workflow Service AWS CLI での の使用

Amazon Simple Workflow Service の機能の多くは AWS CLIからアクセスできます。 AWS CLI は、で Amazon SWF を使用する代わりに、 AWS マネジメントコンソール 場合によっては Amazon SWF API と でプログラミングすることもできます AWS Flow Framework。

たとえば、 を使用して新しいワークフロータイプ AWS CLI を登録できます。

```
aws swf register-workflow-type --domain MyDomain --name "MySimpleWorkflow" --workflow-version "v1"
```

また、登録されたワークフロータイプをリストできます。

```
aws swf list-workflow-types --domain MyDomain --registration-status REGISTERED
```

この JSON のデフォルト出力の例を以下に示します。

```
{
    "typeInfos": [
        {
            "status": "REGISTERED",
            "creationDate": 1377471607.752,
            "workflowType": {
                "version": "v1",
                "name": "MySimpleWorkflow"
            }
        },
        {
            "status": "REGISTERED",
            "creationDate": 1371454149.598,
            "description": "MyDomain subscribe workflow",
            "workflowType": {
                "version": "v3",
                "name": "subscribe"
            }
        }
    ]
}
```

の Amazon SWF コマンド AWS CLI を使用すると、ワークフロー実行の開始と管理、アクティビティタスクのポーリング、タスクハートビートの記録などを行うことができます。Amazon SWF コ

マンドの完全なリスト、および使用可能な引数の説明とその使用法を示す例については、AWS CLI コマンドリファレンス の「Amazon SWF コマンド」を参照してください。

AWS CLI コマンドは Amazon SWF API に密接に従うため、 AWS CLI を使用して基盤となる Amazon SWF API について学習できます。また、既存の API の知識を使用して、コマンドラインでコードのプロトタイプを作成したり Amazon SWF アクションを実行したりできます。

の詳細については AWS CLI、 AWS Command Line Interface ユーザーガイドを参照してください。

Amazon SWF API を使用する

で説明されている AWS SDKs の使用に加えてAWS SDKs、HTTP API を直接使用できます。

API を使用するには、ドメイン、ワークフローおよびアクティビティに使用するリージョンと一致する SWF エンドポイントに HTTP リクエストを送信します。Amazon SWF の HTTP リクエストの作成の詳細については、「Amazon SWF に対する HTTP リクエストの実行」を参照してください。

このセクションでは、Amazon SWF を使用してワークフローを開発するための HTTP API の使用に関するベーシック情報について説明します。タイマーの使用、CloudTrail を使用したロギング、ワークフローのタグ付けなどのより高度な機能については、Amazon SWF の基本的なワークフローの概念 セクションで説明しています。

トピック

- Amazon SWF に対する HTTP リクエストの実行
- Amazon SWF アクションのカテゴリ別リスト
- Amazon SWF によるドメインの登録
- Amazon SWF でのタイムアウト値の設定
- Amazon SWF でワークフロータイプを登録する
- Amazon SWF でのアクティビティタイプの登録
- AWS Lambda Amazon SWF のタスク
- Amazon SWF でのアクティビティワーカーの開発
- Amazon SWF でのディサイダーの開発
- Amazon SWF でのワークフローの開始
- Amazon SWF でのタスク優先度の設定
- Amazon SWF でのエラーの処理

Amazon SWF に対する HTTP リクエストの実行

AWS SDKs のいずれかを使用しない場合は、POST リクエストメソッドを使用して、HTTP 経由で Amazon Simple Workflow Service (Amazon SWF) オペレーションを実行できます。POST メソッドでは、リクエストのヘッダーでオペレーションを指定し、リクエストの本文に、オペレーションのデータを JSON 形式で入力します。

HTTP ヘッダーの内容

Amazon SWF では、HTTP リクエストのヘッダーに次の情報を入力する必要があります。

- host Amazon SWF エンドポイント。
- x-amz-date タイムスタンプは HTTP Dateヘッダーまたは AWS x-amz-date header で指定する必要があります (一部の HTTP クライアントライブラリでは Dateヘッダーを設定できません)。x-amz-date ヘッダーがある場合、リクエストの認証時に Date ヘッダーは無視されます。

  日付は、HTTP/1.1 RFC で規定されている次の 3 つ形式のいずれかで指定する必要があります。

  - Sun, 06 Nov 1994 08:49:37 GMT (RFC 822、RFC 1123 により更新)
  - Sunday, 06-Nov-94 08:49:37 GMT (RFC 850、RFC 1036 により廃止)
  - Sun Nov 6 08:49:37 1994 (ANSI C asctime() 形式)
- x-amzn-authorization 形式の署名付きリクエストパラメータ。

```
AWS3 AWSAccessKeyId=####,Algorithm=HmacSHA256, [,SignedHeaders=Header1;Header2;...]
Signature=S(StringToSign)
```

  AWS3 – これは、リクエストの署名に使用される認証バージョンを示す AWS 実装固有のタグです (現在、Amazon SWF の場合、この値は常に ですAWS3)。

  AWSAccessKeyId – AWS アクセスキー ID。

  Algorithm - HmacSHA256 や HmacSHA1 など、署名対象の文字列の HMAC-SHA 値を作成するために使用されるアルゴリズム。

  Signature - Base64 (アルゴリズム (StringToSign、SigningKey)。詳細については、「Amazon SWF の HMAC-SHA 署名の生成」を参照してください。

  SignedHeaders - (オプション) 存在する場合、正規化された HttpHeaders 計算で使用されるすべての HTTP ヘッダーのリストを含める必要があります。1 つのセミコロンの文字 (;) (ASCII 文字 59) をリスト値の区切り文字として使用する必要があります。
- x-amz-target - 次の形式で指定する、リクエストの送信先サービスおよびデータのオペレーション

  com.amazonaws.swf.service.model.SimpleWorkflowService. + *<action>*

例: com.amazonaws.swf.service.model.SimpleWorkflowService.RegisterDomain

- content-type - タイプは JSON に、文字セットは application/json; charset=UTF-8 に指定する必要があります。

次に、ドメインを作成する HTTP リクエストのサンプルヘッダーの例を示します。

```
POST http://swf.us-east-1.amazonaws.com/ HTTP/1.1
Host: swf.us-east-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.25) Gecko/20111212
 Firefox/3.6.25 ( .NET CLR 3.5.30729; .NET4.0E)
Accept: application/json, text/javascript, */*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Content-Type: application/json; charset=UTF-8
X-Requested-With: XMLHttpRequest
X-Amz-Date: Fri, 13 Jan 2012 18:42:12 GMT
X-Amz-Target: com.amazonaws.swf.service.model.SimpleWorkflowService.RegisterDomain
Content-Encoding: amz-1.0
X-Amzn-Authorization: AWS3
 AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE,Algorithm=HmacSHA256,SignedHeaders=Host;X-Amz-
Date;X-Amz-Target;Content-Encoding,Signature=tzjkF55lxAxPhzp/BRGFYQRQRq6CqrM254dTDE/
EncI=
Referer: http://swf.us-east-1.amazonaws.com/explorer/index.html
Content-Length: 91
Pragma: no-cache
Cache-Control: no-cache

{"name": "867530902",
 "description": "music",
 "workflowExecutionRetentionPeriodInDays": "60"}
```

以下に、対応する HTTP レスポンスの例を示します。

```
HTTP/1.1 200 OK
Content-Length: 0
Content-Type: application/json
x-amzn-RequestId: 4ec4ac3f-3e16-11e1-9b11-7182192d0b57
```

HTTP 本文の内容

HTTP リクエストの本文には、HTTP リクエストのヘッダーで指定されたオペレーションのデータが含まれます。JSON データ形式を使用すると、データ値とデータ構造を同時に送信できます。エレメントは、ブラケット表記を使用することで他のエレメント内にネストすることができます。次の例では、指定された 2 つのポイント間で開始されたすべてのワークフローの実行を、Unix Time 表記を使用して一覧表示するリクエストを示しています。

```
{
 "domain": "867530901",
 "startTimeFilter":
 {
   "oldestDate": 1325376070,
  "latestDate": 1356998399
 },
 "tagFilter":
 {
   "tag": "music purchase"
 }
}
```

Amazon SWF JSON のリクエストと応答のサンプル

次の例は、以前に作成したドメインの説明についての Amazon SWF へのリクエストを示しています。次に、Amazon SWF 応答が表示されます。

HTTP POST リクエスト

```
POST http://swf.us-east-1.amazonaws.com/ HTTP/1.1
Host: swf.us-east-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.25) Gecko/20111212
 Firefox/3.6.25 ( .NET CLR 3.5.30729; .NET4.0E)
Accept: application/json, text/javascript, */*
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Content-Type: application/json; charset=UTF-8
X-Requested-With: XMLHttpRequest
X-Amz-Date: Sun, 15 Jan 2012 03:13:33 GMT
X-Amz-Target: com.amazonaws.swf.service.model.SimpleWorkflowService.DescribeDomain
```

```
Content-Encoding: amz-1.0
X-Amzn-Authorization: AWS3
 AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE,Algorithm=HmacSHA256,SignedHeaders=Host;X-Amz-
Date;X-Amz-Target;Content-
Encoding,Signature=IFJtq3M366CHqMlTpyqYqd9z0ChCoKDC5SCJBsLifu4=
Referer: http://swf.us-east-1.amazonaws.com/explorer/index.html
Content-Length: 21
Pragma: no-cache
Cache-Control: no-cache

{"name": "867530901"}
```

Amazon SWF 応答

```
HTTP/1.1 200 OK
Content-Length: 137
Content-Type: application/json
x-amzn-RequestId: e86a6779-3f26-11e1-9a27-0760db01a4a8

{"configuration":
  {"workflowExecutionRetentionPeriodInDays": "60"},
 "domainInfo":
  {"description": "music",
   "name": "867530901",
   "status": "REGISTERED"}
}
```

プロトコル (`HTTP/1.1`) の後にステータスコード (`200`) が付いていることに注意してください。コードの値 `200` は、オペレーションが成功したことを示します。

Amazon SWF は null 値をシリアル化しません。JSON パーサーをリクエストに対する null 値をシリアル化するように設定している場合、Amazon SWF ではそれらの値が無視されます。

Amazon SWF の HMAC-SHA 署名の生成

Amazon SWF へのリクエストはすべて認証されている必要があります。 AWS SDKsリクエストに署名し、トークンベースの認証を管理します。ただし、独自の HTTP `POST` リクエストを作成する場合は、リクエストの認証の一部として HTTP `POST` `Header` コンテンツの `x-amzn-authorization` 値を作成する必要があります。

ヘッダーの書式設定の詳細については、「HTTP ヘッダーの内容」を参照してください。 AWS バージョン 3 署名の AWS SDK for Java 実装については、AWSSigner.java クラスを参照してください。

リクエスト署名の作成

HMAC-SHA リクエスト署名を作成する前に、 AWS 認証情報 (アクセスキー ID とシークレットキー) を取得する必要があります。

> **Important**
>
> SHA1 または SHA256 のいずれかを使用してリクエストに署名できます。ただし、署名プロセス全体で同じ方法を使用するようにしてください。選択するメソッドは、HTTP ヘッダーの `Algorithm` 名の値と一致する必要があります。

リクエストの署名を作成するには

1. HTTP リクエストヘッダーの正規形式を作成します。HTTP ヘッダーの正規形式には次のものが含まれます。

   - `host`
   - `x-amz-` で始まるヘッダー要素

   含まれるヘッダーの詳細については、「HTTP ヘッダーの内容」を参照してください。

   a. 各ヘッダー名と値のペアについて、ヘッダー名 (ヘッダー値ではない) を小文字に変換します。

   b. カンマ区切りのヘッダー値にヘッダー名のマップを作成します。

   ```
   x-amz-example: value1
   x-amz-example: value2  =>  x-amz-example:value1,value2
   ```

   詳細については、「RFC 2616 のセクション 4.2」を参照してください。

   c. ヘッダー名と値のペアごとに、名前と値のペアを `headerName:headerValue` 形式の文字列に変換します。`headerName` と `headerValue` の両方の先頭と末尾にある空白を削除します。コロンの前後に空白が無いようにします。

   ```
   x-amz-example1:value1,value2
   x-amz-example2:value3
   ```

   d. 最後の文字列も含め、変換された各文字列の後に改行 (U+000A) を挿入します。

e. 変換された文字列のコレクションをヘッダー名ごとに並べ替えます。

2. 以下の項目を含む署名文字列の値を作成します。

   - 1 行目: HTTP メソッド (POST)、その後に改行が続きます。
   - 2 行目: リクエスト URI (/)、その後に改行が続きます。
   - 3 行目: 空の文字列、その後に改行が続きます。

   > **Note**
   >
   > 通常、クエリ文字列はここに表示されますが、Amazon SWF ではクエリ文字列を使用しません。

   - 4–n 行目: ステップ 1 で作成した正規化されたリクエストヘッダーを表す文字列。改行が続きます。この改行はヘッダーと HTTP リクエストの本文の間に空白行を作成します。詳細については、「RFC 2616」を参照してください。
   - リクエストの本文、改行は続きません。

3. 署名文字列の値の SHA256 または SHA1 ダイジェストを計算します。プロセス全体で同じ SHA メソッドを使用します。

4. 前のステップで得られた値の SHA256 または SHA1 ダイジェスト (使用した方法によって異なります) と、 API アクションを使用して Security Token Service からの一時的なシークレットアクセスキーを使用して、HMAC-SHA をコンピューティングおよび Base64-encodeします。SHA1 AWS `GetSessionToken`

   > **Note**
   >
   > Amazon SWF では、Base64 エンコードされた HMAC-SHA 値の末尾に等号 (=) が必要です。Base64 エンコードルーチンに等号が追加されていない場合は、値の最後に 1 を追加します。

