Amazon Simple Workflow Service

开发人员指南

API Version 2012-01-25

Amazon Web Services

Amazon Simple Workflow Service: 开发人员指南

Amazon Web Services

欢迎使用 Amazon Simple Workflow Service

欢迎使用 *Amazon Simple Workflow Service Developer Guide*。

通过 Amazon Simple Workflow Service (Amazon SWF) 可轻松构建在分布式组件中协调工作的应用程序。在 Amazon SWF 中，任务表示的是由您的应用程序组件所执行的逻辑工作单位。在应用程序中协调任务涉及依据应用程序逻辑流程管理任务间的依赖性，时间安排和并发性。Amazon SWF 可使您完全控制任务的执行和协调，而不必担心底层复杂性，例如跟踪它们的进度和维护它们的状态。

使用 Amazon SWF 时，可以实施工作程序来执行任务。这些工作程序可在云基础设施（例如 Amazon Elastic Compute Cloud (Amazon EC2)）或您自己的本地设施上运行。您可以创建长时间运行的任务，或者可能失败、超时或需要重新启动的任务—或者可能以不同吞吐量和延迟完成的任务。Amazon SWF 存储任务并在工作程序就绪后分配给它们，并且跟踪它们的进度，以及维护它们的状态，包括有关它们完成情况的详细信息。要协调任务，应编写可从 Amazon SWF 获得每个任务最新状态的程序，然后使用该程序启动后续任务。Amazon SWF 持久保持应用程序的执行状态，以便使该应用程序能够灵活应对单个组件的故障。借助 Amazon SWF，您可以独立执行、部署、扩展和修改这些应用程序组件。

Amazon SWF 能够支持各种不同的应用程序需求。它适用于一系列需要任务协作的使用案例，包括媒体处理、Web 应用程序后端、业务处理流程和分析管道。

Amazon SWF 简介

越来越多的应用程序依靠异步、分布式处理。这类应用程序的可扩展性是使用此方法的主要推动力。通过自主设计分布式组件，开发人员可以在应用程序负载增加时灵活独立地部署和扩展应用程序部件。云服务的可用性是另一个推动力。当应用程序开发人员开始利用云计算时，他们就需要将其现有本地资产与基于云的附加资产组合到一起。然而分布式异步方法的另一个推动力是应用程序所建立的过程模型的固有分布式特性；例如，订单履行业务过程的自动化可跨越多个系统和人工任务。

开发此类应用程序的过程很复杂。它需要您协作多个分布式组件的执行，并处理增长的延迟与远程通信固有的不可靠性。要实现这一点，您通常需要编写涉及消息队列和数据库的复杂基础设施，以及复杂的逻辑来同步组件。

Amazon Simple Workflow Service (Amazon SWF) 提供编程模型和基础设施来协作分布式组件并采用可靠方法来维护其执行状态，可使分布式异步应用程序的开发更简单。借助 Amazon SWF，您可以自由关注与其有区别的应用程序方面的构建。

简单工作流程概念

下面介绍了解 Amazon SWF 工作流程所需要的基本概念，本指南的后续章节进行进一步说明。下述讨论是对工作流程结构和组件的高层次概述。

Amazon SWF 的基本概念是*工作流程*。工作流程是一组开展一些目标的活动，以及协作这些活动的逻辑。例如，工作流程可以接收客户订单并采取任何执行该订单需要的操作。AWS 资源中运行的每一个工作流程都称为*域*，域用于控制工作流程的范围。一个 AWS 账户可以有多个域，每一个域都能包含多个工作流程，但不同域中的工作流程不能相互作用。

设计 Amazon SWF 工作流程时，要准确定义每一个必需的活动。然后，用 Amazon SWF 给每个活动注册一个活动类型。注册活动时，您要提供姓名和版本之类的信息，并要根据您期望活动花费的时长来提供一些超时值。例如，客户可能期望订单在 24 小时内发货。这类期望会报告您在注册活动时指定的超时值。

工作流程执行过程中，有些活动可能需要执行多次，可能要用不同的输入。例如，在客户订购工作流程中，您可能要执行一个处理已购买项目的活动。如果客户购买多个项目，则必须将此活动运行多次。Amazon SWF 具有表示活动调用的*活动任务*的概念。在我们的示例中，每一个项目的处理都可以用一个活动任务表示。

*活动工作人员*是一个接收活动任务、执行任务并返回结果的程序。请注意，任务本身可能实际由人来执行，在这种情况下，人将会使用活动工作人员软件来接收和处置任务。可能会以统计分析师作为示例，这个分析师会接收数据集、分析之，然后发回分析结果。

活动任务以及执行任务的活动工作人员可以同步运行，也可以异步运行。它们可以跨过多个可能处于不同地理区域的计算机分布，或者可以全部运行在同一台计算机上。不同的活动工作人员可以用不同的编程语言编写，且可运行在不同的操作系统上。例如，一个活动工作人员可能在亚洲运行于一台台式计算机，而另一个活动工作人员可能在北美洲于一台便携式计算机上运行。

工作流程中的协作逻辑包含在被称为*决策者*的软件程序中。决策者排定活动任务、提供输入数据给活动工作人员、处理在工作流程处于进程中时到达的事件，并在目标完成时最终结束（或关闭）工作流程。

Amazon SWF 服务的角色是用作可靠的中央枢纽，决策者、活动工作人员与其他相关实体（如工作流程管理人员）通过它可以交换数据。Amazon SWF 还可以维持每个工作流程执行的状态，这样，您的应用程序不必持久存储状态。

决策者从 Amazon SWF 接收决策任务并将决策返回 Amazon SWF，以此来管理工作流程。决策表示的是一个操作或一组操作，是工作流程中的下一步。一般的决策为排定活动任务。决策还可以用于设置计时器以延迟活动任务执行、请求取消已处在进程中的活动任务以及完成或关闭工作流程。

活动工作人员和决策者接收其任务（分别为活动任务和决策任务）的机制是轮询 Amazon SWF 服务。

Amazon SWF 向决策者通知工作流程状态，其中包括每个决策任务、一份当前工作流程执行历史。工作流程执行历史由事件组成，其中事件代表工作流程执行状态的重要更改。示例事件为任务完成、任务已超时的通知或在工作流程执行之前设置的计时器过期。历史是工作流程进程的完整、一致且权威的记录。

用户必须具有授权的 AWS 访问密钥才能用您的账户运行工作流程。但是，访问密钥提供对您的账户的所有资源的完全访问，难于撤销，因此不适用于所有应用程序。Amazon SWF 访问控制使用 AWS Identity and Access Management (IAM)，以不公开访问密钥的受控受限方式提供对 AWS 资源的访问权限。例如，您可以允许用于访问您的账户，但只运行其在特定的域中运行特定工作流程。

工作流程执行

综合前述章节中讨论的想法，下面是对在 Amazon SWF 中开发和运行工作流程的步骤的概述：

1. 编写在您的工作流程中执行处理步骤的活动工作人员。
2. 编写决策者以执行您的工作流程的协作逻辑。
3. 用 Amazon SWF 注册您的活动和工作流程。

   您可以以编程的方式或使用 AWS 管理控制台执行此操作。
4. 启动您的活动工作人员和决策者。

   上述参与者能在可访问 Amazon SWF 终端节点的任何计算设备上运行。例如，您可以在云中使用计算实例，例如 Amazon Elastic Compute Cloud (Amazon EC2)；在数据中心使用服务器；甚至是移动设备，以托管决策者或活动工作人员。一旦启动，决策者和活动工作人员应该向 Amazon SWF 轮询任务。
5. 启动工作流程的一个或多个执行。

   执行可以通过编程的方式或 AWS 管理控制台启动。

   每一个执行都独立运行，您可以为每个执行提供其自己的输入数据集。执行启动后，Amazon SWF排定初始决策任务。作为响应，您的决策者开始生成启动活动任务的决策。执行会继续，直到您的决策者做出关闭执行的决策。
6. 使用 AWS 管理控制台查看工作流程执行。

   您可以筛选并查看完整的运行详情以及已完成的执行。例如，您可以选择开启的执行以查看已完成的任务及其结果。

从这里开始，本开发人员指南将更深入地介绍此处提及的每个想法。

开始设置 Amazon SWF

Topics

本节讨论使用 Amazon Simple Workflow Service (Amazon SWF) 进行开发的先决条件以及可用的开发选项。使用任何 AWS 服务的第一步都是注册 AWS 账户，注册详情见下述章节所述。设置账户后，您可以选择使用 AWS 支持的任何编程语言来进行 Amazon SWF 开发。针对 Java 和 Ruby 开发人员，还提供了 AWS Flow Framework。通过 AWS Identity and Access Management，您可以授予 AWS 账户拥有者以外的人员对 Amazon SWF 资源的访问权限。

AWS 账户和访问密钥

若要访问 Amazon SWF，您需要注册 AWS 账户。

如需注册 AWS 账户

1. 请转至 http://aws.amazon.com，然后单击“注册”。
2. 按照屏幕上的说明进行操作。

   在注册过程中，您会接到一个电话，需要您使用电话按键输入 PIN 码。

注册过程完成后，AWS 会向您发送一封确认电子邮件。若要随时查看当前账户活动并管理账户，请转到 http://aws.amazon.com，然后单击我的账户/控制台。

注册时，AWS 会向您提供与您的账户相关的访问密钥。下面是这些访问密钥的示例。

- 访问密钥 ID 示例：AKIAIOSFODNN7EXAMPLE
- 私有访问密钥示例：wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

您的密钥只有您和 AWS 知道。为密钥保密，以保护您的账户。在安全位置安全存储密钥，切勿通过电子邮件发送它。请勿对组织外部共享密钥，即使有来自 AWS 或 Amazon.com 的询问。合法代表 Amazon 的任何人永远都不会要求您提供密钥。

For console access, use your IAM user name and password to sign in to the AWS Management Console using the IAM sign-in page. IAM lets you securely control access to AWS services and resources in your AWS account. For more information about creating access keys, see How Do I Get Security Credentials? in the *AWS General Reference*.

开发选项

有多种方式可通过 Amazon Simple Workflow Service 实现工作流程解决方案。

Topics

HTTP 服务 API

Amazon SWF 提供可通过 HTTP 请求访问的服务操作。您可以使用这些操作直接与 Amazon SWF 通信，还可以使用这些操作以任何可通过 HTTP 与 Amazon SWF 通信的语言来开发自己的库。

您可以使用服务 API 开发决策者、活动工作人员或工作流程启动者。您还可以通过 API 访问可视性操作，以开发您自己的监控和报告工具。

有关如何使用 API 的信息，请参阅向 Amazon SWF 发出 HTTP 请求 (p. 86)。有关 API 操作的详细信息，请转到 *Amazon Simple Workflow Service API Reference*。

AWS 软件开发工具包

适用于 Java、.NET、Node.js、PHP、PHP2、Python 和 Ruby 的 AWS 软件开发工具包支持 Amazon SWF，可简便地在您选择的编程语言中使用 Amazon SWF HTTP API。

您可以利用这些库暴露的 API 开发决策者、活动或工作流程启动者。此外，还可通过这些库访问可视性操作，以使您可开发您自己的 Amazon SWF 监控和报告工具。

要下载任何 AWS 软件开发工具包，请访问 http://aws.amazon.com/code。

有关每个软件开发工具包中 Amazon SWF 方法的详细信息，请参阅所使用的软件开发工具包的语言特有的参考文档。

下面是可用的 AWS 开发工具包文档列表。

- AWS SDK for Java API Reference
- AWS SDK for .NET API Reference
- AWS SDK for JavaScript in Node.js API Reference
- AWS SDK for PHP API Reference
- AWS SDK for PHP API Reference
- AWS SDK for Python (Boto) API Reference
- AWS SDK for Ruby API Reference

AWS Flow Framework

AWS Flow Framework 是一种增强型开发工具包，用于编写可在 Amazon SWF 上作为工作流程运行的分布式异步程序。它可用于 Java 和 Ruby 编程语言，并提供了可简化复杂分布式程序编写的类。

借助于 AWS Flow Framework，您可以使用预配置的类型将工作流程定义直接映射到程序中的方法。

AWS Flow Framework 支持面向对象的标准概念（例如，基于异常的错误处理），这样就可更方便地实现复杂工作流程。您可以完全在自己首选的编辑器或 IDE 中创建、执行和调试使用 AWS Flow Framework 编写的程序。有关详细信息，请参阅 AWS Flow Framework 网站。

以下是 AWS Flow Framework 文档的链接：

- AWS Flow Framework for Java Developer Guide
- AWS Flow Framework Java 参考
- AWS Flow Framework for Ruby Developer Guide
- AWS Flow Framework for Ruby API Reference

AWS Flow Framework 示例代码

除了 AWS Flow Framework 文档中显示的代码段外，还可在以下位置获取完整、可下载的代码示例：

- AWS Flow Framework 诀窍
- &SWF; 的 &FFlong;示例

开发环境

您需要设置适用于您将使用的编程语言的开发环境。例如，如果您打算使用 Java 进行 Amazon SWF 开发，则需要在每个开发工作站上安装 Java 开发环境，例如，适用于 Java 的 AWS 开发工具包。如果您使用 Eclipse IDE 进行 Java 开发，您可能还要考虑安装 AWS Toolkit for Eclipse。Toolkit 是一种 Eclipse 插件，可增加对 AWS 开发有用的功能。