   Amazon SWF およびその他の AWS サービスで一時的なセキュリティ認証情報を使用する方法の詳細については、IAM ユーザーガイドのAWS 「IAM と連携するサービス」を参照してください。

5. 結果の値を、Amazon SWF への HTTP リクエストの `x-amzn-authorization` ヘッダーの `Signature` 名の値として配置します。

6. Amazon SWF はリクエストを認証し、指定されたオペレーションを実行します。

Amazon SWF アクションのカテゴリ別リスト

このセクションでは、Amazon SWF アプリケーションプログラミングインターフェイス (API) における Amazon SWF アクションのリファレンストピックを示します。これらは機能カテゴリ別に一覧表示されます。

アクションの アルファベット順 のリストについては、「Amazon Simple Workflow Service API Reference」(Amazon Simple Workflow Service API リファレンス) を参照してください。

トピック

- アクティビティに関連するアクション
- ディサイダーに関連するアクション
- ワークフロー実行に関連するアクション
- 管理に関するアクション
- アクションの可視化

アクティビティに関連するアクション

アクティビティワーカーは PollForActivityTask を使用して新しいアクティビティタスクを取得します。ワーカーは Amazon SWF からアクティビティタスクを受け取った後、タスクを実行し、成功した場合 RespondActivityTaskCompleted または失敗した場合は、RespondActivityTaskFailed を使用して応答します。

アクティビティワーカーによって実行されるアクションを以下に示します。

- PollForActivityTask
- RespondActivityTaskCompleted
- RespondActivityTaskFailed
- RespondActivityTaskCanceled
- RecordActivityTaskHeartbeat

ディサイダーに関連するアクション

ディサイダーは PollForDecisionTask を使用して決定タスクを取得します。ディサイダーは Amazon SWF からデシジョンタスクを受け取った後、そのワークフロー実行履歴を調べて、次の

動作を決定します。決定タスクを完了するために RespondDecisionTaskCompleted が呼び出され、ゼロ以上の次の判断が提供されます。

ディサイダーによって実行されるアクションを以下に示します。

- PollForDecisionTask
- RespondDecisionTaskCompleted

ワークフロー実行に関連するアクション

次のアクションはワークフロー実行で動作します。

- RequestCancelWorkflowExecution
- StartWorkflowExecution
- SignalWorkflowExecution
- TerminateWorkflowExecution

管理に関するアクション

Amazon SWF コンソールから管理タスクを実行できますが、このセクションのアクションを使用して機能を自動化したり、独自の管理ツールを構築したりできます。

アクティビティ管理

- RegisterActivityType
- DeprecateActivityType
- UndeprecateActivityType
- DeleteActivityType

ワークフロー管理

- RegisterWorkflowType
- DeprecateWorkflowType
- UndeprecateWorkflowType

- DeleteWorkflowType

ドメイン管理

これらのアクションを使用して、Amazon SWF ドメインの登録および廃止ができます。

- RegisterDomain
- DeprecateDomain
- UndeprecateDomain

これらのドメイン管理アクションの詳細と例については、「Amazon SWF によるドメインの登録」を参照してください。

ワークフロー実行管理

- RequestCancelWorkflowExecution
- TerminateWorkflowExecution

アクションの可視化

Amazon SWF コンソールから可視性アクションを実行できますが、このセクションのアクションを使用して、独自のコンソールまたは管理ツールを作成できます。

アクティビティの可視化

- ListActivityTypes
- DescribeActivityType

ワークフローの可視化

- ListWorkflowTypes
- DescribeWorkflowType

ワークフロー実行の可視化

- DescribeWorkflowExecution

- ListOpenWorkflowExecutions
- ListClosedWorkflowExecutions
- CountOpenWorkflowExecutions
- CountClosedWorkflowExecutions
- GetWorkflowExecutionHistory

ドメインの可視化

- ListDomains
- DescribeDomain

タスクリストの可視化

- CountPendingActivityTasks
- CountPendingDecisionTasks

Amazon SWF によるドメインの登録

ワークフローとアクティビティの種類とワークフローの実行自体はすべてドメインにスコープされています。ドメインは、同じアカウント内の他のメンバーからタイプのセット、実行、およびタスクリストを分離します。

ドメインを登録するには、 を使用する AWS マネジメントコンソール か、Amazon SWF API の RegisterDomainアクションを使用します。次の例では、API を使用します。

```
https://swf.us-east-1.amazonaws.com
RegisterDomain
{
  "name" : "867530901",
  "description" : "music",
  "workflowExecutionRetentionPeriodInDays" : "60"
}
```

パラメータは JavaScript Object Notation (JSON) 形式で指定します。ここで、保持期間は 60 日間に設定されています。保持期間中、ワークフロー実行に関するすべての情報は、 AWS マネジメントコンソール または Amazon SWF API を使用した可視性オペレーションを通じて利用できます。

ドメインを登録したら、ワークフロータイプとワークフローで使用されるアクティビティタイプを登録する必要があります。登録されたドメイン名は、ワークフローおよびアクティビティタイプの登録に必要な情報の一部であるため、まずドメインを登録する必要があります。

以下の資料も参照してください。

「Amazon Simple Workflow Service API Reference」(Amazon Simple Workflow Service API リファレンス) の「RegisterDomain」

Amazon SWF でのタイムアウト値の設定

トピック

タイムアウト値のクォータ

タイムアウト値は常に秒で表され、ワークフローやアクティビティの最大実行制限である 1 年 (31536000 秒) までの範囲で、何秒にでも設定できます。特殊な値 NONE は、タイムアウトパラメータを「制限なし」または無限に設定するときに使用しますが、最大制限の 1 年は適用されます。

ワークフロー実行と決定タスクのタイムアウト

ワークフロータイプを登録するときに、ワークフローや決定タスクにタイムアウト値を設定できます。例:

```
https://swf.us-east-1.amazonaws.com
RegisterWorkflowType
{
  "domain": "867530901",
  "name": "customerOrderWorkflow",
  "version": "1.0",
  "description": "Handle customer orders",
  "defaultTaskStartToCloseTimeout": "600",
  "defaultExecutionStartToCloseTimeout": "3600",
  "defaultTaskList": { "name": "mainTaskList" },
```

```
  "defaultChildPolicy": "TERMINATE"
}
```

このワークフロータイプの登録では、defaultTaskStartToCloseTimeout を 600 秒 (10 分)、および defaultExecutionStartToCloseTimeout を 3600 秒 (1 時間) に設定します。

ワークフロータイプの登録の詳細については、「Amazon Simple Workflow Service API Reference」(Amazon Simple Workflow Service API リファレンス) の「Amazon SWF でワークフロータイプを登録する」および「RegisterWorkflowType」を参照してください。

defaultExecutionStartToCloseTimeout に設定されている値を上書きするには、executionStartToCloseTimeout を指定します。

アクティビティタスクのタイムアウト

アクティビティタイプを登録するときに、アクティビティタスクにタイムアウト値を設定できます。例:

```
https://swf.us-east-1.amazonaws.com
RegisterActivityType
{
  "domain": "867530901",
  "name": "activityVerify",
  "version": "1.0",
  "description": "Verify the customer credit",
  "defaultTaskStartToCloseTimeout": "600",
  "defaultTaskHeartbeatTimeout": "120",
  "defaultTaskList": { "name": "mainTaskList" },
  "defaultTaskScheduleToStartTimeout": "1800",
  "defaultTaskScheduleToCloseTimeout": "5400"
}
```

このアクティビティタイプの登録では、defaultTaskStartToCloseTimeout を 600 秒 (10 分)、defaultTaskHeartbeatTimeout を 120 秒 (2 分)、defaultTaskScheduleToStartTimeout を 1800 秒 (30 分)、および defaultTaskScheduleToCloseTimeout を 5400 秒 (1.5 時間) に設定します。

アクティビティタイプの登録の詳細については、「Amazon Simple Workflow Service API Reference」(Amazon Simple Workflow Service API リファレンス) の「Amazon SWF でのアクティビティタイプの登録」および「RegisterActivityType」を参照してください。

defaultTaskStartToCloseTimeout に設定されている値を上書きするには、アクティビティタスクをスケジュールを設定するときに taskStartToCloseTimeout を指定します。

以下の資料も参照してください。

Amazon SWF タイムアウトの種類

Amazon SWF でワークフロータイプを登録する

このセクションで説明する例では、Amazon SWF API を使用してワークフロータイプを登録します。登録中に指定する名前とバージョンは、ワークフロータイプの一意の識別子を形成します。指定されたドメインは、RegisterDomain API アクションを使用して既に登録されている必要があります。

次の例のタイムアウトパラメータは秒単位で指定された期間値です。defaultTaskStartToCloseTimeout パラメータでは、長さの指定子 NONE を使用して、タイムアウトなしを指定することができます。ただし、NONE の値を defaultExecutionStartToCloseTimeout に指定することはできません。実行ワークフローが実行できる時間は最大で 1 年間の制限があります。この制限を超えると、ワークフロー実行が必ずタイムアウトします。defaultExecutionStartToCloseTimeout に 1 年より大きい値を指定すると、登録が失敗します。

```
https://swf.us-east-1.amazonaws.com
RegisterWorkflowType
{
  "domain" : "867530901",
  "name" : "customerOrderWorkflow",
  "version" : "1.0",
  "description" : "Handle customer orders",
  "defaultTaskStartToCloseTimeout" : "600",
  "defaultExecutionStartToCloseTimeout" : "3600",
  "defaultTaskList" : { "name": "mainTaskList" },
  "defaultChildPolicy" : "TERMINATE"
}
```

以下の資料も参照してください。

「Amazon Simple Workflow Service API Reference」(Amazon Simple Workflow Service API リファレンス) の「RegisterWorkflowType」

Amazon SWF でのアクティビティタイプの登録

次の例では、Amazon SWF API を使用して、アクティビティタイプを登録します。登録中に指定する名前とバージョンは、ドメイン内のアクティビティタイプの一意の識別子を形成します。指定されたドメインは、RegisterDomain アクションを使用して既に登録されている必要があります。

この例のタイムアウトパラメータは秒単位で指定された期間値です。長さの指定子 NONE を使用して、タイムアウトなしを指定することができます。

```
https://swf.us-east-1.amazonaws.com
RegisterActivityType
{
  "domain" : "867530901",
  "name" : "activityVerify",
  "version" : "1.0",
  "description" : "Verify the customer credit",
  "defaultTaskStartToCloseTimeout" : "600",
  "defaultTaskHeartbeatTimeout" : "120",
  "defaultTaskList" : { "name" : "mainTaskList" },
  "defaultTaskScheduleToStartTimeout" : "1800",
  "defaultTaskScheduleToCloseTimeout" : "5400"
}
```

以下の資料も参照してください。

「Amazon Simple Workflow Service API Reference」(Amazon Simple Workflow Service API リファレンス) の「RegisterActivityType」

AWS Lambda Amazon SWF のタスク

トピック

- について AWS Lambda
- Lambda タスクを使用する利点と制限
- ワークフローでの Lambda タスクの使用

について AWS Lambda

AWS Lambda は、カスタムコードまたは Amazon S3、DynamoDB DynamoDB 、Amazon Kinesis、Amazon SNS、Amazon Cognito などのさまざまな AWS サービスによって生成されたイベ

ントに応答してコードを実行するフルマネージド型のコンピューティングサービスです。Lambda の詳細については、 デベロッパーガイドAWS Lambda を参照してください。

Amazon Simple Workflow Service は Lambda タスクを提供し、従来の Amazon SWF アクティビティの代わりに、またはそれと一緒に Lambda 関数を実行できるようにします。

> **Important**
>
> AWS アカウントは、Amazon SWF がユーザーに代わって実行した Lambda 実行 (リクエスト) に対して課金されます。Lambda の料金の詳細については、https://aws.amazon.com/lambda/pricing/ を参照してください。

Lambda タスクを使用する利点と制限

従来の Amazon SWF アクティビティの代わりに Lambda タスクを使用することには、多くの利点があります。

- Lambda タスクは、Amazon SWF アクティビティタイプのように登録またはバージョン管理する必要はありません。
- 既にワークフローで定義している既存の Lambda 関数を使用することができます。
- Lambda 関数は Amazon SWF によって直接呼び出されます。従来のアクティビティのように実行するためのワーカープログラムを実装する必要はありません。
- Lambda では、関数の実行を追跡し分析するためのメトリクスとログが用意されています。

Lambda タスクには注意すべきいくつかの制限があります。

- Lambda タスクは、Lambda をサポートする AWS リージョンでのみ実行できます。Lambda で現在サポートされているリージョンの詳細については、「Amazon Web Services General Reference」(Amazon Web Services 全般リファレンス) の「Lambda Regions and Endpoints」(Lambda リージョンとエンドポイント) を参照してください。
- Lambda タスクは現在、ベース SWF HTTP API と AWS Flow Framework for Java でのみサポートされています。現在、 AWS Flow Framework for Ruby では Lambda タスクはサポートされていません。

ワークフローでの Lambda タスクの使用

Amazon SWF ワークフローで Lambda タスクを使用するには、以下の操作が必要です。

1. Lambda 関数を呼び出すアクセス許可を Amazon SWF に付与するように IAM ロールをセットアップします。
2. IAM ロールをワークフローにアタッチします。
3. ワークフロー実行に Lambda 関数を呼び出します。

IAM ロールのセットアップ

Amazon SWF から Lambda 関数を呼び出す前に、Amazon SWF から Lambda へのアクセス権を付与する IAM ロールを準備する必要があります。次のいずれかを行うことができます。

- あらかじめ定義されたロール、AWSLambdaRole を選択して、ワークフローにアカウントに関連する Lambda 関数を呼び出すアクセス許可を付与します。
- 独自のポリシーと関連付けられたロールを定義して、Amazon リソースネーム (ARN) で指定された特定の Lambda 関数を呼び出すためのワークフローのアクセス許可を付与します。