如果您的编程需要需要运行时环境，您需要在运行这些过程的每个计算机上设置该环境。

终端节点

为了降低延迟并将数据存储在符合要求的位置，Amazon SWF 在不同地区提供了终端节点。

Amazon SWF 中的每个终端节点完全独立；在一个地区注册的任何域、工作流程和活动不与另一地区的这些项共享任何数据或属性。换言之，注册 Amazon SWF 域、工作流程或活动时，它仅*存在于注册它的地区*。例如，可在两个不同的地区注册名为 **`SWF-Flows-1`** 的域，但二者彼此不共享任何数据或属性 — 其中每个都是一个完全独立的域。

有关 Amazon SWF 终端节点的列表，请参阅地区和终端节点。

教程：Amazon SWF 和 Amazon SNS 的订阅工作流程

本节提供一个教程，其中介绍如何创建一个 Amazon SWF 工作流程应用程序，它由一组四个顺序运行的活动组成。其中还涵盖：

- 设置*默认*和*执行时间*工作流程和活动选项。
- 向 Amazon SWF 轮询决策和活动任务。
- 在活动与工作流程之间通过 Amazon SWF 传递数据。
- 等待*人工任务*并从活动任务向 Amazon SWF 报告检测信号。
- 使用 Amazon SNS 创建主题、让用户订阅它以及将消息发布到订阅的终端节点。

可配合使用 Amazon Simple Workflow Service (Amazon SWF) 和 Amazon Simple Notification Service (Amazon SNS) 以仿真“人工任务”工作流程 — 其中要求工作人员执行某些操作，然后与 Amazon SWF 通信以开始该工作流程中的下一活动。

由于 Amazon SWF 是基于云的 Web 服务，因此可在任何有 Internet 连接的地方发起与 Amazon SWF 的通信。在这种情况下，我们将使用 Amazon SNS 通过电子邮件和/或手机短信与用户通信。

本教程使用AWS SDK for Ruby访问 Amazon SWF 和 Amazon SNS，但另有许多开发方法（包括 AWS Flow Framework for Ruby）可供选择，通过这些方法可更简便地与 Amazon SWF 协作和通信。

有关 Amazon SWF 开发选项的完整列表，请参阅开发选项 (p. 5)。

本节内容：

关于工作流程

我们将开发的工作流程由以下四个主要步骤组成：

1. 向用户获取订阅地址（电子邮件或手机短信）。
2. 创建 SNS 主题，然后让所提供的终端节点订阅该主题。
3. 等待用户确认订阅。
4. 如果用户确认，则向该主题发布一条祝贺消息。

这些步骤包括完全自动化的活动（第 2 步和第 4 步）和其他活动（第 1 步和第 3 步），后者要求工作流程等待工作人员向活动提供某些数据，然后工作流程才能前进。

由于每一步都依赖于上一步生成的数据（必须先有终端节点，然后才能让其订阅主题，并且必须先有主题订阅，然后才能等待确认等）。本教程还将涵盖在完成时可怎样提供活动结果以及如何将输入传递给所安排的任务。Amazon SWF 处理活动与工作流程之间的协作和信息传递，反之亦然。

我们还使用键盘输入和 Amazon SNS 处理 Amazon SWF 与向工作流程提供数据的人员之间的通信。实际上，可使用许多不同的方法与真人用户通信，但 Amazon SNS 提供一种非常简便的方式，其中使用电子邮件或手机短信向用户通知工作流程中的事件。

先决条件

要按本教程进行操作，您需要以下各项：

- Amazon Web Services (AWS) 账户
- Ruby 解释器
- AWS SDK for Ruby

如果已设置好这些，即准备就绪，可以继续。如果不想运行示例，则仍可按本教程操作 — 本教程中的大多数内容适用于使用 Amazon SWF 和 Amazon SNS，无论使用何种开发方法 (p. 5)都是如此。

下载源代码

可从以下网址下载本教程的完整源代码：https://s3.amazonaws.com/codesamples/ruby/swf_sns_sample.zip

Tip

即使您有意将本教程中的代码直接键入（或剪切并粘贴）到您自己的源文件中，准备好下载的源代码以供与您自己的代码进行比较也有助于发现并解决后续过程中出现的问题。

教程步骤

本教程分为以下几步：

1. 第 1 部分：将 Amazon SWF 与SDK for Ruby配合使用 (p. 9)
2. 第 2 部分：实现工作流程 (p. 11)
3. 第 3 部分：实现活动 (p. 16)
4. 第 4 部分：实现活动任务轮询器 (p. 22)
5. 运行工作流程 (p. 23)

订阅工作流程教程第 1 部分：将 Amazon SWF 与 AWS SDK for Ruby配合使用

Topics

加入AWS SDK for Ruby

首先，创建一个名为 `utils.rb` 的文件。此文件中的代码将包含或创建（如有必要）工作流程和活动代码均使用的 Amazon SWF 域，并将提供一个位置以放置所有类共用的代码。

首先，需要在代码中加入 `aws-sdk` 库，这样即可使用SDK for Ruby提供的功能。

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

这样可访问 AWS 命名空间，其中可设置与全局会话相关的值（如 AWS 证书和地区），还可访问 AWS 服务 API。

配置 AWS 会话

我们将通过设置要使用的 AWS 证书（访问 AWS 服务需要这些证书）和 AWS 地区，配置 AWS 会话。

有多种方法可在适用于 Ruby 的 AWS 开发工具包中设置 AWS 证书：在环境变量 (AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY) 中设置这些证书，或用 AWS.config 设置这些证书。我们将使用后一种方法，从一个名为 `aws-config.txt` 的 YAML 配置文件中加载这些证书，该文件内容类似于此。

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

现在创建此文件，将以 *REPLACE_WITH_* 开头的字符串替换为您的 AWS 访问密钥 ID 和私有访问密钥。有关 AWS 访问密钥的详细信息，请参阅《Amazon Web Services 常规参考》中的如何获取安全证书？。

我们还需要设置要使用的 AWS 地区。由于我们将通过 Amazon SNS 使用手机短信 (SMS) 向用户的手机发送文字消息，因此需要确保使用地区 us-east-1。*当前仅该地区支持手机短信。*

Note

如果您无权访问 us-east-1，或不在乎运行演示时是否启用了手机短信，则可随意使用任何地区。可从示例中删除手机短信功能，并使用电子邮件作为单一终端节点以订阅 Amazon SNS 主题。

有关发送手机短信的详细信息，请参阅 *Amazon Simple Notification Service Developer Guide* 中的使用 Amazon SNS 发送和接收手机短信通知。

我们现在将一些代码添加到 `utils.rb` 以加载配置文件、获取用户的证书，然后向 AWS.config 同时提供证书和地区。

```
# 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 creds 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)
```

注册 Amazon SWF 域

要使用 Amazon SWF，需要设置*域*：将容纳工作流程和活动的具名实体。可注册多个 Amazon SWF 域，但这些域在您的 AWS 账户中的名称必须独一无二，并且工作流程不能跨越多个域进行交互：应用程序的所有工作流程和活动必须在同一域中才能彼此交互。

由于将在整个应用程序中使用同一域，因此将在 `utils.rb` 中创建一个名为 `init_domain` 的函数，它将检索名为 *SWFSampleDomain* 的 Amazon SWF 域。

注册域后，可将其重用于多个工作流程执行。但是，*尝试注册已存在的域是错误行为*，因此我们的代码首先将检查是否存在该域，如果可找到这个现有的域，则将使用它。如果无法找到该域，则我们将创建它。

要在SDK for Ruby中处理 Amazon SWF 域，请使用 AWS::SimpleWorkflow.domains，它返回一个可用于枚举和注册域的 DomainCollection：

- 要检查以了解是否已注册了域，可查看 AWS::Simpleworkflow.domains.registered 提供的列表。
- 要注册新域，请使用 AWS::Simpleworkflow.domains.register。

以下是 `utils.rb` 中 `init_domain` 的代码。

```
# 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
```

后续步骤

`utils.rb` 就此结束。接下来，我们将在第 2 部分：实现工作流程 (p. 11)中创建工作流程和启动器代码。

订阅工作流程教程第 2 部分：实现工作流程

到目前为止，我们的代码都是比较通用的。在此部分，我们将开始实际定义工作流程执行的操作以及需要什么活动才能实现它。

Topics

设计工作流程

回想一下，此工作流程的初步构想由以下步骤组成：

1. 向用户获取订阅地址（电子邮件或手机短信）。
2. 创建 SNS 主题，然后让所提供的终端节点订阅该主题。
3. 等待用户确认订阅。
4. 如果用户确认，则向该主题发布一条祝贺消息。

我们可将工作流程中的每步视为工作流程必须执行的一个*活动*。我们的*工作流程*负责在适当的时间安排每个活动以及协调活动之间的数据传输。

对于此工作流程，我们将为其中每个步骤创建一个单独的活动，并为其提供描述性名称：

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

这些活动将按顺序执行，并且后续步骤中将使用每步中的数据。

我们可将应用程序设计为所有代码都存在于一个源文件中，但这与 Amazon SWF 的设计宗旨相悖。它适合的工作流程可跨越整个 Internet 范围，因此我们至少要将应用程序分为两个单独的执行文件：

- `swf_sns_workflow.rb` - 包含工作流程和工作流程启动者。
- `swf_sns_activities.rb` - 包含活动和活动启动者。

可在单独的时段、单独的计算机甚至世界上的不同地点运行工作流程和活动实现。由于 Amazon SWF 跟踪工作流程和活动的详细信息，因此无论二者在何处运行，工作流程均可协调活动的安排和数据传输。

设置工作流程代码

我们首先将创建一个名为 `swf_sns_workflow.rb` 的文件。在此文件中，声明一个名为 SampleWorkflow 的类。以下是类声明及其构造函数 `initialize` 方法。

```
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(task_list) # the
 domain to look for decision tasks in. @domain = init_domain # the task list
is used to poll for decision tasks. @task_list = task_list # 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 => 'sub
scribe_topic_activity', :version => 'v1' }, { :name => 'wait_for_confirma
tion_activity', :version => 'v1' }, { :name => 'send_result_activity', :version
 => 'v1' }, ].reverse! # reverse the order... we're treating this like a stack.
 register_workflow end
```

如您所见，我们保留以下类实例数据：

- `domain` - 从 `utils.rb` 中的 `init_domain` 检索的域名。
- `task_list` - 传入到 `initialize` 中的任务列表。
- `activity_list` - 活动列表，其中具有我们将运行的活动的名称和版本。

域名、活动名称和活动版本足以使 Amazon SWF 明确地识别活动类型，因此为了安排活动，只需保留这些数据。

工作流程的 *decider* 代码将使用任务列表轮询决策任务并安排活动。

在此函数的结尾处，我们调用一个尚未定义的方法：`register_workflow`.接下来，我们将定义此方法。

注册工作流程

要使用工作流程类型，必须先注册它。如同活动类型一样，工作流程类型按其域、名称和版本进行标识。此外，如同域和活动类型二者一样，无法重新注册现有的工作流程类型。如果需要更改有关工作流程类型的任何内容，则必须为其提供新版本，基本上就是新建一个类型。

以下是 `register_workflow` 的代码，它用于检索我们在上次运行时注册的现有工作流程类型，如果尚未注册该工作流程，则注册它。

```
  # 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.work
flow_types.each do | a | if (a.name == workflow_name) && (a.version == work
flow_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}" @work
flow_type = @domain.workflow_types.register(workflow_name, workflow_version,
options) end puts "** registered workflow: #{workflow_name}" end
```

首先，我们通过循环访问域的 workflow_types 集合，查看是否已注册工作流程名称和版本。如果找到了匹配项，则将使用已注册的工作流程类型。

如果未找到匹配项，则注册一个新的工作流程类型（通过对我们从中搜索工作流程的同一workflow_types 集合调用 register），其名称为“swf-sns-workflow”，版本为“1”，并具有以下选项。

```
     options = { :default_task_start_to_close_timeout => 3600, :default_execu
tion_start_to_close_timeout => 24 * 3600 }
```

注册期间传入的选项用于为工作流程类型设置*默认行为*，因此不需要在每次开始执行新工作流程时设置这些值。

此处，我们只设置了一些超时值：从任务开始到其结束时可用的最长时间（一小时）以及工作流程执行完毕可用的最长时间（24 小时）。如果超出其中任意一个时间，则任务或工作流程将超时。

有关超时值的详细信息，请参阅超时类型 (p. 38)。

轮询决策

在每个工作流程执行的核心部分都有一个*决策者*。决策者的职责是管理工作流程自身的执行。决策者接收*决策任务*，然后通过安排新活动、删除并重新启动活动，或通过将工作流程执行的状态设置为完成、已取消或失败，响应这些任务。

决策者按照工作流程执行的*任务列表* 名称接收要响应的决策任务。要轮询决策任务，请对域的 decision_tasks 集合调用 poll 以遍历可用的决策任务。然后，可通过循环访问决策任务的 new_events 集合，检查其中是否有新事件。

返回的事件为 AWS::SimpleWorkflow::HistoryEvent 对象，并可使用返回的事件的 event_type 成员获取该事件的类型。有关历史记录事件类型的列表和描述，请参阅 *Amazon Simple Workflow Service API Reference* 中的 HistoryEvent。