IAM ロールのアクセス許可を制限する

Amazon SWF に提供する IAM ロールに対するアクセス許可を制限するには、リソースの信頼ポリシーの `SourceArn` および `SourceAccount` コンテキストキーを使用します。これらのキーは、指定されたドメイン ARN に属する Amazon Simple Workflow Service の実行からのみ使用されるように、IAM ポリシーの使用を制限します。これらのグローバル条件コンテキストキーの両方を、同じポリシーステートメントで使用する場合、`aws:SourceAccount` 値と `aws:SourceArn` 値の中の参照されるアカウントには、同じアカウント ID を使用する必要があります。

次の例では、`SourceArn`コンテキストキーは、アカウント `someDomain`の に属する Amazon Simple Workflow Service の実行でのみ IAM サービスロールを使用するように制限します`123456789012`。

- ステートメント 1

  プリンシパル : `"Service": "swf.amazonaws.com"`

  アクション: `sts:AssumeRole`

```
"Condition": {
   "ArnLike": {
     "aws:SourceArn": "arn:aws:swf:*:123456789012:/domain/someDomain"
   }
}
```

次の例では、SourceAccountコンテキストキーは、アカウント の Amazon Simple Workflow Service の実行でのみ IAM サービスロールを使用するように制限します123456789012。

```
"Condition": {
   "StringLike": {
     "aws:SourceAccount": "123456789012"
   }
}
```

Amazon SWF に Lambda ロールを呼び出すためのアクセスを提供する

あらかじめ定義されたロール、AWSLambdaRole を使用して、Amazon SWF ワークフローがアカウントに関連する Lambda 関数を呼び出せるようにできます。

AWSLambdaRole を使用して、Lambda 関数を呼び出すアクセス権を Amazon SWF に付与するには

1. Amazon IAM コンソールを開きます。
2. [Roles] (ロール) を選択してから [Create New Role] (ロールの新規作成) を選びます。
3. ロールに swf-lambda などの名前を付け、[Next Step] (次のステップ) を選択します。
4. [AWS サービスロール] で、[Amazon SWF]、[次のステップ] の順に選択します。
5. [ポリシーのアタッチ] 画面で、リストから [AWSLambdaRole] を選択します。
6. ロールを確認したら、[Next Step] (次のステップ) を選択してから、[Create Role] (ロールの作成) を選択します。

特定の Lambda 関数を呼び出すためのアクセス権を付与する IAM ロールの定義

ワークフローから特定の Lambda 関数を呼び出すためのアクセスを提供する場合は、独自の IAM ポリシーを定義する必要があります。

特定の Lambda 関数へのアクセスを提供する IAM ポリシーを作成するには

1. Amazon IAM コンソールを開きます。

2. [Policies] (ポリシー) を選択して、[Create Policy] (ポリシーの作成) を選択します。
3. AWS 管理ポリシーのコピーを選択し、リストから AWSLambdaRole を選択します。ポリシーが生成されます。必要に応じて名前と説明を編集することもできます。
4. [Policy Document] (ポリシードキュメント) の [Resource] (リソース) フィールドに、Lambda 関数の ARN を追加します。例:

   - リソース: `arn:aws:lambda:us-east-1:111122223333:function:hello_lambda_function`

   > **Note**
   >
   > IAM ロールでリソースを指定する方法の詳細については、「Using IAM」(IAM の使用) の「Overview of IAM Policies」(IAM ポリシーの概要) を参照してください。

5. [Create Policy] (ポリシーの作成) を選択してポリシーの作成を完了します。

新しい IAM ロールを作成するときにこのポリシーを選択し、そのロールを使用して Amazon SWF ワークフローへのアクセスを呼び出すことができます。この手順は、AWSLambdaRole ポリシーを使用してロールを作成する場合と非常によく似ています。代わりに、ロールを作成するときに独自のポリシーを選択します。

Lambda ポリシーを使用して Amazon SWF ロールを作成するには

1. Amazon IAM コンソールを開きます。
2. [Roles] (ロール) を選択してから [Create New Role] (ロールの新規作成) を選びます。
3. ロールに `swf-lambda-function` などの名前を付け、[Next Step] (次のステップ) を選択します。
4. [AWS サービスロール] で、[Amazon SWF]、[次のステップ] の順に選択します。
5. [ポリシーのアタッチ] 画面で、リストから Lambda 関数固有のポリシーを選択します。
6. ロールを確認したら、[Next Step] (次のステップ) を選択してから、[Create Role] (ロールの作成) を選択します。

IAM ロールをワークフローにアタッチします。

IAM ロールを定義したら、それを使用して Amazon SWF にアクセス許可を付与した Lambda 関数を呼び出すワークフローにそのロールをアタッチする必要があります。

ワークフローにロールをアタッチできる場所は 2 つあります。

- ワークフロータイプの登録中。次に、このロールは、そのワークフロータイプのすべての実行のデフォルトの Lambda ロールとして使用できます。
- ワークフロー実行の開始時。このロールは、このワークフローの実行中 (および実行全体を通じて) のみ使用されます。

ワークフロータイプのデフォルトの Lambda ロールを提供するには

- RegisterWorkflowType を呼び出す場合は、defaultLambdaRole フィールドを、定義したロールの ARN に設定します。

ワークフロー実行中に Lambda ロールを提供するには

- StartWorkflowExecution を呼び出す場合は、lambdaRole フィールドを、定義したロールの ARN に設定します。

> **Note**
>
> RegisterWorkflowType または StartWorkflowExecution を呼び出すアカウントに、特定のロールを使用するアクセス権限がない場合、呼び出しは OperationNotPermittedFault で失敗します。

Amazon SWF ワークフローからの Lambda 関数の呼び出し

ScheduleLambdaFunctionDecisionAttributes データ型を使用して、ワークフロー実行中に呼び出す Lambda 関数を指定できます。

RespondDecisionTaskCompleted の呼び出し中に、ScheduleLambdaFunctionDecisionAttributes を決定リストで指定します。例:

```
{
```

```
  "decisions": [{
    "ScheduleLambdaFunctionDecisionAttributes": {
      "id": "lambdaTaskId",
      "name": "myLambdaFunctionName",
      "input": "inputToLambdaFunction",
      "startToCloseTimeout": "30"
    },
  }],
}
```

以下のパラメータを設定します。

- Lambda タスクの識別子を示す id。これは 1 ~ 256 文字の文字列である必要があり、: (コロン)、/ (スラッシュ)、| (垂直棒)、またはその他の制御文字 (\u0000 ~ \u001f、\u007f ~ \u009f)、リテラル文字列 `arn` を含めることはできません。
- Lambda 関数名の名前を示す name。Amazon SWF ワークフローは、Lambda 関数を呼び出すアクセス権を付与する IAM ロールで提供する必要があります。提供される名前は、Lambda Invoke アクションなど、FunctionName パラメータの制約に従う必要があります。
- 関数のオプションの入力データを示す input。設定された場合、Lambda Invoke アクションなど、ClientContext パラメータの制約に従う必要があります。
- オプションでタスクがタイムアウトの例外により失敗するまでに関数で許可される最大期間 (秒数) を示す startToCloseTimeout。値 NONE を使用して、無制限の期間を指定することができます。

詳細については、AWS Lambda 「タスクの実装」を参照してください。

Amazon SWF でのアクティビティワーカーの開発

アクティビティワーカーは、1 つ以上のアクティビティタイプの実装を提供します。アクティビティワーカーは、Amazon SWF と通信してアクティビティタスクを受け取り、実行します。同じアクティビティタイプのアクティビティタスクを実行する複数のアクティビティワーカーのフリートを持つことができます。

Amazon SWF は、ディサイダーがアクティビティタスクをスケジュールするときに、アクティビティワーカーがアクティビティタスクを使用できるようにします。ディサイダーがアクティビティタスクをスケジュールするときに、アクティビティワーカーがアクティビティタスクを実行するために必要なデータ (ユーザーが決定) を提供します。Amazon SWF は、アクティビティワーカーに送信する前に、このデータをアクティビティタスクに挿入します。

アクティビティワーカーは、ユーザーによって管理されます。アクティビティワーカーは任意の言語で記述できます。ワーカーは、API を通じて Amazon SWF と通信できる限り、どこでも実行できます。Amazon SWF はアクティビティタスクを実行するために必要なすべての情報を提供するため、すべてのアクティビティワーカーはステートレスにすることができます。ステートレスであることにより、ワークフローは高度にスケーラブルになり、高いキャパシティー要件を処理して、より多くのアクティビティワーカーを追加できます。

このセクションでは、アクティビティワーカーを実装する方法を説明します。アクティビティワーカーは繰り返し次の操作を行う必要があります。

1. アクティビティタスクについて Amazon SWF をポーリングします。
2. タスクの実行を開始する。
3. タスクの存続期間が長い場合は、ハートビートを Amazon SWF に定期的に報告する。
4. タスクが完了または失敗したことを報告し、結果を Amazon SWF に返す。

トピック

アクティビティタスクのポーリング

アクティビティタスクを実行するには、各アクティビティワーカーは定期的に `PollForActivityTask` アクションを呼び出して Amazon SWF をポーリングする必要があります。

次の例では、アクティビティワーカー `ChargeCreditCardWorker01` は、タスクリスト `ChargeCreditCard-v0.1` でタスクをポーリングします。アクティビティタスクを使用できない場合、Amazon SWF は 60 秒後に空のレスポンスを送信します。空のレスポンスは、`taskToken` の値が空の文字列になっている `Task` 構造です。

```
https://swf.us-east-1.amazonaws.com
PollForActivityTask
{
```

```
  "domain" : "867530901",
  "taskList" : { "name": "ChargeCreditCard-v0.1" },
  "identity" : "ChargeCreditCardWorker01"
}
```

アクティビティタスクが使用可能になると、Amazon SWF はそれをアクティビティワーカーに返します。タスクには、アクティビティをスケジュールするときにディサイダーが指定するデータが含まれます。

アクティビティワーカーがアクティビティタスクを受け取ると、ワークを実行する準備が整います。次のセクションでは、アクティビティタスクの実行に関する情報を示します。

アクティビティタスクの実行

アクティビティワーカーがアクティビティタスクを受け取ると、そのタスクを実行する準備が整います。

アクティビティタスクを実行するには

1. タスクの入力フィールドでコンテンツの内容を解釈するようにアクティビティワーカーをプログラムします。このフィールドには、タスクがスケジュールされたときにディサイダーによって指定されたデータが含まれます。
2. アクティビティワーカーがデータの処理を処理し、ロジックを実行するようにプログラムします。

次のセクションでは、長時間実行されるアクティビティについて Amazon SWF にステータスの更新を提供するようにアクティビティワーカーをプログラムする方法について説明します。

アクティビティタスクのハートビートの報告

ハートビートタイムアウトがアクティビティタイプに登録された場合、アクティビティワーカーはハートビートタイムアウトを超える前にハートビートを記録する必要があります。アクティビティタスクがタイムアウト内にハートビートを提供しない場合、タスクはタイムアウトし、Amazon SWF がそれをクローズして新しい決定タスクをスケジュールします。それにより、タイムアウトについてディサイダーに通知します。次に、ディサイダーは、アクティビティタスクを再スケジュールするか、別のアクションを実行できます。

アクティビティワーカーがタイムアウト後に (RespondActivityTaskCompleted の呼び出しなどにより) Amazon SWF への接続を試みると、Amazon SWF は UnknownResource エラーを返します。

このセクションでは、アクティビティのハートビートを提供する方法を説明します。

アクティビティタスクのハートビートを記録するには、RecordActivityTaskHeartbeat アクションを呼び出すようにアクティビティワーカーをプログラムします。このアクションにより、アプリケーションにとって有効な方法で進捗状況を数量化するための自由形式のデータを保存するために使用できる文字列フィールドも提供されます。

この例では、アクティビティワーカーは Amazon SWF へのハートビートを報告し、詳細フィールドを使用して、アクティビティタスクが 40 パーセント完了したことを報告しています。ハートビートを報告するには、アクティビティワーカーはアクティビティタスクのタスクトークンを指定する必要があります。

```
https://swf.us-east-1.amazonaws.com
RecordActivityTaskHeartbeat
{
  "taskToken" : "12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "details" : "40"
}
```

このアクション自体では、ワークフロー実行履歴にイベントは作成されません。ただし、タスクがタイムアウトすると、ワークフロー実行履歴には、アクティビティワーカーによって生成された最後のハートビートからの情報が含まれた ActivityTaskTimedOut イベントが含まれます。

アクティビティタスクの完了または失敗

アクティビティワーカーは、タスクの実行後に、アクティビティタスクの完了または失敗を報告します。

アクティビティタスクの完了

アクティビティタスクを完了するには、アクティビティタスクを正常に完了した後で RespondActivityTaskCompleted アクションを呼び出し、タスクトークンを指定するようにアクティビティワーカーをプログラムします。

この例で、アクティビティワーカーは、タスクが正常に完了したことを示します。

```
https://swf.us-east-1.amazonaws.com
```

```
RespondActivityTaskCompleted
{
  "taskToken": "12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "results": "40"
}
```

アクティビティが完了すると、Amazon SWF はアクティビティが関連付けられたワークフロー実行の新しい決定タスクをスケジュールします。

対象のタスクの完了後に、別のアクティビティタスクをポーリングするようにアクティビティワーカーをプログラムします。これにより、アクティビティワーカーが継続的にタスクをポーリングして完了するループが作成されます。

アクティビティが StartToCloseTimeout 期間内に応答しないか、ScheduleToCloseTimeout を超えた場合、Amazon SWF はアクティビティタスクをタイムアウトし、決定タスクをスケジュールします。これにより、ディサイダーはタスクの再スケジュールなどの適切なアクションを実行できます。

例えば、Amazon EC2 インスタンスがアクティビティタスクを実行し、タスクが完了する前にインスタンスが失敗した場合、ディサイダーはワークフロー履歴でタイムアウトイベントを受け取ります。アクティビティタスクがハートビートを使用中の場合、Amazon EC2 インスタンスの失敗後、タスクが次のハートビートの提供に失敗すると、ディサイダーはイベントを受け取ります。そうでない場合、全体的なタイムアウト値の 1 つに一致する前にアクティビティタスクが完了に失敗すると、ディサイダーは最終的にイベントを受け取ります。次に、タスクを再割り当てするか、その他のアクションを実行するかをディサイダーが決定します。

アクティビティタスクの失敗

アクティビティワーカーが何らかの理由でタスクを実行できないが、Amazon SWF とまだ通信できる場合、タスクを失敗するようにプログラムできます。

アクティビティタスクを失敗するようにアクティビティワーカーをプログラムするには、タスクのタスクトークンを指定する RespondActivityTaskFailed アクションを呼び出します。

```
https://swf.us-east-1.amazonaws.com
RespondActivityTaskFailed
{
  "taskToken" : "12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "reason" : "CC-Invalid",
  "details" : "Credit Card Number Checksum Failed"
}
```

開発者として、理由および詳細フィールドに保存される値を定義します。これらは自由形式の文字列です。アプリケーションに対応する任意のエラーコード規則を使用できます。Amazon SWF はこれらの値を処理しません。ただし、Amazon SWF はこれらの値をコンソールに表示する場合があります。

アクティビティタスクが失敗すると、Amazon SWF はアクティビティタスクが関連付けられているワークフロー実行の決定タスクをスケジュールし、失敗のディサイダーに通知します。失敗の特性に応じて、アクティビティを再スケジュールする、ワークフロー実行を失敗するなどにより、失敗したアクティビティを処理するようにディサイダーをプログラムします。

アクティビティワーカーの起動

アクティビティワーカーを起動するには、アクティビティワーカープラットフォームで使用できる実行可能ファイルにロジックをパッケージ化します。たとえば、Linux サーバーと Windows サーバーの両方で実行できる Java 実行可能ファイルとしてアクティビティコードをパッケージ化します。

ワーカーは、起動するとタスクに対してポーリングを開始します。ただし、ディサイダーがアクティビティタスクをスケジュールするまで、このポーリングはタスクなしでタイムアウトし、ワーカーはポーリングを続行します。

ポーリングはアウトバウンドリクエストであるため、アクティビティワーカーは、Amazon SWF エンドポイントにアクセスできるすべてのネットワークで実行できます。

必要な数のアクティビティワーカーを起動できます。ディサイダーがアクティビティタスクをスケジューリングする際に、Amazon SWF は自動的にアクティビティタスクをポーリング中のアクティビティワーカーに分散します。

Amazon SWF でのディサイダーの開発

ディサイダーとは、ワークフロー実行中に実行されるワークフロータイプの調整ロジックの実装です。単一のワークフロータイプに複数のディサイダーを実行できます。

ワークフロー実行の実行状態はワークフロー履歴に保存されるため、ディサイダーはステートレスにできます。Amazon SWF はワークフロー実行履歴を維持し、それを各決定タスクを持つディサイダーに提供します。これにより、必要に応じて動的にディサイダーの追加や削除ができ、ワークフローの処理が大変スケーラブルになります。システム負荷の増加に合わせて、増加した容量を処理するためにディサイダーをさらに追加するだけです。ただし、特定のワークフロー実行に対して、いつでも決定タスクは 1 つしかオープンにできないことに注意してください。

ワークフロー実行に状態変更が発生するたびに、Amazon SWF は決定タスクをスケジュールします。ディサイダーは、決定タスクを受信するたびに以下を行います。

- 決定タスクを提供されたワークフロー実行履歴を解釈します。
- ワークフロー実行履歴に基づいて調整ロジックを適用し、次に実行すべきことを決定します。各決定は決定構造によって表されます。
- 決定タスクを完了し、Amazon SWF に決定のリストを提供します。

このセクションではディサイダーを開発する方法を説明しますが、それには以下の点が含まれます。

- 決定タスクをポーリングするためのディサイダーのプログラミング
- ワークフロー実行履歴を解釈し決定を行うためのディサイダーのプログラミング
- 決定タスクに応答するためのディサイダーのプログラミング

このセクションの例では、e コマースのサンプルワークフローにディサイダーをプログラムする方法を示します。

ディサイダーは、サービス API を通して Amazon SWF と通信できれば、任意の言語で実装でき、どこででも実行できます。

トピック

調整ロジックの定義

ディサイダーの開発時にまず行うのは、調整ロジックを定義することです。e コマースの例では、前のアクティビティの完了後に各アクティビティをスケジュールする調整ロジックは、以下のようになります。

```
IF lastEvent = "StartWorkflowInstance"
```

```
 addToDecisions ScheduleVerifyOrderActivity

ELSIF lastEvent = "CompleteVerifyOrderActivity"
 addToDecisions ScheduleChargeCreditCardActivity

ELSIF lastEvent = "CompleteChargeCreditCardActivity"
 addToDecisions ScheduleCompleteShipOrderActivity

ELSIF lastEvent = "CompleteShipOrderActivity"
 addToDecisions ScheduleRecordOrderCompletion

ELSIF lastEvent = "CompleteRecordOrderCompletion"
 addToDecisions CloseWorkflow

ENDIF
```

ディサイダーはワークフロー実行履歴に調整ロジックを適用し、RespondDecisionTaskCompleted アクションを使用して決定タスクを完了するときに決定のリストを作成します。

決定タスクのポーリング

各ディサイダーは、決定タスクをポーリングします。決定タスクには、アクティビティタスクのスケジューリングのような決定を生成するのにディサイダーが使用する情報が含まれます。決定タスクをポーリングするには、ディサイダーは PollForDecisionTask アクションを使用します。

この例では、ディサイダーは customerOrderWorkflow-0.1 TaskList を指定して決定タスクをポーリングします。

```
https://swf.us-east-1.amazonaws.com
PollForDecisionTask
{
  "domain": "867530901",
  "taskList": {"name": "customerOrderWorkflow-v0.1"},
  "identity": "Decider01",
  "maximumPageSize": 50,
  "reverseOrder": true
}
```

指定したタスクリストの決定タスクは、利用可能な場合は Amazon SWF により即時返されます。利用可能な決定タスクがない場合、Amazon SWF は接続を最大 60 秒間保持し、利用可能になり次第タスクを返します。タスクが利用可能にならない場合、Amazon SWF は空の応答を返します。空

の応答とは、taskToken の値が空の文字列になっている Task 構造です。空の応答を受信した場合は、必ず別のタスクをポーリングするようディサイダーをプログラムしてください。

決定タスクが利用可能な場合、Amazon SWF が返す応答には、その決定タスクとワークフロー実行履歴のページ分割された表示が含まれます。

この例では、最新のイベントのタイプがワークフロー実行の開始を示し、入力要素には最初のタスクを実行するのに必要な情報が含まれています。

```
{
  "events": [
    {
      "decisionTaskStartedEventAttributes": {
        "identity": "Decider01",
        "scheduledEventId": 2
      },
      "eventId": 3,
      "eventTimestamp": 1326593394.566,
      "eventType": "DecisionTaskStarted"
    }, {
      "decisionTaskScheduledEventAttributes": {
        "startToCloseTimeout": "600",
        "taskList": { "name": "specialTaskList" }
      },
      "eventId": 2,
      "eventTimestamp": 1326592619.474,
      "eventType": "DecisionTaskScheduled"
    }, {
      "eventId": 1,
      "eventTimestamp": 1326592619.474,
      "eventType": "WorkflowExecutionStarted",
      "workflowExecutionStartedEventAttributes": {
        "childPolicy" : "TERMINATE",
        "executionStartToCloseTimeout" : "3600",
        "input" : "data-used-decider-for-first-task",
        "parentInitiatedEventId": 0,
        "tagList" : ["music purchase", "digital", "ricoh-the-dog"],
        "taskList": { "name": "specialTaskList" },
        "taskStartToCloseTimeout": "600",
        "workflowType": {
          "name": "customerOrderWorkflow",
          "version": "1.0"
        }
```

```
      }
    }
  ],
  ...
}
```

ワークフロー実行履歴を受信したら、ディサイダーは履歴を解釈して調整ロジックに基づいて決定を行います。

単一のワークフロー実行のワークフロー履歴のイベント数が大きい場合があるので、返される結果は複数ページに分割される可能性があります。後続ページを取得するには、最初の呼び出しで返された nextPageToken を使用して PollForDecisionTask を追加で呼び出します。この nextPageToken を使用して GetWorkflowExecutionHistory を呼び出さないことに注意してください。代わりに、再度 PollForDecisionTask を呼び出します。

調整ロジックの適用

ディサイダーが決定タスクを受信したら、これまで何が起きたかを判断するためワークフロー実行履歴を解釈するようプログラムします。これに基づいて、決定のリストを生成する必要があります。

e コマースの例では、ワークフロー履歴の最後のイベントにのみ関心があるので、次のロジックを定義します。

```
IF lastEvent = "StartWorkflowInstance"
 addToDecisions ScheduleVerifyOrderActivity

ELSIF lastEvent = "CompleteVerifyOrderActivity"
 addToDecisions ScheduleChargeCreditCardActivity

ELSIF lastEvent = "CompleteChargeCreditCardActivity"
 addToDecisions ScheduleCompleteShipOrderActivity

ELSIF lastEvent = "CompleteShipOrderActivity"
 addToDecisions ScheduleRecordOrderCompletion

ELSIF lastEvent = "CompleteRecordOrderCompletion"
 addToDecisions CloseWorkflow

ENDIF
```

lastEvent が CompleteVerifyOrderActivity の場合、ScheduleChargeCreditCardActivity アクティビティを決定のリストに加えます。

ディサイダーは、行うべき決定を判断すると Amazon SWF に適切な決定で応答できます。

決定の応答

ワークフロー履歴を解釈して決定のリストを生成したら、ディサイダーはそれらの決定を Amazon SWF に返すことができる状態にあります。

ワークフロー実行履歴から必要なデータを抽出し、次にワークフローの次の適切なアクションを指定する決定を作成するようディサイダーをプログラムします。ディサイダーはこれらの決定を RespondDecisionTaskCompleted アクションを使用して Amazon SWF に返します。利用可能な決定タイプのリストについては、「Amazon Simple Workflow Service API Reference」(Amazon Simple Workflow Service API リファレンス) を参照してください。

e コマースの例では、ディサイダーが生成した一連の決定を返すときに、ワークフロー実行履歴にあるクレジットカード入力も含めます。その結果アクティビティワーカーは、アクティビティタスクを実行するのに必要なすべての情報を持つことになります。

ワークフロー実行のすべてのアクティビティが完了すると、ディサイダーはワークフロー実行をクローズします。

```
https://swf.us-east-1.amazonaws.com
RespondDecisionTaskCompleted
{
  "taskToken" : "12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "decisions" : [
    {
      "decisionType" :"ScheduleActivityTask",
      "scheduleActivityTaskDecisionAttributes" : {
        "control" :"OPTIONAL_DATA_FOR_DECIDER",
        "activityType" : {
          "name" :"ScheduleChargeCreditCardActivity",
          "version" :"1.1"
        },
        "activityId" :"3e2e6e55-e7c4-beef-feed-aa815722b7be",
        "scheduleToCloseTimeout" :"360",
        "taskList" : { "name" :"CC_TASKS" },
        "scheduleToStartTimeout" :"60",
        "startToCloseTimeout" :"300",
        "heartbeatTimeout" :"60",
        "input" : "4321-0001-0002-1234: 0212 : 234"
      }
    }
```

```
  ]
}
```

ワークフロー実行のクローズ

ビジネスプロセスが完了している、つまりこれ以上実行するアクティビティがないと判断すると、ディサイダーはワークフロー実行をクローズする決定を生成します。

ワークフロー実行をクローズするには、ワークフロー履歴のイベントを解釈してこれまで何が起きたかを判断し、ワークフロー実行をクローズすべきか確認するようディサイダーをプログラムします。

ワークフローが正常に完了した場合、CompleteWorkflowExecution 決定で RespondDecisionTaskCompleted を呼び出すことでワークフロー実行をクローズします。または、FailWorkflowExecution 決定を使用して誤った実行を失敗させることができます。

e コマースの例では、ディサイダーは履歴を確認し、調整ロジックに基づいてワークフロー実行をクローズする決定を決定のリストに追加して、クローズワークフロー決定で RespondDecisionTaskCompleted アクションを開始します。

> **Note**
>
> ワークフロー実行のクローズが失敗するケースがあります。たとえば、ディサイダーがワークフロー実行をクローズしている間にシグナルを受信すると、そのクローズ決定は失敗します。この可能性に対処するには、ディサイダーが確実に決定タスクをポーリングし続けるようにします。また、次の決定タスクを受信するディサイダーが、実行のクローズを妨げたイベント (このケースではシグナル) に確実に対応するようにします。

ワークフロー実行のキャンセルをサポートする可能性もあります。これは、特に実行時間が長いワークフローの場合に便利なことがあります。キャンセルをサポートするには、ディサイダーは履歴の WorkflowExecutionCancelRequested イベントを処理する必要があります。このイベントは、実行のキャンセルがリクエストされたことを示します。ディサイダーは、たとえば実行中のアクティビティタスクをキャンセルしたり CancelWorkflowExecution 決定で RespondDecisionTaskCompleted アクションを呼び出してワークフローをクローズするといった、適切なクリーンアップアクションを実行する必要があります。