下面是决策任务轮询器逻辑的开头。我们的工作流程类中一个名为 poll_for_decisions 的新方法。

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

我们现在将根据收到的 event_type 划分决策的执行。我们收到的第一个类型可能是 WorkflowExecutionStarted。收到此事件后，它表示 Amazon SWF 正在示意您的决策者应开始执行工作流程。我们首先将通过对轮询时收到的任务调用 schedule_activity_task，安排第一个活动。

我们将在活动列表中声明的第一个活动传递给它，由于我们颠倒了列表，因此可将其用作堆栈，而该活动占据列表上的 last 位置。我们定义的“活动”仅仅是由名称和版本号组成的映射，但 Amazon SWF 只需此项即可识别所安排的活动，假定已注册该活动。

```
        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( @activ
ity_list.last, { :task_list => "#{@task_list}-activities" } )
```

安排活动时，Amazon SWF 向在安排该活动时传入的活动任务列表发送一个*活动任务*，示意任务开始。我们将在 第 3 部分：实现活动 (p. 16) 中处理活动任务，但值得注意的是，我们在此并未执行任务。我们仅告知 Amazon SWF *应安排* 该任务。

我们将需要处理的下一个活动是 ActivityTaskCompleted 事件，当 Amazon SWF 从某个活动任务收到活动已完成的响应时发生该事件。

```
       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(@activ
ity_list.empty?) puts "!! All activities complete! Sending complete_workflow_ex
ecution..." 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.attrib
utes[:result], :task_list => "#{@task_list}-activities" } ) else task.sched
ule_activity_task( @activity_list.last, { :task_list => "#{@task_list}-activit
ies" } ) end end
```

由于我们按线性方式执行任务，并且一次仅执行一个活动，因此我们将借此机会从 `activity_list` 堆栈中弹出已完成的任务。如果这样导致列表变空，则表示我们的工作流程已完成。在此情况下，我们通过对任务调用 complete_workflow_execution，向 Amazon SWF 示意工作流程已完成。

如果列表中仍有条目，则我们将安排列表上的下一活动（仍处于最后一个位置）。但是，这次我们将查看上一活动在完成时是否向 Amazon SWF 返回了任何结果数据（将在事件属性的可选 `result` 键中提供这些结果数据）。如果该活动产生了结果，则我们将其作为 `input` 选项传递给所安排的下一活动以及活动任务列表。

通过检索完成的活动的 `result` 值以及通过设置安排的活动的 `input` 值，我们可将数据从一个活动传递到下一个，或可使用活动中的数据，根据活动得到的结果更改决策者中的行为。

就本教程而言，在定义工作流程的行为时，这两种事件类型最为重要。但是，活动可生成 ActivityTaskCompleted 以外的事件。我们将通过为 ActivityTaskTimedOut 和 ActivityTaskFailed 事件以及为 WorkflowExecutionCompleted 事件（当 Amazon SWF 处理我们在用尽要运行的活动时作出的 `complete_workflow_execution` 调用时将生成它）提供演示处理程序代码，包装决策者代码。

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

启动工作流程执行

在将为要轮询的工作流程生成任何决策任务之前，我们需要启动工作流程执行。

要启动工作流程执行，请对您已注册的工作流程类型 (AWS::SimpleWorkflow::WorkflowType) 调用 start_execution。我们将在此周围定义一个小型包装器，以利用在类构造函数中检索的 workflow_type 实例成员。

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

一旦执行工作流程，工作流程的任务列表（它作为工作流程执行选项传入 start_execution）上即开始显示决策事件。

与注册工作流程类型时提供的选项不同，不将传递给 start_execution 的选项视为工作流程类型的一部分。无需更改工作流程的版本，即可随意在每次执行工作流程时更改这些选项。

由于我们希望在运行文件时开始执行工作流程，因此添加一些将类实例化的代码，然后调用刚刚定义的 start_execution 方法。

```
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. task_list = 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 sep
arate command-line window, with" puts "the following command:" puts "" puts ">
 ruby swf_sns_activities.rb #{task_list}-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(task_list)
sample_workflow.start_execution end
```

为避免任务列表命名发生任何冲突，我们将使用 SecureRandom.uuid 生成可用作任务列表名称的随机 UUID，确保将不同的任务列表名称用于每次工作流程执行。

Note

任务列表用于记录有关工作流程执行的事件，因此，如果将同一任务列表用于多次执行同一工作流程类型，则可能获得在上一次执行期间生成的事件，当多次执行间隔较小（试验新代码或运行测试时经常这样）时尤为如此。

为了避免必须处理以前执行中的项目的问题，可对每次执行使用新任务列表，在开始工作流程执行时指定该列表。

此处还有一些代码，可向运行代码的人员（很可能是您）提供说明以及提供任务列表的“活动”版本。决策者使用此任务列表名称为工作流程安排活动，而活动实现将对此任务列表名称侦听活动事件以了解何时开始安排的活动并提供有关活动执行的最新消息。

这段代码还等待用户开始运行活动启动者，*然后再*开始工作流程执行，因此在所提供的任务列表上开始出现活动任务时，活动启动者将准备好作出响应。

后续步骤

我们已完成工作流程实现。接下来，在第 3 部分：实现活动 (p. 16)中，我们将定义活动和活动启动者。

订阅工作流程教程第 3 部分：实现活动

我们现在将实现工作流程中的每个活动，首先是基类，它为活动代码提供某些共有功能。

Topics

定义基本活动类型

设计工作流程时，我们指定了以下活动：

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

我们现在将实现这些活动中的每个。由于这些活动将共用某些功能，因此我们来做一点基础工作，并创建这些活动可共用的一些共有代码。我们将它称为 BasicActivity，并在一个名为 `basic_activity.rb` 的新文件中定义它。

如同其他源文件一样，我们将加入 `utils.rb` 以使用 `init_domain` 函数设置示例域。

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

接下来，我们将声明基本活动类和我们在每种活动中将感兴趣的一些共有数据。我们将保存活动的 AWS::SimpleWorkflow::ActivityType 实例、其 *name*，并提供一个特殊的数据成员，用于存储活动的 *results*。

```
class BasicActivity attr_accessor :activity_type attr_accessor :name attr_ac
cessor :results
```

这些属性访问在类的 `initialize` 方法中定义的实例数据，该方法采用活动的 *name*、可选的 *version* 以及将该活动注册到 Amazon SWF 时要使用的 *options* 的映射。

```
    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.activ
ity_types.each do | a | if (a.name == @name) && (a.version == version) @activ
ity_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_sched
ule_to_start_timeout => 120, :default_task_schedule_to_close_timeout => 3800,
```

```
:default_task_start_to_close_timeout => 3600 } end @activity_type = @domain.activ
ity_types.register(@name, version, options) end end
```

与工作流程类型注册一样，如果已注册某个活动类型，则可通过查看域的 activity_types 集合，检索该类型。如果找不到该活动，则将注册该活动。

此外，与工作流程类型一样，可设置在注册活动类型时与其存储在一起的*默认选项*。

我们的基本活动最后得到的是运行它的一致方式。我们将定义一个 do_activity 方法，该方法采用活动任务。如下所示，我们可使用传入的活动任务通过其 input 实例属性接收数据。

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

这样即封装 BasicActivity 类。现在，我们将用它使我们的活动变得简单一致。

定义 GetContactActivity

在工作流程执行期间运行的第一个活动是 get_contact_activity，该活动检索用户的 Amazon SNS 主题订阅信息。

新建一个名为 get_contact_activity.rb 的文件，并且需要 yaml（我们将使用它准备用于传递给 Amazon SWF 的字符串）和 basic_activity.rb（我们将使用它作为此 GetContactActivity 类的基础）。

```
  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
```

由于我们将活动注册代码放入 BasicActivity 中，因此 GetContactActivity 的 initialize 方法相当简单。我们仅仅用活动名称 get_contact_activity 调用基类构造函数。只需此项即可注册我们的活动。

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

现在我们将定义 do_activity 方法，它提示输入用户的电子邮件和/或电话号码。

```
    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_con
firmed = 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
```

在 do_activity 的结尾处，我们获得从用户检索的电子邮件和电话号码，将其放入映射中，然后使用 to_yaml 将整个映射转换为 YAML 字符串。这样做有一个重要原因：在完成活动时传递给 Amazon SWF 的任何结果均*必须仅为字符串数据*。Ruby 可轻松地将对象转换为 YAML 字符串，然后再转换回对象，这一点非常适合此用途。

至此，get_contact_activity 实现结束。接下来将在 subscribe_topic_activity 实现中使用此数据。

定义 SubscribeTopicActivity

我们现在将深入研究 Amazon SNS 并创建一个活动，该活动使用由 get_contact_activity 生成的信息让用户订阅 Amazon SNS 主题。

新建一个名为 subscribe_topic_activity.rb 的文件，添加我们用于 get_contact_activity 的相同要求，声明您的类，然后提供其 initialize 方法。

```
  require 'yaml' require_relative 'basic_activity.rb' # **SubscribeTopicActiv
ity** sends an SMS / email message to the user, asking for # confirmation. When
 this action has been taken, the activity is complete. class SubscribeTopicActiv
ity < BasicActivity def initialize super('subscribe_topic_activity') end
```

既然代码已就位，可设置和注册活动，那么添加一些代码以创建 Amazon SNS 主题。为此，我们将使用 AWS::SNS::Client 对象的 create_topic 方法。

将 create_topic 方法添加到您的类，该方法采用传入的 Amazon SNS 客户端对象。

```
    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
```

一旦具有该主题的亚马逊资源名称 (ARN)，即可将其与 Amazon SNS 客户端的 set_topic_attributes 方法配合使用以设置该主题的 *DisplayName*，用 Amazon SNS 发送手机短信时需要该项。

最后，我们将定义 do_activity 方法。首先将收集在安排该活动时通过 input 选项传递的任何数据。如前所述，必须以字符串（我们使用 to_yaml 创建了它）形式传递此项。在检索它时，我们将使用 YAML.load 将数据转换为 Ruby 对象。

以下是 do_activity 的开头，我们在此处检索输入数据。

```
    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 `swf_sns_utils.rb`). sns_client = AWS::SNS::Client.new( :config =>
AWS.config.with(:region => $SMS_REGION))
```

如果我们未收到任何输入，则无计可施，因此我们只好将活动视为失败。

但是，假如一切正常，则我们将继续充实 `do_activity` 方法，用AWS SDK for Ruby获取 Amazon SNS 客户端，然后将其传递给 `create_topic` 方法以创建 Amazon SNS 主题。

```
      # Create the topic and get the ARN activity_data[:topic_arn] = create_top
ic(sns_client) if activity_data[:topic_arn].nil? return false end
```

在此，有几点值得注意：

- 我们使用 AWS.config.with 设置 Amazon SNS 客户端的地区。由于我们要发送手机短信，因此我们使用在 `utils.rb` 中声明的支持手机短信的地区。
- 将该主题的 ARN 保存在 `activity_data` 映射中。这是将传递给我们的工作流程中下一活动的部分数据。

最后，此活动使用传入的终端节点（电子邮件和手机短信）让用户订阅该 Amazon SNS 主题。我们不要求用户同时输入*两个*终端节点，但是，我们至少需要一个。

```
      # 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.sub
scribe( { :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 采用主题 ARN、*protocol*（我们巧妙地将它伪装成相应终端节点的 `activity_data` 映射键）。

最后，我们以 YAML 格式重新打包下一活动的信息，以使我们可将其发回 Amazon SWF。

```
      # if at least one subscription arn is set, consider this a success. if
(activity_data[:email][:subscription_arn] != nil) or (activity_data[:sms][:sub
scription_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
```

至此，即完成 `subscribe_topic_activity` 的实现。接下来，我们将定义 `wait_for_confirmation_activity`。

定义 WaitForConfirmationActivity

用户订阅 Amazon SNS 主题后，仍需要确认订阅请求。在这种情况下，我们将等待用户通过电子邮件或手机短信进行确认。

等待用户确认订阅的活动称为 `wait_for_confirmation_activity`，下面我们将定义它。首先，新建一个名为 `wait_for_confirmation_activity.rb` 的文件，并像设置以前的活动那样设置它。

```
  require 'yaml' require_relative 'basic_activity.rb' # **WaitForConfirmation
Activity** 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
```

接下来，我们将开始定义 `do_activity` 方法，然后检索向名为 `subscription_data` 的本地变量中输入的任何数据。

```
    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)
```

既然我们已有主题 ARN，那么可通过新建 AWS::SNS::Topic 实例，检索主题，然向其传递该 ARN。

```
     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
```

现在，我们将检查该主题以了解用户是否已使用某个终端节点确认了订阅。我们只需要一个终端节点得到确认，即将活动视为成功。

Amazon SNS 主题保留该主题作出的订阅的列表，并且我们可通过了解订阅的 ARN 是否设置为 `PendingConfirmation` 以外的内容，检查用户是否已确认某个特定的订阅。