以下の例では、RespondDecisionTaskCompleted を呼び出して現在のワークフロー実行をキャンセルするよう指定します。

```
https://swf.us-east-1.amazonaws.com
RespondDecisionTaskCompleted
{
  "taskToken" : "12342e17-80f6-FAKE-TASK-TOKEN32f0223",
  "decisions" : [
    {
      "decisionType":"CancelWorkflowExecution",
      "CancelWorkflowExecutionAttributes":{
        "Details": "Customer canceled order"
      }
    }
  ]
}
```

Amazon SWF は、ワークフロー実行のクローズやキャンセルの決定が確実にディサイダーが送信する最後の決定であるかを確認します。つまり、ワークフローをクローズする決定の後にも決定がある一連の決定を持つことは無効です。

ディサイダーの起動

ディサイダー開発が完了したら、1 つ以上のディサイダーを起動することができます。

ディサイダーを起動するには、ディサイダープラットフォームで使用できる実行可能ファイルに調整ロジックをパッケージ化します。たとえば、Linux コンピュータと Windows コンピュータの両方で実行できる Java 実行可能ファイルとしてディサイダーコードをパッケージ化します。

起動すると、ディサイダーはタスクを求めて Amazon SWF のポーリングを開始するはずです。ワークフロー実行が開始され Amazon SWF が決定タスクをスケジュールするまで、これらのポーリングはタイムアウトして空の応答を取得します。空の応答とは、taskToken の値が空の文字列になっている Task 構造です。ディサイダーは単にポーリングし続けます。

Amazon SWF は、いつでもワークフロー実行に対して 1 つの決定タスクのみがアクティブになれるようにします。これにより、競合する決定といった問題を回避できます。さらに、Amazon SWF は実行中のディサイダーの数に関係なく単一の決定タスクが単一のディサイダーに割り当てられるようにします。

ディサイダーが別の決定タスクを処理中に決定タスクを生成する何かが発生すると、Amazon SWF は現在のタスクが完了するまでその新しいタスクをキューに入れます。現在のタスクが完了した後、Amazon SWF は新しい決定タスクを使用可能にします。また、ディサイダーが決定タスクを処理している間に複数のアクティビティーが完了した場合、Amazon SWF は複数のタスク完了を考

慮して 1 つの新しい決定タスクしか作成しないという意味で、決定タスクを一括処理します。ただし、各タスク完了は、ワークフロー実行履歴で個々のイベントを受け取ります。

ポーリングはアウトバウンドリクエストであるため、ディサイダーは、Amazon SWF エンドポイントにアクセスできるすべてのネットワークで実行できます。

ワークフロー実行が進行するには、1 つ以上のディサイダーが実行中である必要があります。ディサイダーは必要な数だけ起動できます。Amazon SWF は同じタスクリストでポーリングする複数のディサイダーをサポートします。

Amazon SWF でのワークフローの開始

`StartWorkflowExecution` アクションを使用して、任意のアプリケーションから登録されたワークフロータイプのワークフロー実行を開始できます。開始時に、`workflowId` と呼ばれる ID を実行に関連付けます。`workflowId` は、注文処理アプリケーションの注文番号のような、アプリケーションに対して適切な任意の文字列とすることができます。同じドメイン内の複数のオープンなワークフロー実行に対して同じ `workflowId` を使用することはできません。たとえば、`workflowId` `Customer Order 01` でワークフロー実行を開始した場合、2 番目のワークフロー実行は開始されず、リクエストは失敗します。ただし、閉じた実行の `workflowId` を再利用することはできます。Amazon SWF は、`runId` と呼ばれる、システムで生成された一意の識別子をワークフローの実行ごとに関連付けます。

ワークフローおよびアクティビティのタイプが登録されたら、`StartWorkflowExecution` アクションを呼び出してワークフローを起動します。`input` パラメータの値は、ワークフローを開始するアプリケーションによって指定される任意の文字列とすることができます。`executionStartToCloseTimeout` は、ワークフロー実行の開始からクローズまで許可される時間 (秒単位) です。この制限を超えると、ワークフロー実行がタイムアウトします。Amazon SWF の他のタイムアウトパラメータの一部とは異なり、このタイムアウトに値 `NONE` を指定することはできません。ワークフロー実行の時間には、最大 1 年間という制限があります。同様に、taskStartToCloseTimeout は、このワークフロー実行に関連付けられた決定タスクがタイムアウトするまで許可される時間 (秒数) です。

```
https://swf.us-east-1.amazonaws.com
StartWorkflowExecution
{
  "domain" : "867530901",
  "workflowId" : "20110927-T-1",
  "workflowType" : {
    "name" : "customerOrderWorkflow", "version" : "1.1"
```

```
  },
  "taskList" : { "name" : "specialTaskList" },
  "input" : "arbitrary-string-that-is-meaningful-to-the-workflow",
  "executionStartToCloseTimeout" : "1800",
  "tagList" : [ "music purchase", "digital", "ricoh-the-dog" ],
  "taskStartToCloseTimeout" : "1800",
  "childPolicy" : "TERMINATE"
}
```

StartWorkflowExecution アクションが成功すると、Amazon SWF はワークフロー実行に対して runId を返します。ワークフロー実行の runId は、特定のリージョン内で一意です。後で、Amazon SWF への呼び出しでこのワークフロー実行を指定する必要がある場合は、runId を保存します。たとえば、後でワークフロー実行にシグナルを送信する必要がある場合は、runId を使用します。

```
{"runId": "9ba33198-4b18-4792-9c15-7181fb3a8852"}
```

Amazon SWF でのタスク優先度の設定

デフォルトでは、タスクリストのタスクは到着時間に基づいて提供されます。つまり、最初にスケジュールされたタスクは、通常は可能な限り最初に実行されます。オプションのタスクの優先順位を設定することで、特定のタスクに優先順位を与えることができます。Amazon SWF は、タスクリストで優先順位が低いものよりも先に、優先順位の高いタスクの提供を試みます。

> **Note**
>
> 通常、最初にスケジュールされるタスクは最初に実行されますが、これは保証されません。

ワークフローとアクティビティの両方のタスクの優先順位を設定できます。ワークフローのタスクの優先順位は、スケジュールされるいずれのアクティビティタスクの優先順位にも影響しません。また、起動されるいずれの子ワークフローにも影響しません。アクティビティまたはワークフローのデフォルトの優先順位は、登録中に (ユーザーまたは Amazon SWF によって) 設定され、アクティビティのスケジュール中またはワークフロー実行の開始中にオーバーライドされない限り、登録されたタスクの優先順位が常に使用されます。

タスクの優先順位の値は、"-2147483648" から "2147483647" の範囲であり、値が高いほど優先順位も高くなります。アクティビティまたはワークフローのタスク優先順位を設定しない場合、優先順位としてゼロ ("0") が割り当てられます。

トピック

ワークフローのタスクの優先順位の設定

登録または起動時に、ワークフローのタスクの優先順位を設定できます。ワークフロータイプの登録時に設定されるタスクの優先順位は、ワークフロー実行の開始時にオーバーライドされない限り、その種類の任意のワークフロー実行のデフォルトとして使用されます。

ワークフロータイプをデフォルトのタスク優先度で登録するには、RegisterWorkflowType アクションを使用するときに defaultTaskPriority オプションを設定します。

```
{
  "domain": "867530901",
  "name": "expeditedOrderWorkflow",
  "version": "1.0",
  "description": "Expedited customer orders workflow",
  "defaultTaskStartToCloseTimeout": "600",
  "defaultExecutionStartToCloseTimeout": "3600",
  "defaultTaskList": {"name": "mainTaskList"},
  "defaultTaskPriority": "10",
  "defaultChildPolicy": "TERMINATE"
}
```

StartWorkflowExecution でワークフロー実行を開始するときに、ワークフロータイプの登録されたタスクの優先順位をオーバーライドできます。

```
{
  "childPolicy": "TERMINATE",
  "domain": "867530901",
  "executionStartToCloseTimeout": "1800",
  "input": "arbitrary-string-that-is-meaningful-to-the-workflow",
  "tagList": ["music purchase", "digital", "ricoh-the-dog"],
  "taskList": {"name": "specialTaskList"},
  "taskPriority": "-20",
  "taskStartToCloseTimeout": "600",
  "workflowId": "20110927-T-1",
```

```
  "workflowType": {"name": "customerOrderWorkflow", "version": "1.0"},
}
```

子ワークフローを開始するときや、RespondDecisionTaskCompleted で決定に応答するときなど、新規にワークフローを続行するときにも、登録されたタスクの優先順位をオーバーライドできます。

子ワークフローのタスクの優先順位を設定するには、startChildWorkflowExecutionDecisionAttributes に値を指定します。

```
{
  "taskToken": "AAAAKgAAAAEAAAAAAAAAA...",
  "decisions": [
    {
      "decisionType": "StartChildWorkflowExecution",
      "startChildWorkflowExecutionDecisionAttributes": {
        "childPolicy": "TERMINATE",
        "control": "digital music",
        "executionStartToCloseTimeout": "900",
        "input": "201412-Smith-011x",
        "taskList": {"name": "specialTaskList"},
        "taskPriority": "5",
        "taskStartToCloseTimeout": "600",
        "workflowId": "verification-workflow",
        "workflowType": {
          "name": "MyChildWorkflow",
          "version": "1.0"
        }
      }
    }
  ]
}
```

新しくワークフローを続行する場合は、continueAsNewWorkflowExecutionDecisionAttributes でタスクの優先順位を設定します。

```
{
  "taskToken": "AAAAKgAAAAEAAAAAAAAAA...",
  "decisions": [
    {
      "decisionType": "ContinueAsNewWorkflowExecution",
      "continueAsNewWorkflowExecutionDecisionAttributes": {
```

```
        "childPolicy": "TERMINATE",
        "executionStartToCloseTimeout": "1800",
        "input": "5634-0056-4367-0923,12/12,437",
        "taskList": {"name": "specialTaskList"},
        "taskStartToCloseTimeout": "600",
        "taskPriority": "100",
        "workflowTypeVersion": "1.0"
      }
    }
  ]
}
```

アクティビティのタスクの優先順位の設定

登録時またはスケジュール時に、アクティビティのタスクの優先順位を設定できます。アクティビティタイプを登録するときに設定されるタスクの優先順位は、アクティビティのスケジュール時にオーバーライドされない限り、アクティビティ実行時のデフォルトの優先順位として使用されます。

アクティビティタイプを登録するときにタスクの優先度を設定するには、RegisterActivityType アクションを使用するときに defaultTaskPriority オプションを設定します。

```
{
  "defaultTaskHeartbeatTimeout": "120",
  "defaultTaskList": {"name": "mainTaskList"},
  "defaultTaskPriority": "10",
  "defaultTaskScheduleToCloseTimeout": "900",
  "defaultTaskScheduleToStartTimeout": "300",
  "defaultTaskStartToCloseTimeout": "600",
  "description": "Verify the customer credit card",
  "domain": "867530901",
  "name": "activityVerify",
  "version": "1.0"
}
```

タスク優先度でタスクをスケジュールするには、RespondDecisionTaskCompleted アクションでアクティビティをスケジュールするときに taskPriority オプションを使用します。

```
{
  "taskToken": "AAAAKgAAAAEAAAAAAAAAA...",
  "decisions": [
    {
```

```
      "decisionType": "ScheduleActivityTask",
      "scheduleActivityTaskDecisionAttributes": {
        "activityId": "verify-account",
        "activityType": {
            "name": "activityVerify",
            "version": "1.0"
        },
        "control": "digital music",
        "input": "abab-101",
        "taskList": {"name": "mainTaskList"},
        "taskPriority": "15"
      }
    }
  ]
}
```

タスクの優先順位情報を返すアクション

次の Amazon SWF アクションから、設定されたタスクの優先順位 (または、設定されたデフォルトのタスクの優先順位) に関する情報を取得できます。

- DescribeActivityType は、レスポンスの `configuration` セクションでアクティビティタイプの defaultTaskPriority を返します。
- DescribeWorkflowExecution は、レスポンスの `executionConfiguration` セクションでワークフロー実行の taskPriority を返します。
- DescribeWorkflowType は、レスポンスの `configuration` セクションでワークフロータイプの defaultTaskPriority を返します。
- GetWorkflowExecutionHistory と PollForDecisionTask は、レスポンスの `activityTaskScheduledEventAttributes`、`decisionTaskScheduledEventAttributes`、`wor` セクションでタスクの優先順位情報を提供します。

Amazon SWF でのエラーの処理

ワークフローの実行中に発生する可能性のあるさまざまな種類のエラーがあります。

トピック

- 検証エラー
- アクションまたは決定を実行する際のエラー

- タイムアウト
- ユーザーコードによって発生したエラー
- ワークフロー実行の終了に関連するエラー

検証エラー

検証エラーは、正しく構成されていないか、無効なデータが含まれているため、Amazon SWF へのリクエストが失敗した場合に発生します。このコンテキストでは、リクエストは `DescribeDomain` などのアクション、または `StartTimer` などの決定であることもあります。リクエストがアクションの場合、Amazon SWF はレスポンスにエラーコードを返します。このエラーコードをチェックすると、リクエストのどの部分がエラーを引き起こしたかに関する情報が提供される可能性があります。たとえば、リクエストで渡された 1 つ以上の引数が無効である可能性があります。一般的なエラーコードのリストについては、「Amazon Simple Workflow Service API Reference」(Amazon Simple Workflow Service API リファレンス) のアクションのトピックにアクセスしてください。

失敗したリクエストが決定の場合は、適切なイベントがワークフロー実行履歴に表示されます。たとえば、`StartTimer` 決定が失敗した場合は、履歴に `StartTimerFailed` イベントが表示されます。ディサイダーは、`PollForDecisionTask` または `GetWorkflowExecutionHistory` の応答として履歴を受け取るときに、これらのイベントをチェックする必要があります。以下は、決定の形式が正しくないか、無効なデータが含まれている場合に発生する可能性のある決定失敗イベントのリストです。