```
      # loop until we get some indication that a subscription was confirmed.
subscription_confirmed = false while(!subscription_confirmed) topic.subscrip
tions.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 sub
scribed? if sub.arn != 'PendingConfirmation' subscription_data[sub.pro
tocol.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
```

如果获取订阅的 ARN，则将其保存在活动的结果数据中，将其转换为 YAML，然后从 `do_activity` 返回 true，这表示活动成功完成。

由于等待确认订阅可能需要一段时间，因此我们将偶尔对活动任务调用 `record_heartbeat`。这样可告知 Amazon SWF 活动仍在进行处理，并可用于提供有关活动进度的最新消息（如果正在进行某些可报告进度的操作，如处理文件）。

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

至此，我们的 `while` 循环结束。如果我们由于某种原因未成功即退出 while 循环，则我们将报告失败并结束 `do_activity` 方法。

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

至此，`wait_for_confirmation_activity` 的实现结束。我们只剩一个活动要定义：`send_result_activity`.

定义 SendResultActivity

如果工作流程已进展到这里，则我们已成功地让用户订阅了 Amazon SNS 主题，并且用户已确认该订阅。

我们的最后一个活动 `send_result_activity`，使用用户订阅的主题和用户确认订阅的终端节点，向用户发送成功主题订阅的确认。

新建一个名为 `send_result_activity.rb` 的文件，并像设置至今为止的所有活动那样设置它。

```
  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_activ
ity') end
```

`do_activity` 方法的开头相似，也从工作流程获取输入数据，从 YAML 转换这些数据，然后使用主题 ARN 创建 AWS::SNS::Topic 实例。

```
    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
```

具有主题后，我们将向其发布一条消息（并且将其回显到屏幕上）。

```
     @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
```

发布到 Amazon SNS 主题将所提供的消息发送到对于该主题存在的*所有*已订阅和确认的终端节点。因此，如果用户*同时*通过电子邮件和手机短信进行确认，则用户将收到两条确认消息，每个终端节点一条。

后续步骤

至此，send_result_activity 的实现结束。现在，我们将在处理活动任务的活动应用程序中将所有这些活动联系在一起，并可启动活动作为回应，如第 4 部分：实现活动任务轮询器 (p. 22)所述。

订阅工作流程教程第 4 部分：实现活动任务轮询器

在 Amazon SWF 中，运行工作流程执行的活动任务显示在*活动任务列表*上，在工作流程中安排活动时提供该列表。

我们将实现一个基本活动轮询器，用于为我们的工作流程处理这些任务，并在 Amazon SWF 将任务放入活动任务列表以启动我们的活动时将其用于启动该活动。

首先，新建一个名为 swf_sns_activities.rb 的文件。我们将使用它：

- 将我们创建的活动类实例化。
- 将每个活动注册到 Amazon SWF。
- 轮询活动，并在其名称出现在活动任务列表上时对每个活动调用 do_activity。

在 swf_sns_activities.rb 中，添加以下语句以需要我们定义的每个活动类。

```
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'
```

现在，我们将创建类并提供一些初始化代码。

```
class ActivitiesPoller def initialize(domain, task_list) @domain = domain
@task_list = task_list @activities = {} # These are the activities we'll run
activity_list = [ GetContactActivity, SubscribeTopicActivity, WaitForConfirma
tionActivity, 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
```

除了保存传入的*域*和*任务列表*以外，此代码还将我们创建的每个活动类实例化。由于每个类均注册与其关联的活动（如果需要查看该代码，请参阅 basic_activity.rb），因此这足以让 Amazon SWF 了解我们将运行的所有活动。

对于每个实例化的活动，我们都将其存储在使用活动名称（如 get_contact_activity）作为键的映射上，因此我们可在活动轮询器代码（接下来我们将定义这段代码）中轻松查找这些活动。

新建一个名为 poll_for_activities 的方法，并对域持有的 activity_tasks 调用 poll 以获取活动任务。

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

我们可从任务的 activity_type 成员获取活动名称。接下来，我们将使用与此任务关联的活动名称查找要对其运行 do_activity 的类，并向其传递此任务（其中包括应传输到活动的任何输入数据）。

```
      # find the task on the activities list, and run it. if @activit
ies.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[:reas
on], :details => activity.results[:detail] } ) end else puts "couldn't find key
 in @activities list: #{activity_name}" puts "contents: #{@activities.keys}"
end end end end
```

这段代码仅仅等待 do_activity 完成，然后根据返回代码对此任务调用 complete! 或 fail!。

Note

启动最后一个活动后，此代码即退出轮询器，因为轮询器已完成其任务并已启动所有活动。在您自己的 Amazon SWF 代码中，如果可能会再次运行您的活动，则可能要使活动轮询器无限期运行。

至此，我们的 ActivitiesPoller 类代码结束，但我们将在文件结尾再添加一些代码，以使用户可在命令行下运行它。

```
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
```

如果用户在命令行下运行该文件（向其传递活动任务列表作为第一个参数），则此代码将轮询器类实例化，并启动它，轮询活动。轮询器结束后（在它启动最后一个活动后），我们打印一条消息即退出。

活动轮询器就此结束。接下来，您只需运行代码，观察其如何发挥作用，如运行工作流程 (p. 23)所述。

订阅工作流程教程：运行工作流程

既然已完成工作流程、活动以及工作流程和活动轮询器的实现，那么我们即准备就绪，可运行工作流程。

如果尚未这样做，则将需要在 aws-config.txt 文件中提供 AWS 访问密钥，如本教程的第 1 部分配置 AWS 会话 (p. 9)所述。

现在，转到命令行，然后转到本教程源文件所在的目录。您应有以下文件：

```
.
|-- 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
```

现在，使用以下命令启动工作流程。

```
ruby swf_sns_workflow.rb
```

此命令将开始工作流程，并应显示一条消息，其中有一行，您可将该行复制并粘贴到单独的命令行窗口（甚至其他计算机，如果已将本教程源文件复制到它上面）中。

```
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...
```

工作流程代码将耐心等待您在单独的窗口中启动活动轮询器。

打开一个新的命令行窗口，再次转到源文件所在的目录，然后使用 swf_sns_workflow.rb 文件提供的命令启动活动轮询器。例如，如果收到了前面的输出，则要键入（粘贴）以下内容。

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

一旦开始运行活动轮询器，它即开始输出有关活动注册的信息。

```
** 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
```

现在可返回原有的命令行窗口，然后按回车键以启动工作流程执行。它将注册工作流程并安排第一个活动。

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

返回另一窗口，活动轮询器运行的位置。现在应看到所运行的第一个活动的结果，其中还提示您输入电子邮件和/或手机短信号码。输入其中一项或两项数据，然后确认所输入的文本。

```
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

此处提供的电话号码为虚构，仅作说明用途。请在此处使用您自己的手机号码和电子邮件地址！

输入此信息后不久，您应从 Amazon SNS 收到电子邮件或文字消息，要求您确认主题订阅。如果输入了手机短信号码，则将看到手机上显示类似如下内容。

如果向此短信回复 YES，则将获得在 send_result_activity 中提供的响应。

在发生所有这些情况时，您看到命令行窗口中发生了什么情况？工作流程和活动轮询器均在努力工作。

以下是来自工作流程轮询器的输出。

```
** 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...
```

以下是来自活动轮询器的输出，同一时间在另一命令行窗口中发生。

```
++ 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 com
plete!
++ Activity task completed: send_result_activity
All done!
```

恭喜，您的工作流程已完成，而本教程也已完成！

您可能要再次重新运行工作流程以了解超时如何发挥作用或输入其他数据。请记住，订阅主题后，*即已订阅，直到取消订阅为止*。取消订阅主题之前重新运行工作流程可能将导致自动成功，因为 wait_for_confirmation_activity 将发现已确认您的订阅。

要从 Amazon SNS 主题取消订阅，请执行以下操作

- 向该手机短信回复拒绝（发送 STOP）。
- 单击在电子邮件中收到的取消订阅链接。

现已准备就绪，可重新订阅该主题。

我从这里可以继续进行哪些内容？

本教程涉及很多知识点，但对于AWS SDK for Ruby、Amazon SWF 或 Amazon SNS 仍有多得多的内容可供学习。有关详细信息和更多示例，请参阅以下各项的官方文档：

- AWS SDK for Ruby 文档
- Amazon Simple Notification Service 文档
- Amazon Simple Workflow Service 文档

Amazon SWF 中的基本概念

本章概括介绍了 Amazon Simple Workflow Service 并描述了其主要功能。本章中的主题提供了 Amazon SWF 的一些使用示例，不过，若要了解实现这里所述功能的更具体示例，需要参考标题为使用 API (p. 54) 的章节。

Topics

Amazon SWF 工作流程

Topics

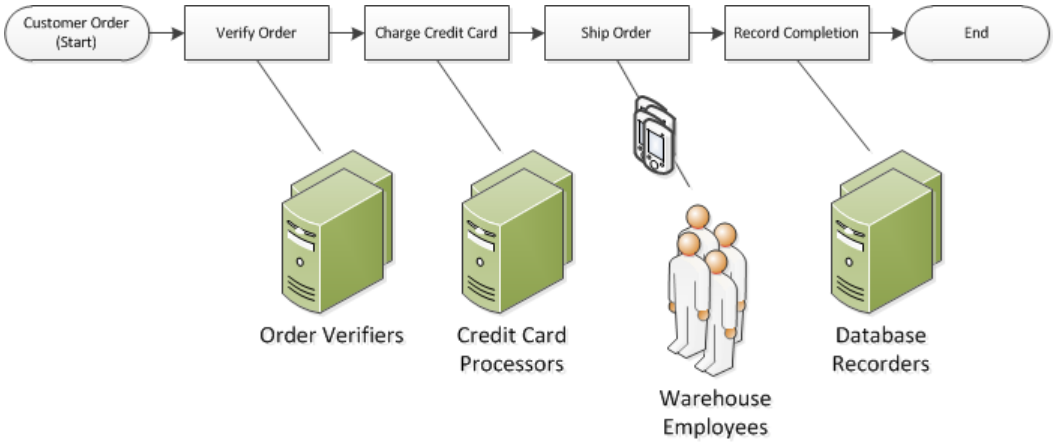
什么是工作流程？

使用 Amazon Simple Workflow Service (Amazon SWF)，您可以实现像*工作流程*这样的分布式异步应用程序。工作流程对可以跨多个计算设备异步运行并可进行顺序和并行处理的活动的执行进行协调和管理。

在设计工作流程时，需要对应用程序进行分析以识别其组件*任务*。在 Amazon SWF 中，这些任务由*活动*来表示。活动的执行顺序由工作流程的协作逻辑决定。

简单工作流程示例：电子商务应用程序

举例来说，下图显示的是简单的电子商务订单处理工作流程，其中涉及有人工过程和自动过程。



客户下订单时，这个工作流程启动。该工作流程包括四个*任务*：

1. 验证订单。
2. 如果订单有效，要求客户付款。
3. 如果付款完成，则按订单发货。
4. 如果发货完成，则保存订单详情。

此工作流程中的任务是*顺序*任务：用信用卡付款之前必须验证订单；按订单发货之前必须用信用卡成功付款；而在记录订单之前订单必须已发货。即便如此，也能在不同位置执行上述任务，因为 Amazon SWF 支持分布式过程。如果任务本质上可编程，则还可以用不同的编程语言或不同的工具编写这些任务。

除任务的顺序处理之外，Amazon SWF 还支持对任务并行处理的工作流程。并行任务是同时执行的，并且可由不同的应用程序或工作人员来完成。工作流程会作出有关在一个或更多并行任务执行完之后如何继续操作的决策。

工作流程注册和执行

在设计了协作逻辑和活动之后，您可以向 Amazon SWF 将这些组件注册为工作流程和活动类型。注册期间，您要为每种类型指定一个名称、一个版本和一些默认配置值。

只有经过注册的工作流程和活动类型才能用于 Amazon SWF。在电子商务示例中，您将注册 CustomerOrder 工作流程类型与 VerifyOrder、ChargeCreditCard、ShipOrder 和 RecordCompletion 活动类型。

注册完您的工作流程类型后，您可以按经常使用的方式运行它。*工作流程执行*是正在运行的工作流程实例。在电子商务示例中，新工作流程执行从每一个客户订单开始。

工作流程执行可从任何过程或应用程序，甚至是另一个工作流程开始执行。在电子商务示例中，用何种类型的应用程序启动工作流程取决于客户下订单的方式。工作流程可由网站或移动应用程序或使用公司内部应用程序的客户服务代表启动。

借助于 Amazon SWF，您可以将一个标识符（称为 `workflowId`）与该工作流程执行相关联，从而可以将现有业务标识符集成到工作流程中。在电子商务示例中，可以使用客户发票编号标识每个工作流程执行。

除了您提供的标识符外，Amazon SWF 还会将一个系统生成的唯一标识符（即 `runId`）与每个工作流程执行相关联。在任意给定时间，Amazon SWF 只允许有一个工作流程执行使用此标识符运行；虽然您有同一个工作流程类型的多个工作流程执行，但每个工作流程执行都具有不同的 `runId`。

另请参阅

- 工作流程历史 (p. 29)

Amazon SWF 工作流程历史

每个工作流程执行的进度都记录在 Amazon SWF 所保持的工作流程历史中。工作流程历史是每个自工作流程执行启动后发生的事件的详细、完整和一致性记录。事件表示的是您的工作流程执行状态的不连续更改，如被排定的新活动或完成的运行中活动。工作流程历史中包含每个导致工作流程执行状态更改的事件，如已排定和完成的活动、任务超时和信号。

一般而言，工作流程历史中不会出现不对工作流程执行状态造成更改的操作。例如，工作流程历史中不显示轮询尝试次数或可见性操作的使用。

工作流程历史有几个主要优势：

- 它可以使应用程序无状态，因为有关工作流程执行的所有信息都存储在其工作流程历史中。
- 对于每个工作流程执行，其历史记录中都包含被排定的活动、其当前状态和结果。工作流程执行使用此信息确定下面要执行的步骤。
- 历史记录提供具体的审查跟踪，您可以用它监控正在运行的工作流程执行并验证已完成的工作流程执行。

以下所示为电子商务工作流程历史的概念视图：

```
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
```

前述示例中，订单正等待发货。以下示例中，订单已完成。由于工作流程历史是累积形成的，会附加较新的事件：

```
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
```

工作流程执行历史中的事件以编程方式被描述为 JavaScript Object Notation (JSON) 数据元。历史记录本身是上述数据元的 JSON 阵列。每个事件都有以下内容：

- 类型，如 WorkflowExecutionStarted 或 ActivityTaskCompleted
- Unix 时间格式的时间戳
- 唯一标识事件用 ID

此外，每种类型的事件都有一组适用于该类型的独特描述性属性。例如，ActivityTaskCompleted 事件具有包含与该活动任务的排定时间和启动时间相对应的事件 ID 的属性，以及保存结果数据的属性。

您可以通过使用 GetWorkflowExecutionHistory 操作来获取工作流程执行历史的当前状态的副本。此外，作为 Amazon SWF 与您的工作流程决策者之间交互的一部分，决策者还将定期接收历史记录的副本。

以下部分是 JSON 格式的示例工作流程执行历史。

```
[ { "eventId": 11, "eventTimestamp": 1326671603.102, "eventType": "WorkflowExe
cutionTimedOut", "workflowExecutionTimedOutEventAttributes": { "childPolicy":
"TERMINATE", "timeoutType": "START_TO_CLOSE" } }, { "decisionTaskScheduledEvent
Attributes": { "startToCloseTimeout": "600", "taskList": { "name": "specialTask
List" } }, "eventId": 10, "eventTimestamp": 1326670566.124, "eventType": "De
cisionTaskScheduled" }, { "activityTaskTimedOutEventAttributes": { "details":
"Waiting for confirmation", "scheduledEventId": 8, "startedEventId": 0,
"timeoutType": "SCHEDULE_TO_START" }, "eventId": 9, "eventTimestamp":
1326670566.124, "eventType": "ActivityTaskTimedOut" }, { "activityTaskSchedule
dEventAttributes": { "activityId": "verification-27", "activityType": { "name":
 "activityVerify", "version": "1.0" }, "control": "digital music", "de
cisionTaskCompletedEventId": 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": "Activ
ityTaskScheduled" }, { "decisionTaskCompletedEventAttributes": { "executionCon
text": "Black Friday", "scheduledEventId": 5, "startedEventId": 6 }, "eventId":
 7, "eventTimestamp": 1326670266.103, "eventType": "DecisionTaskCompleted" },
{ "decisionTaskStartedEventAttributes": { "identity": "Decider01", "schedule
dEventId": 5 }, "eventId": 6, "eventTimestamp": 1326670161.497, "eventType":
"DecisionTaskStarted" }, { "decisionTaskScheduledEventAttributes": { "startTo
CloseTimeout": "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" }, { "decisionTaskStarte
dEventAttributes": { "identity": "Decider01", "scheduledEventId": 2 }, "eventId":
 3, "eventTimestamp": 1326668152.648, "eventType": "DecisionTaskStarted" }, {
"decisionTaskScheduledEventAttributes": { "startToCloseTimeout": "600", "task
List": { "name": "specialTaskList" } }, "eventId": 2, "eventTimestamp":
1326668003.094, "eventType": "DecisionTaskScheduled" } ]
```

有关工作流程执行历史中可能出现的不同类型事件的详细列表，请参阅 *Amazon Simple Workflow Service API Reference* 中的 HistoryEvent 数据类型。

Amazon SWF 会在执行关闭后将所有工作流程执行的完整历史存储可配置的天数。此段时间称为工作流程历史保留期，是在您注册该工作流程的*域*时指定的。本章节的后面部分将更详细地讨论域。

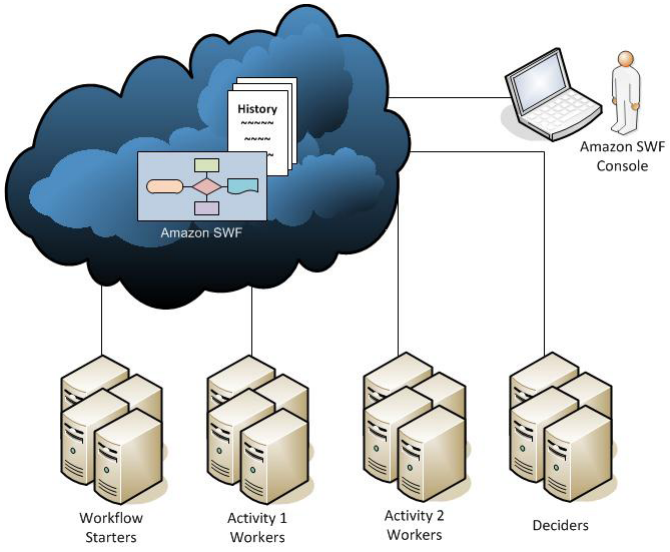
Amazon SWF 参与者

Topics

Amazon SWF 中的参与者是什么？

Amazon SWF 在其运行过程中，会与很多不同类型的程序*参与者*交互。参与者可以是 工作流程启动者 (p. 31)、决策者 (p. 31)或活动工作人员 (p. 32)。这些参与者通过其 API 与 Amazon SWF 通信。您可以用任何编程语言开发这些参与者。

下图显示了 Amazon SWF 架构，包括 Amazon SWF及其参与者。



工作流程启动者

工作流程启动者是能启动工作流程执行的任何应用程序。在电子商务示例中，一个工作流程启动者可以是客户下订单所在网站。而另一个工作流程启动者可以是客户服务代表代表客户下订单时所用移动应用程序或系统。

决策者

决策者是工作流程写作逻辑的执行。决策者控制工作流程执行中的活动任务流程。每当工作流程执行期间发生更改时（例如，活动任务完成），Amazon SWF 都会创建一个决策任务，该任务包含截至该时点的工作流程历史并将该任务分配给决策者。当决策者从 Amazon SWF 接收决策任务时，它会分析工作流程执行历史，以确定工作流程执行中接下来的相应步骤。决策者会使用*决策*将上述步骤反馈给 Amazon SWF。决策就是一个可代表接下来的各种操作的 Amazon SWF 数据类型。有关可能决策的列表，请转至 Amazon Simple Workflow Service API Reference 中的决策。

下面是 JSON 格式的决策示例，该决策通过这种格式传送到 Amazon SWF。此决策会排定新活动任务。

```
{ "decisionType" : "ScheduleActivityTask", "scheduleActivityTaskDecisionAttrib
utes" : { "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", "startToClose
Timeout" : "600", "heartbeatTimeout" : "120" } }
```

决策者会在工作流程执行启动时以及每当工作流程执行状态发生更改时接收决策任务。决策者接收决策任务并用更多决策来响应 Amazon SWF 以不断推进工作流程执行，直到决策者确定该工作流程执行完成为

止。然后，它会回应决策，以关闭工作流程执行。工作流程执行关闭后，Amazon SWF 不会为该执行安排其他任务。

在电子商务示例中，决策者会决定每个步骤是否都适当执行，然后回排定下一步或管理任何错误条件。

决策者代表的是单个计算过程或线程。多个决策者可为同一种工作流程类型处理任务。

活动工作人员

活动工作人员是一个进程或线程，此进程或线程执行作为工作流程一部分的*活动任务*。活动任务表示的是您在应用程序中标识的任务之一。

若要在工作流程中使用活动任务，您必须使用 Amazon SWF 控制台或 RegisterActivityType 操作来注册该任务。

每个活动工作人员都会轮询 Amazon SWF 以获取适合该活动工作人员执行的新任务；特定任务只能由特定活动工作人员执行。接收任务后，活动工作人员处理该任务直至完成，然后向 Amazon SWF 报告该任务已完成并提供结果。然后，活动工作人员会轮询新任务。与活动流程执行相关的活动工作人员会用此方式继续处理任务，直到工作流程执行本身已完成。在电子商务示例中，活动工作人员是信用卡处理人员和物流人员等执行过程中单独步骤的人所使用的独立过程和应用程序。

活动工作人员表示单个计算过程（或线程）。多个活动工作人员可处理同一种活动类型的任务。

参与者之间的数据交换

工作流程执行启动时会提供输入数据到工作流程执行。同样地，当活动工作人员排定活动任务时会提供输入数据给工作人员。当活动任务完成时，活动工作人员可将结果返回到 Amazon SWF。同样地，决策者会在执行完成时报告工作流程执行的结果。每个参与者都可通过用户自定义形式的字符串将数据发送到 Amazon SWF 或从其接收数据。根据数据的大小和敏感性，您可以直接传递数据，或传递指向存储在其他系统或服务上的数据的指针（如 Amazon S3 或 Amazon DynamoDB）。直接传递的数据和指向其他数据存储的指针都会记录在工作流程执行历史中；但是，Amazon SWF 不会从外部存储复制或缓存任何数据以作为历史记录的一部分。

由于 Amazon SWF 会维持每个工作流程执行的完整执行状态（包括任务的输入和结果），所有参与者都可能是无状态的。因此，工作流程处理具有高度可扩展。随着系统负载增加，您可以简单增加更多参与者来提高容量。

Amazon SWF 任务

Amazon SWF 通过为活动工作人员和决策者提供称为任务的工作分配来与它们交互。Amazon SWF 中有两种类型的任务：

- 活动任务。活动任务会指示活动工作人员执行其功能，如检查库存或向信用卡收费。活动任务中包含活动工作人员执行其功能需要的所有信息。
- 决策任务。决策任务告诉决策者工作流程执行状态已更改，从而使决策者能够确定下一个需要执行的活动。决策任务包含当前工作流程历史。

Amazon SWF 在工作流程启动时以及每当工作流程状态发生改变时（例如，活动任务完成时）会排定决策任务。每个决策任务中都包含整个工作流程执行历史标有页码的视图。决策者分析工作流程执行历史并向 Amazon SWF 回应一组决策，这些决策指定了工作流程执行中接下来应完成的操作。实质上，每个决策任务都会给决策者提供评估工作流程以及向 Amazon SWF 提供指示的机会。

为确保处理的决策没有冲突，Amazon SWF 会将每个决策任务分配给一个决策者，并且一次只允许一个决策任务在工作流程执行中处于活动状态。

下表显示的是与工作流程相关的不同构建与决策者之间的关系。

逻辑设计	注册为	执行人	接收和执行	生成
工作流程	工作流程类型	决策者	决策任务	决策

当活动工作人员完成活动任务时，它会向 Amazon SWF 报告该任务已完成，并且包含所生成的所有相关结果。Amazon SWF 通过一个指示任务已完成的事件来更新工作流程执行历史，然后排定一个决策任务以将更新后的历史记录传送给决策者。

Amazon SWF 将每个活动任务分配给一个活动工作人员。一旦任务分配后，其他活动工作人员就不能认领或执行该任务。

下图所示为与活动相关的不同构建之间的关系。

逻辑设计	注册为	执行人	接收和执行	生成
活动	活动类型	活动工作人员	个活动任务	结果数据

Amazon SWF域

域提供了在您的 AWS 账户内限定 Amazon SWF 资源的一种方法。工作流程的所有组成部分，如工作流程类型和活动类型，都必须指定在一个域中。一个域中可以有多个工作流程；但是，不同域中的工作流程不能相互作用。

设置新工作流程时，您需要在设置任何其它工作流程组成部分之前注册域，如果您此前没有注册。

注册域时，您需要指定*工作流程历史保留期*。该期限是 Amazon SWF 在工作流程执行完成后继续保留工作流程执行信息的时间长度。

Amazon SWF 数据元标识符

下面的列表描述了如何对 Amazon SWF 数据元（如工作流程执行）进行唯一标识。

- 工作流程类型:已注册的工作流程类型通过其域、名称和版本来标识。工作流程类型是在对 `RegisterWorkflowType` 的调用中指定的。
- 活动类型：已注册的活动类型通过其域、名称和版本来标识。活动类型是在对 `RegisterActivityType` 的调用中指定的。
- 决策任务和活动任务：每个决策任务和活动任务均通过一个唯一任务令牌来标识。任务令牌由 Amazon SWF 生成并在来自 `PollForDecisionTask` 或 `PollForActivityTask` 的响应中返回且含有关于该任务的其他信息。虽然令牌最常用于接收任务的过程，但该过程可以将令牌传给另一个过程，然后这个过程会报告任务完成或失败。
- 工作流程执行：工作流程执行通过域、工作流程 ID 和运行 ID 来标识。前两项是传递给 StartWorkflowExecution 的参数。运行 ID 由 `StartWorkflowExecution` 返回。

Amazon SWF 任务列表

任务列表提供与工作流程相关的各种任务的组织方法。您可以考虑任务列表与动态队列相似。在 Amazon SWF 中排定任务时，您可以指定用于放置该任务的队列（任务列表）。同样，在轮询 Amazon SWF 以获取任务时，您需要指定用于获取该任务的队列（任务列表）。