アクションまたは決定を実行する際のエラー

リクエストが正しい形式でも、Amazon SWF がリクエストを実行しようとするとエラーが発生することがあります。このような場合、履歴内の次のいずれかのイベントが、エラーが発生したことを示します。イベントの `reason` フィールドを調べて、障害の原因を特定します。

- `CancelTimerFailed`
- `RequestCancelActivityTaskFailed`
- `RequestCancelExternalWorkflowExecutionFailed`
- `ScheduleActivityTaskFailed`
- `SignalExternalWorkflowExecutionFailed`
- `StartChildWorkflowExecutionFailed`
- `StartTimerFailed`

タイムアウト

ディサイダー、アクティビティワーカー、およびワークフロー実行は、すべてタイムアウト期間の制約内で動作します。このタイプのエラーでは、タスクまたは子ワークフローがタイムアウトします。タイムアウトを説明するイベントが履歴に表示されます。ディサイダーは、たとえば、タスクの再スケジュールや子ワークフローの再開などによってこのイベントを処理する必要があります。タイムアウトの詳細については、Amazon SWF タイムアウトの種類 を参照してください。

- `ActivityTaskTimedOut`
- `ChildWorkflowExecutionTimedOut`
- `DecisionTaskTimedOut`
- `WorkflowExecutionTimedOut`

ユーザーコードによって発生したエラー

このタイプのエラー状態の例は、アクティビティタスクの失敗と子ワークフローの失敗です。タイムアウトエラーと同様に、Amazon SWF はワークフロー実行履歴に適切なイベントを追加します。ディサイダーは、タスクの再スケジュールや子ワークフローの再開などによってこのイベントを処理する必要があります。

- `ActivityTaskFailed`
- `ChildWorkflowExecutionFailed`

ワークフロー実行の終了に関連するエラー

ディサイダーが保留中の決定タスクのあるワークフローを終了しようとすると、以下のイベントが表示されることがあります。

- `FailWorkflowExecutionFailed`
- `CompleteWorkFlowExecutionFailed`
- `ContinueAsNewWorkflowExecutionFailed`
- `CancelWorkflowExecutionFailed`

上記のイベントの詳細については、「Amazon SWF API Reference」(Amazon SWF API リファレンス) の「History Event」(履歴イベント) を参照してください。

Amazon SWF クォータ

Amazon SWF では、アカウントごとのドメイン数やワークフロー実行履歴のサイズなど、特定のワークフローパラメータのサイズにクォータを設定します。これらのクォータは、誤ったワークフローがシステムのすべてのリソースを消費するの防ぐために設計されていますが、ハードの制限ではありません。アプリケーションがこれらのクォータを頻繁に超えている場合、サービスクォータの増加を申請できます。

内容

Amazon SWF の全般アカウントクォータ

- 最大登録ドメイン数 - 100

  このクォータには登録済みドメインと廃止ドメインの両方が含まれています。
- ワークフローとアクティビティタイプの最大数 – ドメインごとに 10,000

  このクォータには登録済みドメインと廃止タイプの両方が含まれています。
- API コールのクォータ – 頻繁でないスパイクを超えてアプリケーションが非常に短期間に大量の API コールを実行した場合、アプリケーションは調整されることがあります。
- 最大のリクエストサイズ – リクエストあたり 1 MB

  これは Amazon SWF API リクエストあたりの合計データサイズで、リクエストヘッダーおよびその他すべての関連するリクエストデータを含みます。
- Count API の切り捨てられたレスポンス - 内部クォータに達したこと、およびレスポンスが総数ではないことを示します。

一部のクエリでは、完全なレスポンスを返す前に、内部的に 1 MB のクォータに達します。以下では、総数ではなく切り捨てられたレスポンスが返される場合があります。

- CountClosedWorkflowExecutions
- CountOpenWorkflowExecutions
- CountPendingActivityTasks
- CountPendingDecisionTasks

これらのそれぞれに対して、`truncated` レスポンスが true に設定されている場合、数は総数より小さくなります。この内部クォータを引き上げることはできません。

- タグの最大数 - リソースあたり 50 個。

  50 個を超えるタグを追加しようとすると、400 エラー `TooManyTagsFault` が発生します。

ワークフロー実行のクォータ

- オープン状態のワークフロー実行の最大数 - ドメインあたり 100,000

  この数には、子ワークフローの実行が含まれます。
- 最大ワークフロー実行時間 - 1 年 これはハードクォータであり、変更できません。
- 最大ワークフロー実行履歴サイズ - 25,000 イベント これはハードクォータであり、変更できません。

  ベストプラクティスは、各ワークフローの履歴が 10,000 イベントを超えないように構成することです。ディサイダーはワークフロー履歴を取得する必要があるため、履歴を小さくすると、ディサイダーはより迅速に完了できます。Flow Framework を使用している場合、ContinueAsNew を使用して新しい履歴でワークフローを継続して使用できます。
- 子ワークフロー実行の最大数 - ワークフロー実行あたり 1,000

ユースケースでこれらのクォータを超える必要がある場合は、Amazon SWF の機能を使用して実行を継続し、子ワークフロー実行を使用してアプリケーションを構成します。さらにクォータを増やす必要がある場合は、「クォータ引き上げのリクエスト」を参照してください。

タスク実行におけるクォータ

- タスクリストあたりの最大ポーラー数 - タスクリストあたり 1,000

特定のタスクリストに対して同時にポーリングを実行するポーラーを、最大 1,000 個持つことができます。1,000 個を超えた場合は、`LimitExceededException` が発生します。

> **Note**
>
> 最大値は 1,000 個ですが、このクォータの前に `LimitExceededException` エラーが発生する場合があります。このエラーは、タスクが遅れていることを意味するものではありません。そうではなく、1 つのタスクリストでアイドル状態のポーラーの数が最大数に達していることを意味します。Amazon SWF は、クライアント側とサーバー側の両方でリソースを節約するためにこの制限を設定しています。制限を設定することで、過剰な数のポーラーが不必要に待機することを防ぎます。複数のタスクリストを使用してポーリングを分散することで、`LimitExceededException` エラーを減らすことができます。

- 1 秒あたりにスケジュールされるタスクの最大数 — タスクリストあたり 2,000 個

  特定のタスクリストに対して、1 秒あたり最大 2,000 個のタスクをスケジュールできます。2,000 個を超えると、`ScheduleActivityTask` 決定が `ACTIVITY_CREATION_RATE_EXCEEDED` エラーで失敗します。

  > **Note**
  >
  > 最大値は 2,000 個ですが、このクォータの前に `ACTIVITY_CREATION_RATE_EXCEEDED` エラーが発生する場合があります。これらのエラーを減らすには、複数のタスクリストを使用して負荷を分散します。

- 最大タスク実行時間 - 1 年 (ワークフロー実行時間の最大値による制約)

  アクティビティのタイムアウトを設定して、アクティビティタスク実行の特定の段階で大幅に時間がかかる場合にタイムアウトイベントを発生できます。

- SWF がタスクをキューに保持する最大時間 – 1 年 (ワークフロー実行時間のクォータにより制約されます)

  アクティビティ登録中にデフォルトのアクティビティのタイムアウトを設定すると、アクティビティタスク実行の特定の段階で大幅に時間がかかる場合にタイムアウトイベントが発生します。ディサイダーコードでアクティビティタスクをスケジュールするときに、デフォルトのアクティビティタイムアウトを上書きすることもできます。

- 最大オープンアクティビティタスク - ワークフロー実行あたり 1,000

このクォータには、ワーカーによってスケジュールされたアクティビティタスクと処理中のアクティビティタスクの両方が含まれます。

- 最大オープンタイマー - ワークフロー実行あたり 1,000
- 最大入力/結果データサイズ - 32,768 文字

  このクォータは、アクティビティまたはワークフロー実行結果データ、アクティビティタスクまたはワークフロー実行をスケジューリングする時の入力データ、およびワークフロー実行シグナルと共に送信される入力に影響します。
- 決定タスクの応答の最大決定 - それぞれで異なる

  最大 API リクエストのサイズに 1 MB のクォータがあるため、`RespondDecisionTaskCompleted` への単一の呼び出しで返される決定の数は、スケジューリングされたアクティビティタスクまたはワークフロー実行に提供される入力データのサイズを含む、各決定で使用されるデータのサイズに応じて制限されます。

Amazon SWF スロットリングのクォータ

前に説明したサービスのクォータに加え、特定の Amazon SWF API コールおよび決定イベントは、トークンバケットスキームを使用してサービス帯域幅を維持するために調整されます。リクエストのレートがここに記載されているレートを継続的に超えている場合、スロットルクォータの増加をリクエストできます。

スロットリングクォータと決定クォータはすべてのリージョンで同じです。

すべてのリージョンのスロットリングクォータ

以下のクォータは個々のアカウントレベルで適用されます。次のクォータの引き上げをリクエストすることもできます。これを行う方法については、「クォータ引き上げのリクエスト」を参照してください。

API 名	[バケットサイズ]	補充レート/秒
`CountClosedWorkflowExecutions`	2000	6
`CountOpenWorkflowExecutions`	2000	6
`CountPendingActivityTasks`	200	6

API 名	[バケットサイズ]	補充レート/秒
CountPendingDecisionTasks	200	6
DeleteActivityType	200	6
DeleteWorkflowType	200	6
DeprecateActivityType	200	6
DeprecateDomain	100	6
DeprecateWorkflowType	200	6
DescribeActivityType	2000	6
DescribeDomain	200	6
DescribeWorkflowExecution	2000	6
DescribeWorkflowType	2000	6
GetWorkflowExecutionHistory	2000	60
ListActivityTypes	200	6
ListClosedWorkflowExecutions	200	6
ListDomains	100	6
ListOpenWorkflowExecutions	200	48
ListTagsForResource	50	30
ListWorkflowTypes	200	6
PollForActivityTask	2000	200
PollForDecisionTask	2000	200
RecordActivityTaskHeartbeat	2000	160

API 名	[バケットサイズ]	補充レート/秒
RegisterActivityType	200	60
RegisterDomain	100	6
RegisterWorkflowType	200	60
RequestCancelWorkflowExecution	2000	30
RespondActivityTaskCanceled	2000	200
RespondActivityTaskCompleted	2000	200
RespondActivityTaskFailed	2000	200
RespondDecisionTaskCompleted	2000	200
SignalWorkflowExecution	2000	30
StartWorkflowExecution	2000	200
TagResource	50	30
TerminateWorkflowExecution	2000	60
UndeprecateActivityType	200	6
UndeprecateDomain	100	6
UndeprecateWorkflowType	200	6
UntagResource	50	30

すべてのリージョンの決定クォータ

以下のクォータは個々のアカウントレベルで適用されます。次のクォータの引き上げをリクエストすることもできます。これを行う方法については、「クォータ引き上げのリクエスト」を参照してください。

API 名	[バケットサイズ]	補充レート/秒
RequestCancelExternalWorkflowExecution	1200	120
ScheduleActivityTask	1,000	200
SignalExternalWorkflowExecution	1200	120
StartChildWorkflowExecution	500	12
StartTimer	2000	200

ワークフローレベルのクォータ

以下のクォータはワークフローレベルで適用され、増やすことはできません。

API 名	[バケットサイズ]	補充レート/秒
GetWorkflowExecutionHistory	400	200
SignalWorkflowExecution	1,000	1,000
RecordActivityTaskHeartbeat	1,000	1,000
RequestCancelWorkflowExecution	200	200

クォータ引き上げのリクエスト

詳細については、「AWS 全般のリファレンス」の「AWS の Service Quotas」を参照してください。

Amazon SWF の追加リソースとリファレンス情報

この章では、Amazon SWF を使用してワークフローを開発する場合に有効なその他のリソースおよびリファレンス情報を提供します。

トピック

Amazon SWF タイムアウトの種類

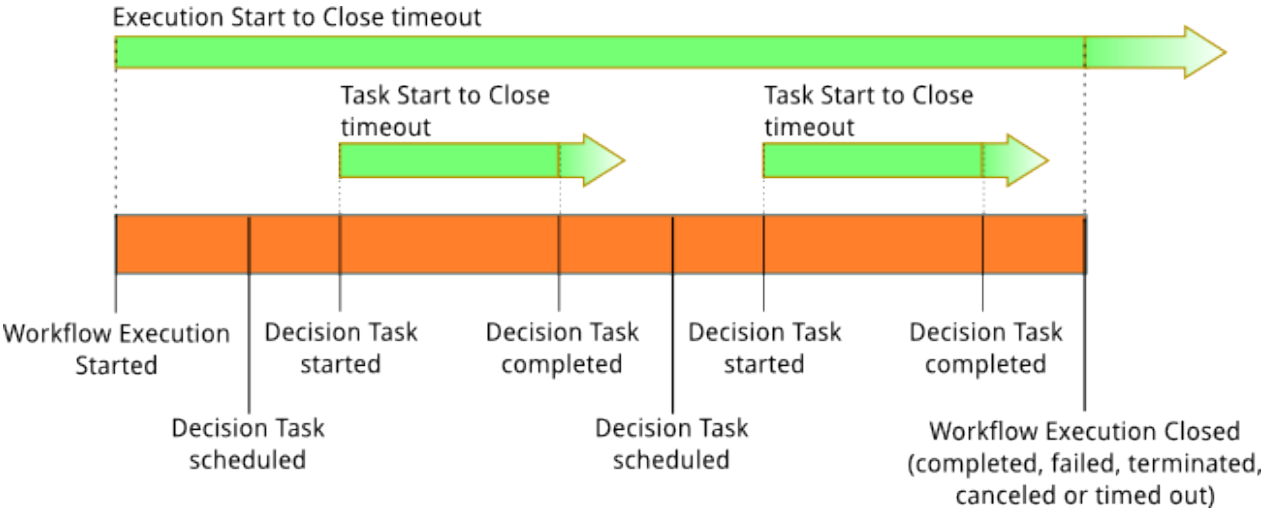
ワークフロー実行が正しく実行されるように、Amazon SWF でさまざまなタイプのタイムアウトを設定できます。一部のタイムアウトでは、ワークフローが完全に実行できる時間を指定します。その他のタイムアウトでは、ワーカーに割り当てる前にアクティビティタスクを実行できる時間と、スケジュールされてから完了までにかけることができる時間を指定します。Amazon SWF API のすべてのタイムアウトは秒単位で指定されます。Amazon SWF は、タイムアウト値として文字列 `NONE` もサポートしています。これは、タイムアウトがないことを示します。