任务列表提供了灵活的机制，以在您的使用案例需要时将任务路由至工作人员。任务列表是动态的，因此您无需注册任务列表或通过一个操作来显式创建任务列表：只需排定一个任务，即可创建任务列表（如果该列表尚不存在）。

活动任务和决策任务的列表是分开的。始终只在一个任务列表上排定任务；不能跨过列表共享任务。

Topics

决策任务列表

每个工作流程执行都与一个特定的决策任务列表相关。注册工作流程类型时（RegisterWorkflowType 操作），您可以为该工作流程类型的执行指定一个默认任务列表。当工作流程启动者启动该工作流程执行时（`StartWorkflowExecution` 操作），它可以选择为该工作流程执行指定不同的任务列表。

当决策者轮询新决策任务（`PollForDecisionTask` 操作）时，该决策者会指定用于获取任务的决策任务列表。一个决策者可通过多次调用 `PollForDecisionTask` 来服务多个工作流程执行，在每个调用中，它会使用不同的任务列表，其中，每个任务列表对于某个工作流程执行来说都是特定的。或者，决策者可以轮询为多个工作流程执行提供决策任务的一个决策任务列表。您还可以使多个决策者一起轮询一个工作流程执行的任务列表，从而使它们全部服务该工作流程执行。

活动任务列表

一个活动任务列表中可包含不同活动类型的任务。任务在任务列表中是按顺序排定的。Amazon SWF 会按最合理的顺序从列表返回任务。某些情况下，任务可能不会按序出现在列表中。

注册活动类型时（RegisterActivityType 操作），您可以为该活动类型指定一个默认任务列表。默认情况下，这种类型的活动任务将在特定任务列表上进行排定；但是，当决策者排定活动任务时（ScheduleActivityTask 决策），它可以选择指定用于排定任务的不同任务列表。如果决策者没有指定任务列表，则会使用默认任务列表。因此，您可以根据任务属性将活动任务置放在特定任务列表中。例如，您可以将给定信用卡类型的活动任务的所有实例置放到特定任务列表中。

任务路由

当活动工作人员轮询新任务时（PollForActivityTask 操作），它可以指定用于获取任务的活动任务列表。如果指定了列表，活动工作人员将只能从该列表中接受任务。用此方法，您可以确保特定任务只会分配给特定的活动工作人员。例如，您可以创建任务列表以保存需要使用高性能计算机的任务。只有运行适当硬件的活动工作人员才能轮询该任务列表。另一个示例是，为某个特定地理区域创建一个任务列表。然后，您可以确保只有在该地区部署的工作人员才能接收这些任务。或者，您可以为高优先级订单创建一个任务列表，并始终优先检查该列表。

通过这种方式将特定任务分配给特定活动工作人员的操作称为*任务路由*。任务路由是可选操作；如果您没有在排定活动任务时指定任务列表，则任务会自动置放在默认任务列表中。

Amazon SWF 工作流程执行关闭

一旦您启动工作流程执行，该工作流程就会处于开启状态。开启的工作流程执行可以在完成、取消、失败或超时的情况下终止。它还可以继续作为新执行，或可以被终止。工作流程执行可由决策者、管理该工作流程的人员或 Amazon SWF 来关闭。

如果决策者确定工作流程活动已结束，则它应使用 RespondDecisionTaskCompleted 操作来关闭该工作流程执行，并传递 CompleteWorkflowExecution 决策。

或者，决策者还可以终止取消或失败状态下的工作流程执行。为了取消执行，决策者应使用 `RespondDecisionTaskCompleted` 操作并传递 CancelWorkflowExecution 决策。

如果工作流程执行进入到正常完成范围之外的状态，决策者应舍弃该工作流程执行。为了使执行失败，决策者应使用 `RespondDecisionTaskCompleted` 操作并传递 FailWorkflowExecution 决策。

Amazon SWF 对工作流程执行进行监视，以确保其不会超过用户指定的任何超时设置。如果工作流程执行超时，则 Amazon SWF 会自动将其关闭。有关超时值的更多信息，请参阅 超时类型 (p. 38) 一节。

决策者可能还要关闭该执行，并使用 `RespondDecisionTaskCompleted` 操作并传递 ContinueAsNewWorkflowExecution 决策以将其在逻辑上作为新的执行继续运行。这对于长时间运行的工作流程执行是一个非常有用的策略，因为这种工作流程执行的历史可能会随着时间的推移变得很大。

最后，您可以直接通过 Amazon SWF 控制台或使用 TerminateWorkflowExecution API 以编程方式终止工作流程执行。终止操作会迫使工作流程执行关闭。取消优先于终止，因为您的决策者可以管理工作流程执行的终止。

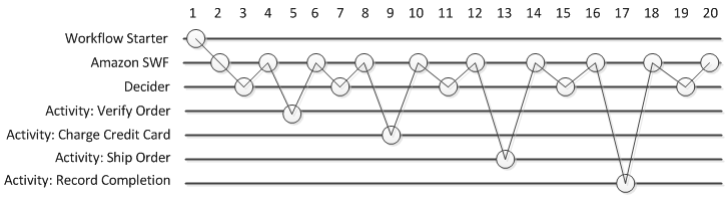
Amazon SWF 会在该执行超出特定的服务定义限制的情况下终止工作流程执行。Amazon SWF 还会在父工作流程已终止且适用的子策略指示子工作流程也应终止的情况下终止该子工作流程。

Amazon SWF 工作流程执行的生命周期

Amazon SWF 工作流程执行的生命周期

从工作流程执行开始到其完成的过程中，Amazon SWF 通过向参与者分配适当任务（活动任务或决策任务）来与其进行交互。

下图从工作流程上所执行组件的角度显示了订单处理工作流程执行的生命周期。



下表解释了上图中的每个任务。

工作流程执行生命周期

描述	操作、决策或事件
1) 工作流程启动者调用适当的 Amazon SWF 操作以启动订单的工作流程执行，从而提供订单信息。	StartWorkflowExecution 操作。
2) Amazon SWF 接收启动工作流程执行请求，然后排定第一个决策任务。	WorkflowExecutionStarted 事件和 DecisionTaskScheduled 事件。

描述	操作、决策或事件
3) 决策者从 Amazon SWF 接收任务，检查历史记录，应用协作逻辑以确定之前没有活动发生，做出决策以利用活动工作人员处理任务所需要的信息排定 Verify Order 活动，并将决策返回到 Amazon SWF。	PollForDecisionTask 操作。RespondDecisionTaskCompleted 操作，带有 ScheduleActivityTask 决策。
4) Amazon SWF 接收决策，排定 Verify Order 活动任务，并等待活动任务完成或超时。	ActivityTaskScheduled 事件。
5) 能够执行 Verify Order 活动的活动工作人员接收任务，执行该任务，并将结果返回到 Amazon SWF。	`PollForActivityTask` 操作和`RespondActivityTaskCompleted` 操作。
6) Amazon SWF 接收 Verify Order 活动的结果，将结果添加到工作流程历史，并排定决策任务。	ActivityTaskCompleted 事件和 DecisionTaskScheduled 事件。
7) 决策者从 Amazon SWF 接收任务，检查历史记录，应用协作逻辑，做出决策以通过活动工作人员处理任务所需要的信息排定 ChargeCreditCard 活动任务，并将决策返回到 Amazon SWF。	PollForDecisionTask 操作。RespondDecisionTaskCompleted 操作，带有 ScheduleActivityTask 决策。
8) Amazon SWF 接收决策，排定 ChargeCreditCard 活动任务，并等待任务完成或超时。	DecisionTaskCompleted 事件和 ActivityTaskScheduled 事件。
9) 能够执行 ChargeCreditCard 活动的活动工作人员接收任务，执行该任务，并将结果返回到 Amazon SWF。	`PollForActivityTask` 和 `RespondActivityTaskCompleted` 操作。
10) Amazon SWF 接收 ChargeCreditCard 活动任务的结果，将结果添加到工作流程历史，并排定决策任务。	ActivityTaskCompleted 事件和 DecisionTaskScheduled 事件。

描述	操作、决策或事件
11) 决策者从 Amazon SWF 接收任务，检查历史记录，应用协作逻辑，做出决策以通过活动工作人员执行任务所需要的信息排定 ShipOrder 活动任务，并将决策返回到 Amazon SWF。	PollForDecisionTask 操作。RespondDecisionTaskCompleted，带有 ScheduleActivityTask 决策。
12) Amazon SWF 接收决策，排定 ShipOrder 活动任务，并等待任务完成或超时。	DecisionTaskCompleted 事件和 ActivityTaskScheduled 事件。
13) 能够执行 ShipOrder 活动的活动工作人员接收任务，执行该任务，并将结果返回到 Amazon SWF。	`PollForActivityTask` 操作和`RespondActivityTaskCompleted` 操作。
14) Amazon SWF 接收 ShipOrder 活动任务的结果，将结果添加到工作流程历史，并排定决策任务。	ActivityTaskCompleted 事件和 DecisionTaskScheduled 事件。
15) 决策者从 Amazon SWF 接收任务，检查历史记录，应用协作逻辑，做出决策以通过活动工作人员执行任务所需要的信息排定 RecordCompletion 活动任务，并将决策返回到 Amazon SWF。	PollForDecisionTask 操作。RespondDecisionTaskCompleted 操作，带有 ScheduleActivityTask 决策。
16) Amazon SWF 接收决策，排定 RecordCompletion 活动任务，并等待任务完成或超时。	DecisionTaskCompleted 事件和 ActivityTaskScheduled 事件。
17) 能够执行 RecordCompletion 活动的活动工作人员接收任务，执行该任务，并将结果返回到 Amazon SWF。	`PollForActivityTask` 操作和`RespondActivityTaskCompleted` 操作。
18) Amazon SWF 接收 RecordCompletion 活动任务的结果，将结果添加到工作流程历史，并排定决策任务。	ActivityTaskCompleted 事件和 DecisionTaskScheduled 事件。

描述	操作、决策或事件
19) 决策者从 Amazon SWF 接收任务，检查历史记录，应用协作逻辑，做出决策以关闭工作流程执行，并将决策与任何结果一起返回到 Amazon SWF。	PollForDecisionTask 操作。RespondDecisionTaskCompleted 操作，带有 CompleteWorkflowExecution 决策
20) Amazon SWF 关闭工作流程执行，并将历史记录归档以便将来参考。	WorkflowExecutionCompleted 事件。

轮询 Amazon SWF 中的任务

决策者和活动工作人员使用*长轮询*与 Amazon SWF 通信。决策者或活动工作人员定期启动与 Amazon SWF 的通信，通知 Amazon SWF 它可以接受任务，然后指定用于获取任务的任务列表。

如果任务位于指定任务列表中，则 Amazon SWF 会立即在响应中返回该任务。如果没有提供任务，则 Amazon SWF 将保持 TCP 连接打开最长 60 秒，这样，如果任务在此时间内变为可用，就可在同一连接中返回该任务。如果 60 秒内没有提供任务，则会返回空响应并结束连接。（空响应是一种 Task 结构，其中的 taskToken 值为空字符串。）如果发生这种情况，决策者或活动工作人员应重新轮询。

长时间轮询对大容量任务处理有效。决策者和活动工作人员可能管理其自己的容量，且当决策者和活动工作人员处于防火墙后时使用方便。

有关更多信息，请参阅 轮询决策任务 (p. 64) 和 轮询活动任务 (p. 60)。

Amazon SWF 超时类型

为了确保工作流程执行正确运行，您可以通过 Amazon SWF 设置不同类型的超时。一些超时指定工作流程的总运行时长。其它超时指定活动任务在被分配给工作人员之前所花费的时间以及从排定到完成所花费的时间。Amazon SWF API 中的所有超时都以秒为单位指定。Amazon SWF 还支持将字符串“NONE”作为超时值，表示没有超时。

对于与决策任务和活动任务相关的超时，Amazon SWF 在工作流程执行历史中添加一个事件。事件属性会提供有关所发生超时类型的信息以及受影响的决策任务或活动任务。Amazon SWF 还会排定决策任务。决策者接收新决策任务时将会在历史中看到超时事件，并会通过调用 RespondDecisionTaskCompleted 操作来采取适当操作。

任务被视为从排定时开始到其关闭为止都处于开启状态。因此，当工作人员处理任务时，任务被报告为开启状态。当工作人员将任务状态报告为已完成、已取消或失败时，任务关闭。任务还有可能会被 Amazon SWF 因超时而关闭。

工作流程和决策任务中的超时

下图显示工作流程和决策超时如何与工作流程的生命周期相关：

与工作流程和决策任务相关的超时类型有两个：

- 工作流程启动到关闭 (timeoutType:START_TO_CLOSE)：此超时指定工作流程执行完成可花费的最大时间。工作流程注册期间，这一超时被设置为默认值，但当工作流程启动时，可用其它值覆盖该默认值。如果此超时被超出，Amazon SWF 会关闭工作流程执行并将事件（WorkflowExecutionTimedOut 类型）添加到工作流程执行历史中。除了 `timeoutType` 之外，事件属性还会指定对此工作流程执行有效的 `childPolicy`。子策略指定上级工作流程执行超时或终止时子工作流程执行的处理方法。例如，如果 `childPolicy` 设置为 TERMINATE，则子工作流程执行将被终止。一旦工作流程执行超时，您不能对其采取任何操作，除了可视性调用。
- 决策任务启动到关闭 (timeoutType:START_TO_CLOSE)：此超时指定相应决策者完成决策任务可花费的最大时间。该超时在工作流程类型注册期间设置。如果该超时被超出，会在工作流程执行历史中将任务标记为已超时，且 Amazon SWF 会将 DecisionTaskTimedOut 类型的事件添加到工作流程历史中。当此决策任务被排定 (`scheduledEventId`) 且启动 (`startedEventId`) 时，事件属性中将包含对应事件的 ID。除了添加事件外，Amazon SWF 还会排定新决策任务以警告决策者此决策任务已超时。此超时发生后，使用 `RespondDecisionTaskCompleted` 完成超时决策任务的尝试将失败。

活动任务中的超时

下图显示超时如何与活动任务的生命周期相关：

与活动任务相关的超时类型有四个：

- 活动任务启动到关闭 (timeoutType:START_TO_CLOSE)：此超时指定活动工作人员在接收到任务后处理此任务可花费的最大时间。使用 RespondActivityTaskCanceled、RespondActivityTaskCompleted 和 RespondActivityTaskFailed 尝试关闭超时的活动任务将会失败。
- 活动任务检测信号 (timeoutType:HEARTBEAT)：此超时指定任务在通过 `RecordActivityTaskHeartbeat` 操作提供其进度前可花费的最大时间。
- 活动任务排定到开始 (timeoutType:SCHEDULE_TO_START)：此超时指定在没有工作人员能执行活动任务时，在此任务超时前 Amazon SWF 的等待时间。一旦超时，过期的任务就不能分配给另一工作人员。
- 活动任务排定到关闭 (timeoutType:SCHEDULE_TO_CLOSE)：此超时指定从排定任务到完成此任务，此任务可花费的时间。最好的做法是，这个值不应该大于任务排定到启动超时与任务启动到关闭超时的总和。

Note

每一个超时类型都具有默认值，一般设置为 NONE（无限）。但是任何活动执行的最长时间均被限制为一年。

您在活动类型注册期间设置这些活动的默认值，但当您排定活动任务时可以用新值覆盖默认值。上述超时中有任何一个发生时，Amazon SWF 会将事件（ActivityTaskTimedOut 类型）添加到工作流程历史中。此事件的 `timeoutType` 值属性将指定发生了何种超时。对于其中每一个超时，`timeoutType` 的值都显示在括号中。当活动任务被排定 (`scheduledEventId`) 且启动 (`startedEventId`) 时，事件属性中还将包含对应事件的 ID。除了添加事件外，Amazon SWF 还会排定新决策任务以警告决策者已发生超时。

Amazon SWF 服务限制

Amazon SWF 对特定工作流程参数的大小进行了限制，如域的数量和工作流程执行历史的大小。有关限制的最新信息，请参阅 Amazon SWF 常见问题。

使用 IAM 管理对 Amazon SWF 资源的访问

访问 Amazon SWF 资源的每一个参与者，如决策者、活动工作人员和工作流程管理员，都必须有授权的 AWS 访问密钥。参与者可通过账户的访问密钥访问资源。但是，访问密钥提供对账户所有资源的无限制访问，难以撤销，所以并不适合所有应用程序。

Amazon SWF 使用 AWS Identity and Access Management (IAM)提供对资源的受控制访问。IAM 提供灵活的方法来管理对账户的 AWS 资源的访问，无需公开访问密钥。使用 IAM，您可以创建与 AWS 账户关联的一个或多个*用户*。每个用户都有一组独立的 IAM 访问密钥，以提供对账户资源的访问。然后，您可以将 *IAM 策略* 附加到用户或用户所在的群组，以指定用户可访问的资源。相比简单地指定是否允许或拒绝账户访问，该策略的粒度可以更细。例如，您可以创建允许用户访问账户的策略，但只能针对一组特定的域。

IAM 还具有这样的优点，您可以撤销 IAM 访问而不会影响到您的访问密钥。事实上，定期*交替轮换访问密钥*（撤销用户 IAM 访问密钥和发布新密钥）是安全最佳实践。

这个主题讨论的是如何使用 IAM 提供对 Amazon SWF 资源的受控访问方面的详细信息。假设您对以下文档中具体描述的 IAM 有大概了解。

- AWS Identity and Access Management (IAM)
- 使用 Identity and Access Management

基本原理

Amazon SWF 访问控制主要基于两种权限：

- 资源权限:用户可以访问哪些 Amazon SWF 资源。

  您可以只针对域授予资源权限。
- API 权限：用户可以调用哪些 Amazon SWF 操作。

最简单的方法是授权账户完全访问权（在任何域中调用任何 Amazon SWF 操作）或完全拒绝访问。但是，IAM 支持更细粒度的访问控制方法，这种方法通常更有用。例如，您可以：

- 允许用户不受限制地调用任何 Amazon SWF 操作，但只能在特定域中。您可以使用这样一个策略允许开发中的工作流程应用程序使用任何操作，但只能是“沙盒”域。

- 允许用户访问任何域，但限制其使用 API 的方法。您可以使用这样一个策略允许“审核员”应用程序在任何域中调用 API，但只允许读取访问。
- 允许用户在特定域中只调用一组有限的操作。您可以使用这类策略允许工作流程启动者只在特定域中调用 `StartWorkflowExecution` 操作。

Amazon SWF 访问控制基于以下原则：

- 访问控制决策只基于 IAM 策略；所有的审核与操作策略都通过 IAM 完成。
- 访问控制模型使用默认拒绝策略；任何未明确允许的访问都会被拒绝。
- 您可以通过附加适当的 IAM 策略到工作流程参与者，控制对 Amazon SWF 资源的访问。
- 资源权限只能针对域授予。
- 您可以通过将条件应用到一个或多个参数上来进一步限制某些操作的使用。
- 如果您授权使用 RespondDecisionTaskCompleted，则可以针对该操作中的决策列表授予权限。

  每个决策都由一个或多个参数，就像常规 API 调用那样。为了使策略可读性强一点，您可以授予决策权限，就像它们是实际的 API 调用那样，还包括将条件应用于一些参数。这些类型的权限称为*伪 API* 权限。

有关能用条件进行限制的常规和伪 API 参数的摘要，请参阅 API 摘要 (p. 45)。

Amazon SWF IAM 策略

IAM 策略包含一个或多个 Statement 元素，每一个元素都包含一组定义该策略的元素。有关元素的完整列表和如何构建策略的一般性讨论，请参阅访问策略语言。Amazon SWF 访问控制基于以下元素：

效果
: [必需] 语句的效果：拒绝 或允许。

  Note

  您必须明确允许访问；默认情况下，IAM 拒绝访问。

资源
: [必需] 语句适用的资源（用户可交互的 AWS 服务中的实体）。

  您可以只针对域授予资源权限。例如，策略只允许对您的账户中特定域进行访问。为了授予域权限，将 Resource 设置为域的亚马逊资源名称 (ARN)，其格式为“*Region*:*AccountID*:/domain/*DomainName*”。*Region* 是 AWS 地区，*AccountID* 是账户 ID（不含短划线），*DomainName* 是域名。

操作
: [必需] 语句适用的操作，您可以通过以下格式引用：*serviceId*:*action*.对于 Amazon SWF，将 *serviceID* 设置为 `swf`。例如，`swf:StartWorkflowExecution` 指的是 StartWorkflowExecution 操作，它用于控制允许哪些用户启动工作流程。

  如果您授予使用 RespondDecisionTaskCompleted 的权限，您还可以使用操作 来控制对所包含的决策列表的访问，以授予伪 API 的权限。由于 IAM 在默认情况下拒绝访问，决策者的决策必须被明确允许，否则决策将不会被接受。您可以使用 '*' 值允许所有决策。

条件
: [可选] 表达一个或多个操作参数的约束条件，以限制允许的值。

  Amazon SWF 操作的范围通常较广，您可以使用 IAM 条件缩小范围。例如，要对 PollForActivityTask 操作允许访问的任务列表进行限制，可以包含 Condition 并使用 `swf:taskList.name` 密钥指定可允许的列表。

您可以表达下列实体的约束条件。

- 工作流程类型。名称和版本具有单独密钥。
- 活动类型。名称和版本具有单独密钥。
- 任务列表。
- 标签。您可以为某些操作指定多个标签。在此情况下，每个标签都有一个单独的密钥。

Note

对于 Amazon SWF，值都是字符串，所以您可以使用 `StringEquals` 等字符串运算符限制参数，将参数限定为特定字符串。但是，`StringEquals` 等常规字符串比较运算符需要全部请求才能包含参数。如果您没有明确列入参数，且类型注册期间没有提供默认任务列表之类的默认值，访问将被拒绝。

将条件视为可选项通常很有用，这样一来，您可以调用操作，而无需列入相关参数。例如，您可能需要允许决策者指定一组 RespondDecisionTaskCompleted 决策，但同时允许它只为任何特定调用指定其中一个决策。在此情况下，您使用 `StringEqualsIfExists` 运算符限制相应参数，在参数满足条件时允许访问，但不会在参数不存在时拒绝访问。

有关可限制参数的完整列表和相关密钥，请参阅 API 摘要 (p. 45)。

下一节提供 Amazon SWF 策略构建方法示例。有关详细信息，请参阅字符串条件。

Amazon SWF 策略示例

工作流程由多个参与者（活动、决策者等）组成。您可以通过附加适当 IAM 策略控制每个参与者的访问权限。本章节提供了一些示例。以下所示为最简单的案例：

```
{ "Statement" : [ { "Effect" : "Allow", "Action" : "swf:*", "Resource" :
"arn:aws:swf:*:123456789012:/domain/*" } ] }
```

如果您将此策略附加到参与者，它对所有地区的账户都有访问权。您可以使用通配符利用一个值来表示多个资源、操作或地区。

- Resource 值 (...:swf:*:123...) 中的第一个通配符“*”表示资源权限适用于所有地区。要将权限限制到单个地区，可以用相应的地区字符串（如 us-east-1）替代“*”。
- Resource 值 (/domain/*) 中的第二个通配符“*”允许参与者访问特定地区内的任何账户域。
- Action 值中的“*”通配符允许参与者调用任何 Amazon SWF 操作。

有关通配符使用方法的详细信息，请参阅元素描述

以下章节显示的是以更精细方法授予权限的策略的示例。

域权限

如果您想将部门工作流程限制到一个特定域中，您可以使用与下述类似的方法：

```
{ "Statement": [ { "Effect" : "Allow", "Action" : "swf:*", "Resource" :
"arn:aws:swf:*:123456789012:/domain/department1" } ] }
```

如果您附加此策略到参与者，它可以调用任何操作，但只能针对 department1 域。

如果您希望参与者有权访问多个域，您可以针对每个域单独授予权限，如下所述：

```
{ "Statement": [ { "Effect" : "Allow", "Action" : "swf:*", "Resource" :
"arn:aws:swf:*:123456789012:/domain/department1" }, { "Effect" : "Allow", "Ac
tion" : "swf:*", "Resource" : "arn:aws:swf:*:123456789012:/domain/department2"
 } ] }
```

如果您附加此策略到参与者，它可以使用“department1”和“department2”域中的任何 Amazon SWF 操作。有时候，您还可以使用通配符表示多个域。

API 权限和约束条件

您控制用户可以对 Action 操作元素使用哪些操作。您可以选择通过使用 Condition 元素限制操作的可允许参数值。

如果您想将参与者限制为只进行特定操作，您可以使用与下述类似的方法：

```
{ "Statement": [ { "Effect" : "Allow", "Action" : "swf:StartWorkflowExecution",
 "Resource" : "arn:aws:swf:*:123456789012:/domain/department2" } ] }
```

如果将此策略附加到参与者，它能调用 `StartWorkflowExecution` 以在“department2”域中启动工作流程。它不能在任何其它域中使用任何其它操作或启动工作流程。

您可以进一步限制参与者通过限制一个或多个 `StartWorkflowExecution` 参数值启动的工作流程范围，如下所述：

```
{ "Statement": [ { "Effect" : "Allow", "Action" : "swf:StartWorkflowExecution",
 "Resource" : "arn:aws:swf:*:123456789012:/domain/department1", "Condition" :
{ "StringEquals" : { "swf:workflowType.name" : "workflow1" }, "StringEquals" :
 { "swf:workflowType.version" : "version2" } } } ] }
```

此策略限制 `StartWorkflowExecution` 操作的 name 和 version 参数。如果您附加此策略到参与者，它只能在 "department1" 域中运行 "workflow1" 的 "version2"，且两个参数都必须包含在请求中。

您可以使用 `StringEqualsIfExists` 运算符限制参数，无需将其包含在请求中，如下所述：

```
{ "Statement" : [ { "Effect" : "Allow", "Action" : "swf:StartWorkflowExecution",
 "Resource" : "arn:aws:swf:*:123456789012:/domain/some_domain", "Condition" :
{ "StringEqualsIfExists" : { "swf:taskList.name" : "task_list_name" } } } ] }
```

参与者可通过此策略在启动工作流程执行时选择性地指定任务列表。

您可以限制某些操作的标签列表。在此情况下，每个标签都有一个单独的密钥，因此，您使用 `swf:tagList.member.0` 限制列表中的第一个标签，使用 `swf:tagList.member.1` 限制列表中的第二个标签，以此类推，最多能限制 5 个。但是，您必须注意如何限制标签列表。关于实例，下面是*非* 推荐策略的示例：

```
{ "Statement" : [ { "Effect" : "Allow", "Action" : "swf:StartWorkflowExecution",
 "Resource" : "arn:aws:swf:*:123456789012:/domain/some_domain", "Condition" :
{ "StringEqualsIfExists" : { "swf:tagList.member.0" : "some_ok_tag", "anoth
er_ok_tag" } } } ] }
```