決定タスクとアクティビティタスクに関連するタイムアウトの場合、Amazon SWF はワークフロー実行履歴にイベントを追加します。イベントの属性により、発生したタイムアウトの種類と、影響を受けた決定タスクまたはアクティビティタスクに関する情報が提供されます。Amazon SWF は決定タスクもスケジュールします。ディサイダーは、新しい決定タスクを受け取ると、履歴でタイムアウトイベントを確認し、RespondDecisionTaskCompleted アクションを呼び出して適切なアクションを実行します。

タスクは、スケジュールされてからクローズされるまではオープン状態と見なされます。したがって、ワーカーの処理中はタスクはオープン状態と報告されます。タスクは、ワーカーによって完了済み、キャンセル済み、または失敗と報告されるとクローズされます。タスクは、タイムアウトの結果として Amazon SWF によってクローズされる場合もあります。

ワークフローと決定タスクのタイムアウト

次の図は、ワークフローと決定のタイムアウトがワークフローの有効期間にどのように関係するかを示しています。



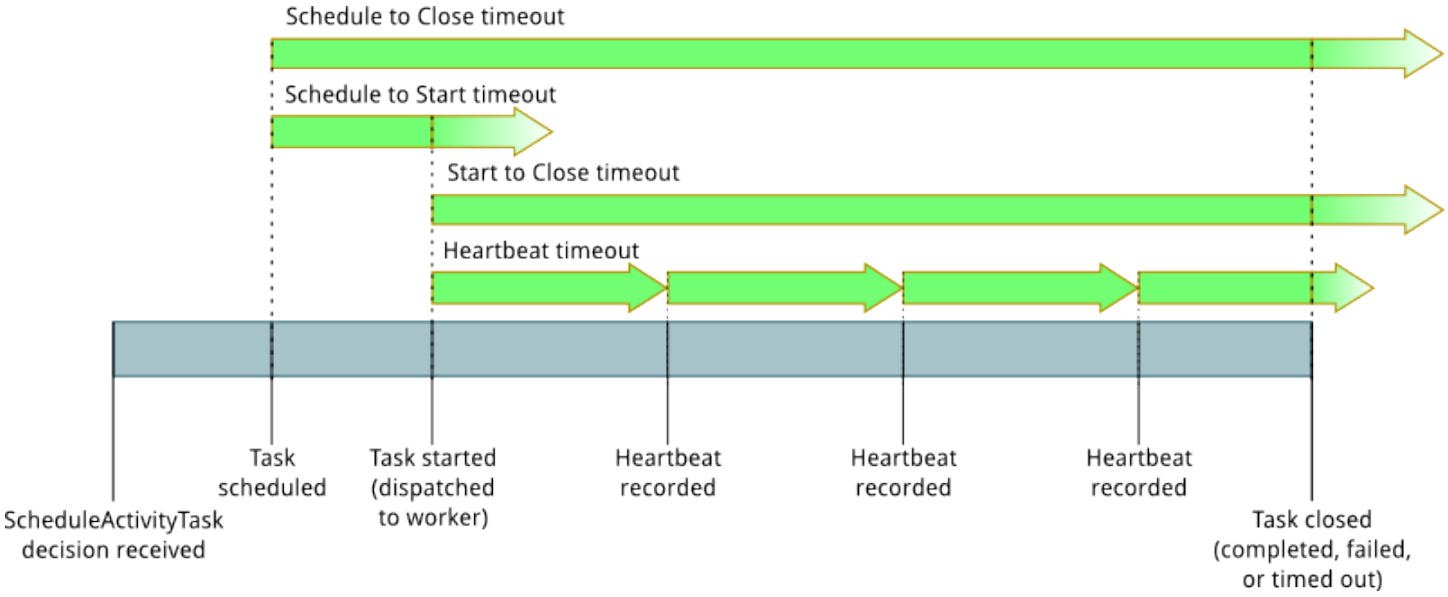
ワークフローと決定タスクに関連して 2 つのタイムアウトの種類があります。

- ワークフローのスタートからクローズ (**timeoutType: START_TO_CLOSE**) - タイムアウトは、ワークフロー実行の完了までにかかる最大時間を指定します。これはワークフローの登録中にデフォルトとして設定されますが、ワークフローのスタート時に別の値でオーバーライドできます。このタイムアウトを超えると、Amazon SWF はワークフローの実行を終了し、WorkflowExecutionTimedOut タイプのイベントをワークフローの実行履歴に追加します。イベント属性は、`timeoutType` に加えて、このワークフロー実行に対して有効である `childPolicy` を指定します。子ポリシーでは、親ワークフロー実行の回数がタイムアウトするか、それ以外に終了した場合に、子ワークフロー実行を処理する方法を指定します。たとえば、`childPolicy` が TERMINATE に設定された場合、子ワークフロー実行は終了します。ワークフロー実行がタイムアウトすると、表示の呼び出し以外にこれに対してアクションを実行することはできません。
- 決定タスクのスタートからクローズ (**timeoutType: START_TO_CLOSE**) - このタイムアウトにより、対応するディサイダーが決定タスクを完了するまでにかかる最大時間を指定します。これはワークフロータイプの登録中に設定されます。このタイムアウトを超えると、タスクはワークフロー実行履歴でタイムアウトとマークされ、Amazon SWF は DecisionTaskTimedOut 型のイベントをワークフロー履歴に追加します。イベントの属性には、この決定タスクがスケジュールされた日時 (`scheduledEventId`) およびスタートされた日時 (`startedEventId`) に対応するイベントの ID が含まれます。イベントの追加に加えて、Amazon SWF はこの決定タスクがタイムアウトしたことをディサイダーにアラートする新しい決定タスクをスケジュールします。このタイムアウト

の発生後は、RespondDecisionTaskCompleted を使用してタイムアウトした決定タスクを完了する試みは失敗します。

アクティビティタスクのタイムアウト

次の図は、タイムアウトがアクティビティタスクの有効期間にどのように関係するかを示します。



アクティビティタスクに関連して 4 つのタイムアウトの種類があります。

- アクティビティタスクのスタートからクローズ (**timeoutType: START_TO_CLOSE**) – このタイムアウトは、ワーカーがタスクを受け取った後でアクティビティワーカーがタスクを処理するためにかかる最大時間を指定します。RespondActivityTaskCanceled、RespondActivityTaskCompleted、および RespondActivityTaskFailed を使用してタイムアウトしたアクティビティタスクを閉じる試みは失敗します。
- アクティビティタスクのハートビート (**timeoutType: HEARTBEAT**) – このタイムアウトは、RecordActivityTaskHeartbeat アクションを通じて進捗状況を提供する前にタスクが実行できる最大時間を指定します。
- アクティビティタスクのスケジュールからスタート (**timeoutType: SCHEDULE_TO_START**) – このタイムアウトは、タスクを実行するワーカーを利用できない場合に、Amazon SWF がアクティビティタスクをタイムアウトするまでに待機する時間を指定します。タイムアウトした期限切れのタスクは、別のワーカーに割り当てられません。
- アクティビティタスクのスケジュールからクローズ (**timeoutType: SCHEDULE_TO_CLOSE**) – このタイムアウトは、タスクがスケジュールされてから完了するまでにかかる時間を指定します。ベ

ストプラクティスとして、この値は、タスクのスケジュールからスタートまでのタイムアウトと、タスクのスタートからクローズまでのタイムアウトの合計よりも大きくしないでください。

> **Note**
>
> タイムアウトの種類ごとにデフォルト値があり、通常は NONE (無限) に設定されます。ただし、すべてのアクティビティの実行の最大時間は 1 年に制限されます。

アクティビティの種類を登録するときにこれらのデフォルト値を設定しますが、アクティビティタスクをスケジュールするときに新しい値でオーバーライドできます。これらのタイムアウトのいずれかが発生すると、Amazon SWF は ActivityTaskTimedOut 型の イベント をワークフロー履歴に追加します。このイベントの `timeoutType` 値属性では、これらのタイムアウトがいつ発生するかを指定します。それぞれのタイムアウトで、`timeoutType` の値は括弧内に示されます。イベントの属性には、アクティビティタスクがスケジュールされた日時 (`scheduledEventId`) およびスタートされた日時 (`startedEventId`) に対応するイベントの ID も含まれます。Amazon SWF は、イベントの追加に加えて、このタイムアウトが発生したことをディサイダーにアラートする新しい決定タスクをスケジュールします。

Amazon Simple Workflow Service エンドポイント

最新の Amazon SWF リージョンとエンドポイントのリストは、他のサービスのエンドポイントとともに「Amazon Web Services 全般のリファレンス」に記載されています。

Amazon SWF ドメインと関連するすべてのワークフローおよびアクティビティは、相互に通信するために同じリージョン内に存在している必要があります。さらに、リージョン内の登録済みのドメイン、ワークフロー、およびアクティビティは、他のリージョンには存在しません。たとえば、us-east-1 および us-west-2 の両方で「MySampleDomain」という名前のドメインを作成する場合、それらは別個のドメインとして存在します。ドメインに関連付けられたワークフロー、タスクリスト、アクティビティ、またはデータがリージョン間で共有されることはありません。

Amazon EC2 インスタンスなど、ワークフローで他の AWS リソースを使用する場合は、Amazon SWF リソースと同じリージョンにも存在する必要があります。この唯一の例外は、Amazon S3 や IAM など、複数のリージョンにまたがるサービスです。このようなサービスには、それをサポートするリージョンに存在するワークフローからアクセスできます。

Amazon Simple Workflow Service のその他のドキュメント

このデベロッパーガイドに加えて、以下のドキュメントが参考になります。

Amazon Simple Workflow Service API Reference (Amazon Simple Workflow Service API リファレンス)

Amazon Simple Workflow Service API Reference (Amazon Simple Workflow Service API リファレンス) は、アクション、リクエスト、およびレスポンスの構造とエラーコードを含め、Amazon SWF HTTP API に関する詳細情報を提供します。

AWS Flow Framework ドキュメント

AWS Flow Framework は、 Amazon SWF を使用してワークフローとアクティビティを管理する分散された非同期アプリケーションの実装プロセスを簡略化し、ワークフローロジックの実装に集中できるようにするプログラミングフレームワークです。

各 AWS Flow Framework は、設計されている言語でイディオマティックに動作するように設計されているため、選択した言語を自然に使用して、Amazon SWF のすべての利点を備えたワークフローを実装できます。

Flow AWS Framework for Java があります。AWS Flow Framework for Java デベロッパーガイドでは、 AWS Flow Framework for Java を取得、セットアップ、使用する方法について説明します。

AWS SDK ドキュメント

AWS Software Development Kit (SDKs) は、さまざまなプログラミング言語で Amazon SWF へのアクセスを提供します。SDK は HTTP API に非常によく似ていますが、一部の Amazon SWF 機能に対して言語固有のプログラミングインターフェイスも提供します。各 SDK の詳細については、以下のリンクを参照してください。

> **Note**
>
> 執筆時点で Amazon SWF をサポートしている SDK のみを以下に示します。利用可能な AWS SDKs「Amazon Web Services のツール」ページを参照してください。

Java

は、 AWS インフラストラクチャサービス用の Java API AWS SDK for Java を提供します。

使用できるドキュメントを表示するには、「AWS SDK for Java Documentation」(ドキュメント) のページを参照してください。また、以下のリンクに従って SDK リファレンスの Amazon SWF セクションに直接移動することもできます。

- Class: AmazonSimpleWorkflowClient
- Class: AmazonSimpleWorkflowAsyncClient
- Interface: AmazonSimpleWorkflow
- Interface: AmazonSimpleWorkflowAsync

JavaScript

AWS SDK for JavaScript を使用すると、開発者はブラウザまたはサーバー上の Node.js アプリケーション内の両方で利用できるシンプルでeasy-to-useを使用して、 AWS サービスを利用するライブラリまたはアプリケーションを構築できます。

使用できるドキュメントを表示するには、「AWS SDK for JavaScript Documentation」(ドキュメント) のページを参照してください。また、以下のリンクに従って SDK リファレンスの Amazon SWF セクションに直接移動することもできます。

- Class: AWS.SimpleWorkflow

.NET

AWS SDK for .NET は、Visual Studio プロジェクトテンプレート、 AWS .NET ライブラリ、C# コードサンプル、ドキュメントを含む単一のダウンロード可能なパッケージです。 AWS SDK for .NET を使用すると、Windows 開発者は Amazon SWF やその他の サービス用の .NET アプリケーションを簡単に構築できます。

使用できるドキュメントを表示するには、「AWS SDK for .NET Documentation」(ドキュメント) のページを参照してください。また、以下のリンクに従って SDK リファレンスの Amazon SWF セクションに直接移動することもできます。

- Namespace: Amazon.SimpleWorkflow
- Namespace: Amazon.SimpleWorkflow.Model

PHP

AWS SDK for PHP は、Amazon SWF への PHP プログラミングインターフェイスを提供します。

使用できるドキュメントを表示するには、「AWS SDK for PHP Documentation」(ドキュメント) のページを参照してください。また、以下のリンクに従って SDK リファレンスの Amazon SWF セクションに直接移動することもできます。

- Class: SwfClient

Python

AWS SDK for Python (Boto) は、Amazon SWF への Python プログラミングインターフェイスを提供します。

利用可能なドキュメントを表示するには、「boto: A Python interface to Amazon Web Services」(boto：Amazon Web Services への Python インターフェイス) ページを参照してください。また、以下のリンクに従ってドキュメントの Amazon SWF セクションに直接移動することもできます。

- Amazon SWF Tutorial (Amazon SWF のチュートリアル)
- Amazon SWF Reference (Amazon SWF リファレンス)

Ruby

AWS SDK for Ruby は、Amazon SWF への Ruby プログラミングインターフェイスを提供します。

使用できるドキュメントを表示するには、「AWS SDK for Ruby Documentation」(ドキュメント) のページを参照してください。また、以下のリンクに従って SDK リファレンスの Amazon SWF セクションに直接移動することもできます。

- クラス: AWS::SimpleWorkflow

AWS CLI ドキュメント

AWS Command Line Interface （ AWS CLI) は、 AWS のサービスを管理するための統合ツールです。ダウンロードと設定のためのツールを 1 つだけ使用すると、コマンドラインから複数の AWS サービスを制御して、スクリプトを使用して自動化できます。

の詳細については AWS CLI、AWS Command Line Interface「」ページを参照してください。

Amazon SWF で使用できるコマンドの概要については、「AWS CLI コマンドリファレンス」の「swf」を参照してください。

Amazon Simple Workflow Service のウエブリソース

Amazon SWF の詳細を学習したり、このサービスを使用してワークフローを開発したりするために、数多くのウェブリソースがあります。

Amazon SWF フォーラム

Amazon SWF フォーラムは、Amazon の他の Amazon SWF 開発者や Amazon SWF 開発チームのメンバーとやり取りし、質問をしたり答えを得たりするための場所です。

このフォーラムには、フォーラム: Amazon Simple WorkflowService でアクセスできます。

Amazon SWF のよくある質問

Amazon SWF のよくある質問では、一般的ユースケース、Amazon SWF 間の違い、その他のサービスなどの概要を含む、Amazon SWF に関してよくある質問の回答を提供しています。

このよくある質問には、「Amazon SWF FAQ」(Amazon SWF のよくある質問) でアクセスできます。

Amazon SWF のビデオ

YouTube の Amazon Web Services チャンネルでは、Amazon SWF を含むすべての Amazon Web Services のビデオトレーニングを提供しています。Amazon SWF 関連の動画の完全なリストについては、Amazon Web Services の Simple Workflowのクエリを使用します。

Ruby Flow の移行オプション

Ruby AWS Flow Framework の は、アクティブな開発が行われなくなりました。既存のコードは今後も機能し続けますが、新しい機能またはバージョンはありません。このトピックは、Amazon SWF を使用し続けるための使用および移行オプション、および Step Functions への移行方法について説明します。

オプション	説明
Ruby Flow Framework を使用し続ける	現状、Ruby Flow Framework は機能しています。何もしないでいると、コードは引き続きそのまま機能し続けます。近い将来、 AWS Flow Framework for Ruby から移行する予定です。
Java Flow Framework に移行する	Java Flow Framework は現在も開発中であり、今後も新しい機能や更新がリリースされます。

オプション	説明
Step Functions に移行する	Step Functions は、ステートマシンと呼ばれる視覚的なワークフローを使用して、分散アプリケーションのコンポーネントを調整する方法を提供します。
Flow Framework を使用せずに SWF API を直接使用する	Ruby を使用し続け、Ruby Flow Framework の代わりに SWF API を直接使用できます。

Ruby であっても Java であっても、Flow Framework が提供する利点は、ワークフローロジックに専念できることです。フレームワークが細かい通信と調整の多くを処理し、複雑性の一部が抽象化されます。Java Flow Framework に移行することで同レベルの抽象化を維持できます。または Amazon SWF SDK を直接操作することもできます。

Ruby Flow Framework を使用し続ける

AWS Flow Framework for Ruby は、短期的には引き続き機能します。 AWS Flow Framework for Ruby にワークフローが書き込まれている場合、これらは引き続き機能します。更新、サポート、セキュリティの修正がないため、近い将来 AWS Flow Framework for Ruby から移行するしっかりとした計画を立てることをお勧めします。

Java Flow Framework に移行する

AWS Flow Framework for Java はアクティブな開発に残ります。概念的には、 AWS Flow Framework for Java は AWS Flow Framework for Ruby に似ています。ワークフローロジックに集中でき、フレームワークはディサイダーロジックの管理に役立ち、Amazon SWF の他の側面の管理が容易になります。

- AWS Flow Framework for Java
- AWS Flow Framework for Java API リファレンス

Step Functions に移行する

AWS Step Functions は、Amazon SWF に似ていますが、ワークフローロジックがステートマシンによって制御されるサービスを提供します。視覚的なワークフローを使用して、分散アプリケーションとマイクロサービスのコンポーネントを調整するには、Step Functions を使用します。それぞれ別個

の関数 (タスク) を実行する個々のコンポーネントからアプリケーションを構築することで、簡単にアプリケーションをスケールおよび変更できます。Step Functions を使えば、安心してコンポーネントを調整し、アプリケーションの関数を配置できます。グラフィカルコンソールは、アプリケーションのコンポーネントを一連のステップとして可視化する手段を提供します。各ステップが自動的にトリガーおよび追跡され、エラーが発生した場合は再試行されるため、アプリケーションが毎回意図したとおりの順序で実行されます。また、Step Functions では各ステップの状態がログに記録されるため、問題が発生した場合は、問題を簡単に診断およびデバッグできます。

Step Functions では、Amazon States Language を使用して定義された宣言型 JSON で記述されたステートマシンを使用して、タスクの調整を管理します。ステートマシンを使用することで、アプリケーションロジックを制御するディサイダープログラムを作成および維持する必要がなくなります。Step Functions は、視覚的なワークフローを使用してアプリケーションコンポーネントを調整する直感的、効率的、迅速なアプローチを提供します。すべての新しいアプリケーション AWS Step Functions に を使用することを検討する必要があります。Step Functions は、 AWS Flow Framework for Ruby で現在実装しているワークフローの に移行するための優れたプラットフォームを提供します。

Ruby 言語のスキルを引き続き活用しながらタスクを Step Functions に移行するために、Step Functions には Ruby アクティビティワーカーのサンプルが用意されています。このサンプルはアクティビティワーカーを実装するベストプラクティスを使用しており、Step Functions にタスクロジックを移行するためのテンプレートとして使用できます。詳細については、「AWS Step Functions デベロッパーガイド」の「Ruby でのアクティビティワーカーの例」トピックを参照してください。

> **Note**
>
> 多くのお客様にとって、Ruby AWS Flow Framework 用 から Step Functions への移行が最適なオプションです。ただし、シグナルがプロセスに介入する必要がある場合、または結果を親に返す子プロセスを起動する必要がある場合は、Amazon SWF API を直接使用すること、または AWS Flow Framework for Java への移行を検討してください。

詳細については AWS Step Functions、以下を参照してください。

- AWS Step Functions デベロッパーガイド
- AWS Step Functions API リファレンス
- AWS Step Functions コマンドラインリファレンス

Amazon SWF API を直接使用する

AWS Flow Framework for Ruby は Amazon SWF の複雑さの一部を管理しますが、Amazon SWF API を直接使用することもできます。API を直接使用することで、タスクの実装と調整を完全に制御するワークフローを構築できます。進行状況を追跡してその状態情報を維持するといった、内部的に複雑な処理を行う必要はありません。

- Amazon Simple Workflow Service Developer Guide (Amazon Simple Workflow Service デベロッパーガイド)
- Amazon Simple Workflow Service API Reference (Amazon Simple Workflow Service API リファレンス)

ドキュメント履歴

次の表に、「Amazon Simple Workflow Service Developer Guide」(Amazon Simple Workflow Service デベロッパーガイド) の前回のリリース以後に行われた、ドキュメントの重要な変更を示します。

変更	説明	変更日
ドキュメントのみの更新。	Amazon SWF に、 AWS ユーザー通知に関するセクションが追加されました。 AWS のサービス これは、 AWS の通知の中心的な場所として機能する です AWS マネジメントコンソール。詳細については、「Amazon Simple Workflow Service AWS User Notifications での の使用」を参照してください。	2023 年 5 月 4 日
更新	Amazon SWF では、SWF ワークフローとその実行関連のアクションを管理するための新しいコンソールエクスペリエンスが提供されるようになりました。詳細については、「Amazon SWF コンソールのチュートリアル」を参照してください。	2022 年 9 月 12 日
更新	「タスク実行におけるクォータ」セクションを更新して `Maximum tasks scheduled per second` を追加し、「CloudWatch の Amazon SWF メトリクス」ページを更新して CloudWatch での非 ASCII リソース名の使用に関する情報を追加しました。	2021 年 5 月 12 日
新機能	Amazon Simple Workflow Service が、Amazon EventBridge をサポートするようになりました。詳細については、以下を参照してください。 • EventBridge for Amazon SWF • EventBridge ユーザーガイド	2020 年 12 月 18 日
新機能	Amazon Simple Workflow Service は、タグを使用した IAM アクセス許可をサポートしています。詳細については、以下を参照してください。	2019 年 6 月 20 日

変更	説明	変更日

	• Amazon SWF のタグ
	  • タグの管理
	  • ワークフロー実行のタグ付け
	  • タグを使用してドメインへのアクセスを制御する
	• TagResource
	• UntagResource
	• ListTagsForResource
	• RegisterDomain	

新機能	Amazon Simple Workflow Service が欧州 (ストックホルム) リージョンで利用可能になりました。	2018 年 12 月 12 日
更新	CloudTrail 統合に関する Amazon Simple Workflow Service のトピックを改善しました。「を使用した API コールの記録 AWS CloudTrail」を参照してください。	2018 年 8 月 7 日
更新	CloudWatch の新しい PendingTasks メトリクスに関する情報を追加しました。詳細については、「Amazon SWF のメトリクス」を参照してください。	2018 年 6 月 18 日
更新	コードサンプルの構文ハイライトを改善しました。	2018 年 3 月 29 日
更新	Ruby Flow ユーザーがそのプラットフォームから移行するための選択肢を説明するトピックを追加しました。詳細については、「Ruby Flow の移行オプション」を参照してください。	2018 年 3 月 9 日
更新	高度な概念のトピックへのナビゲーションの向上。「Amazon SWF の高度なワークフローの概念」を参照してください。	2018 年 2 月 19 日
更新	有効な統計情報を追加して CloudWatch メトリクスのドキュメントを強化しました。「CloudWatch の Amazon SWF メトリクス」を参照してください。	2017 年 12 月 4 日

変更	説明	変更日
更新	ドキュメントの構成を明確にするため、目次を変更しました。「API と決定イベントのメトリクス」に新しい情報を追加しました。	2017 年 11 月 9 日
更新	「Amazon SWF クォータ」セクションを更新して、すべてのリージョンのスロットリング制限を含めました。	2017 年 10 月 18 日
更新	`activity_list` との混同を避けるため、「Amazon SWF の開始方法」で `task_list` を `workflowId` に変更しました。	2017 年 7 月 25 日
更新	このガイド全体でコード例をクリーンアップしました。	2017 年 6 月 5 日
更新	このガイドの構成や内容を簡素化し、改善しました。	2017 年 5 月 19 日
更新	更新とリンクの修正。	2017 年 5 月 16 日
更新	更新とリンクの修正。	2016 年 10 月 1 日
Lambda タスクのサポート	ワークフローで従来のアクティビティタスクに加えて Lambda タスクを指定できます。詳細については、「AWS Lambda Amazon SWF のタスク」を参照してください。	2015 年 7 月 21 日
タスクの優先順位の設定のサポート	Amazon SWF には、タスクリストでタスクで優先順位を設定するためのサポートが追加され、低優先順位のタスクの前に、高優先順位のタスクを提供が試みられるようになりました。ワークフローとアクティビティのタスクの優先順位の設定方法は、「Amazon SWF でのタスク優先度の設定」に説明されています。	2014 年 12 月 17 日
更新	CloudTrail を使用して Amazon SWF API コールをログに記録する方法を説明する新しいトピック「を使用した API コールの記録 AWS CloudTrail」が追加されました。	2014 年 5 月 8 日

変更	説明	変更日
更新	Amazon SWF の CloudWatch メトリクスに関連する 2 つの新しいトピックが追加されました。「CloudWatch の Amazon SWF メトリクス」には、サポートされているメトリクスのリストと説明、「を使用した CloudWatch の Amazon SWF メトリクスの表示 AWS マネジメントコンソール」には、 AWS マネジメントコンソールでメトリクスを表示し、アラームを設定する方法に関する情報が含まれます。	2014 年 4 月 28 日
更新	新しいセクション「Amazon SWF の追加リソースとリファレンス情報」を追加しました。このセクションでは、Amazon SWF 開発者向けに、いくつかのサービスリファレンス情報と、追加のドキュメント、サンプル、コード、および他のウェブリソースに関する情報を示しています。	2014 年 3 月 19 日
更新	ワークフローのチュートリアルを追加しました。「Amazon SWF の開始方法」を参照してください。	2013 年 10 月 25 日
更新	AWS CLI 情報と例 を追加しました。	2013 年 8 月 26 日
更新	更新と修正。	2013 年 8 月 1 日
更新	ドキュメントを更新し、アクセスコントロールのために IAM を使用する方法について説明しました。	2013 年 2 月 22 日
初回リリース	Amazon Simple Workflow Service Developer Guide (Amazon Simple Workflow Service デベロッパーガイド) の初回リリース。	2012 年 10 月 16 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。