您可以通过此策略选择性地指定 "some_ok_tag" 或 "another_ok_tag"。但是，此策略只会限制标签列表的第一个元素。列表中其他元素允许是任意值，因为此策略不会将任何条件应用于 `swf:tagList.member.1`、`swf:tagList.member.2` 等等。

解决此问题的一个方法是禁止使用标签列表。以下策略确保，只有 "some_ok_tag" 或 "another_ok_tag" 能通过需要列表只包含一个元素的方式被允许。

```
{ "Statement" : [ { "Effect" : "Allow", "Action" : "swf:StartWorkflowExecution",
 "Resource" : "arn:aws:swf:*:123456789012:/domain/some_domain", "Condition" :
{ "StringEqualsIfExists" : { "swf:tagList.member.0" : "some_ok_tag", "anoth
er_ok_tag" }, "Null" : { "swf:tagList.member.1" : "true" } } } ] }
```

伪 API 权限和约束条件

如果需要限制提供给 `RespondDecisionTaskCompleted` 的决策，您首先必须允许参与者调用 `RespondDecisionTaskCompleted`。然后，您可以使用常规 API 的句法，授予适当伪 API 成员权限，如下所示：

```
{ "Statement" : [ { "Resource" : "arn:aws:swf:*:123456789012:/domain/*", "Action"
 : "swf:RespondDecisionTaskCompleted", "Effect" : "Allow" }, { "Resource" :
"*", "Action" : "swf:ScheduleActivityTask", "Effect" : "Allow", "Condition" :
{ "StringEquals" : { "swf:activityType.name" : "SomeActivityType" } } } ] }
```

如果您将此策略附加到参与者，第一个 `Statement` 元素将允许参与者调用 `RespondDecisionTaskCompleted`。第二个元素允许参与者使用 `ScheduleActivityTask` 决策，使 Amazon SWF 排定活动任务。为了允许所有决策，用 "swf:*" 替代 "swf:ScheduleActivityTask"。

您可以使用 Condition 运算符像使用常规 API 一样地限制参数。此 Condition 的 `StringEquals` 运算符允许 `RespondDecisionTaskCompleted` 为“SomeActivityType”活动排定一个活动任务，它必须排定该任务。如果需要允许 `RespondDecisionTaskCompleted` 使用一个参数值但不要求它必须使用，您可以使用 `StringEqualsIfExists` 运算符。

IAM 策略的服务模型限制

在创建 IAM 策略时，必须考虑服务模型约束条件。创建一个代表有效 Amazon SWF 请求的在句法上有效的 IAM 策略是有可能的；在访问控制方面得到允许的请求仍然失败，因为它是无效的请求。

对于实例，不建议对 ListOpenWorkflowExecutions 使用以下策略：

```
{ "Statement" : [ { "Effect" : "Allow", "Action" : "swf:ListOpenWorkflowExecu
tions", "Resource" : "arn:aws:swf:*:123456789012:/domain/domain_name", "Condi
tion" : { "StringEquals" : { "swf:typeFilter.name" : "workflow_name" },
"StringEquals" : { "swf:typeFilter.version" : "workflow_version" },
"StringEquals" : { "swf:tagFilter.tag" : "some_tag" } } } ] }
```

Amazon SWF 服务模型不允许 `typeFilter` 和 `tagFilter` 参数用于同一个 `ListOpenWorkflowExecutions` 请求。因此，该策略允许服务通过引发 `ValidationException` 作为无效请求加以拒绝的调用。

API 摘要

本章节简要描述了可以如何使用 IAM 策略对参与者使用各 API 和伪 API 访问 Amazon SWF 资源的方法进行控制。

- 对于除 `RegisterDomain` 和 `ListDomains` 之外的所有操作，您可以通过授予域资源权限的方式允许或拒绝对任何或所有账户域的访问。
- 您可以允许或拒绝任何常规 API 成员的权限，如果您授权调用 RespondDecisionTaskCompleted，还可以允许或拒绝任何伪 API 成员的权限。
- 您可以使用 Condition 限制某些参数的可允许值。

以下章节列出了可针对每个常规和伪 API 成员限制的参数，并提供了相关密钥，指出您可以根据其控制域访问的任何限制条件。

常规 API

本章节列出了常规 API 成员，并简要描述了可进行限制的参数和相关密钥。它还指出了您可以根据其控制域访问的任何限制条件。

CountClosedWorkflowExecutions

- tagFilter.tag：字符串约束条件。密钥是 `swf:tagFilter.tag`
- typeFilter.name：字符串约束条件。密钥是 `swf:typeFilter.name`。
- typeFilter.version：字符串约束条件。密钥是 `swf:typeFilter.version`。

Note

`CountClosedWorkflowExecutions` 要求 typeFilter 和 tagFilter 相互排斥。

CountOpenWorkflowExecutions

- tagFilter.tag：字符串约束条件。密钥是 `swf:tagFilter.tag`
- typeFilter.name：字符串约束条件。密钥是 `swf:typeFilter.name`。
- typeFilter.version：字符串约束条件。密钥是 `swf:typeFilter.version`。

Note

`CountOpenWorkflowExecutions` 要求 typeFilter 和 tagFilter 相互排斥。

CountPendingActivityTasks

- taskList.name：字符串约束条件。密钥是 `swf:taskList.name`。

CountPendingDecisionTasks

- taskList.name：字符串约束条件。密钥是 `swf:taskList.name`。

DeprecateActivityType

- activityType.name：字符串约束条件。密钥是 `swf:activityType.name`。
- activityType.version：字符串约束条件。密钥是 `swf:activityType.version`。

DeprecateDomain

- 您不能限制此操作的参数。

DeprecateWorkflowType

- workflowType.name：字符串约束条件。密钥是 `swf:workflowType.name`。
- workflowType.version：字符串约束条件。密钥是 `swf:workflowType.version`。

DescribeActivityType

- activityType.name：字符串约束条件。密钥是 `swf:activityType.name`。
- activityType.version：字符串约束条件。密钥是 `swf:activityType.version`。

DescribeDomain

- 您不能限制此操作的参数。

DescribeWorkflowExecution

- 您不能限制此操作的参数。

DescribeWorkflowType

- workflowType.name：字符串约束条件。密钥是 `swf:workflowType.name`。
- workflowType.version：字符串约束条件。密钥是 `swf:workflowType.version`。

GetWorkflowExecutionHistory

- 您不能限制此操作的参数。

ListActivityTypes

- 您不能限制此操作的参数。

ListClosedWorkflowExecutions

- tagFilter.tag：字符串约束条件。密钥是 `swf:tagFilter.tag`
- typeFilter.name：字符串约束条件。密钥是 `swf:typeFilter.name`。
- typeFilter.version：字符串约束条件。密钥是 `swf:typeFilter.version`。

Note

`ListClosedWorkflowExecutions` 要求 typeFilter 和 tagFilter 相互排斥。

ListDomains

- 您不能限制此操作的参数。

ListOpenWorkflowExecutions

- tagFilter.tag：字符串约束条件。密钥是 `swf:tagFilter.tag`
- typeFilter.name：字符串约束条件。密钥是 `swf:typeFilter.name`。
- typeFilter.version：字符串约束条件。密钥是 `swf:typeFilter.version`。

Note

`ListOpenWorkflowExecutions` 要求 typeFilter 和 tagFilter 相互排斥。

ListWorkflowTypes

- 您不能限制此操作的参数。

PollForActivityTask

- taskList.name：字符串约束条件。密钥是 `swf:taskList.name`。

PollForDecisionTask

- taskList.name：字符串约束条件。密钥是 `swf:taskList.name`。

RecordActivityTaskHeartbeat

- 您不能限制此操作的参数。

RegisterActivityType

- defaultTaskList.name：字符串约束条件。密钥是 `swf:defaultTaskList.name`。
- name：字符串约束条件。密钥是 `swf:name`。
- version：字符串约束条件。密钥是 `swf:version`。

RegisterDomain

- name：注册的域名可作为此操作的资源。

RegisterWorkflowType

- defaultTaskList.name：字符串约束条件。密钥是 `swf:defaultTaskList.name`。
- name：字符串约束条件。密钥是 `swf:name`。
- version：字符串约束条件。密钥是 `swf:version`。

RequestCancelWorkflowExecution

- 您不能限制此操作的参数。

RespondActivityTaskCanceled

- 您不能限制此操作的参数。

RespondActivityTaskCompleted

- 您不能限制此操作的参数。

RespondActivityTaskFailed

- 您不能限制此操作的参数。

RespondDecisionTaskCompleted

- decisions.member.N：通过伪 API 权限间接限制。有关详细信息，请参阅 伪 API (p. 48)。

SignalWorkflowExecution

- 您不能限制此操作的参数。

StartWorkflowExecution

- tagList.member.0：字符串约束条件。密钥是 `swf:tagList.member.0`
- tagList.member.1：字符串约束条件。密钥是 `swf:tagList.member.1`
- tagList.member.2：字符串约束条件。密钥是 `swf:tagList.member.2`
- tagList.member.3：字符串约束条件。密钥是 `swf:tagList.member.3`
- tagList.member.4：字符串约束条件。密钥是 `swf:tagList.member.4`
- taskList.name：字符串约束条件。密钥是 `swf:taskList.name`。
- workflowType.name：字符串约束条件。密钥是 `swf:workflowType.name`。
- workflowType.version：字符串约束条件。密钥是 `swf:workflowType.version`。

Note

您不能限制超过五个标签。

TerminateWorkflowExecution

- 您不能限制此操作的参数。

伪 API

本章节列出了表示 RespondDecisionTaskCompleted 中所包含决策的伪 API 成员。如果您已授权使用 `RespondDecisionTaskCompleted`，您的策略会用与常规 API 相同的方法来授予此 API 成员权限。您可以通过设置一个或多个参数的条件来限制伪 API 的某些成员。本章节列出了伪 API 成员，并简要描述了可进行限制的参数与相关密钥。

Note

`aws:SourceIP`、`aws:UserAgent` 和 `aws:SecureTransport` 密钥不可用于伪 API。如果您的预期安全策略需要这些密钥对伪 API 进行访问控制，您可以将其用于 `RespondDecisionTaskCompleted` 操作。

`CancelTimer`

- 您不能限制此操作的参数。

`CancelWorkflowExecution`

- 您不能限制此操作的参数。

`CompleteWorkflowExecution`

- 您不能限制此操作的参数。

`ContinueAsNewWorkflowExecution`

- tagList.member.0：字符串约束条件。密钥是 `swf:tagList.member.0`
- tagList.member.1：字符串约束条件。密钥是 `swf:tagList.member.1`
- tagList.member.2：字符串约束条件。密钥是 `swf:tagList.member.2`
- tagList.member.3：字符串约束条件。密钥是 `swf:tagList.member.3`
- tagList.member.4：字符串约束条件。密钥是 `swf:tagList.member.4`
- taskList.name：字符串约束条件。密钥是 `swf:taskList.name`。
- workflowTypeVersion：字符串约束条件。密钥是 `swf:workflowTypeVersion`。

Note

您不能限制超过五个标签。

`FailWorkflowExecution`

- 您不能限制此操作的参数。

`RecordMarker`

- 您不能限制此操作的参数。

`RequestCancelActivityTask`

- 您不能限制此操作的参数。

`RequestCancelExternalWorkflowExecution`

- 您不能限制此操作的参数。

`ScheduleActivityTask`

- activityType.name：字符串约束条件。密钥是 `swf:activityType.name`。
- activityType.version：字符串约束条件。密钥是 `swf:activityType.version`。
- taskList.name：字符串约束条件。密钥是 `swf:taskList.name`。

`SignalExternalWorkflowExecution`

- 您不能限制此操作的参数。

`StartChildWorkflowExecution`

- tagList.member.0：字符串约束条件。密钥是 `swf:tagList.member.0`
- tagList.member.1：字符串约束条件。密钥是 `swf:tagList.member.1`
- tagList.member.2：字符串约束条件。密钥是 `swf:tagList.member.2`
- tagList.member.3：字符串约束条件。密钥是 `swf:tagList.member.3`
- tagList.member.4：字符串约束条件。密钥是 `swf:tagList.member.4`
- taskList.name：字符串约束条件。密钥是 `swf:taskList.name`。
- workflowType.name：字符串约束条件。密钥是 `swf:workflowType.name`。
- workflowType.version：字符串约束条件。密钥是 `swf:workflowType.version`。

Note

您不能限制超过五个标签。

`StartTimer`

- 您不能限制此操作的参数。

Amazon SWF 中的高级概念

Topics

基本理念 (p. 27)一节中的电子商务示例说明了一种简化的工作流程情况。事实上，您可能希望您的工作流程执行并行任务（在核准信用卡的同时发送订单确认）、记录主要事件（所有项目都组到一起）、更新订单更改（增加或删除项目）以及进行其它更多高级决定作为工作流程执行的一部分。本章节描述的是高级工作流程功能，您可以用这些功能构建坚固复杂的工作流程。

版本控制

业务通常需要您对同一工作流程采用不同的实现方式或变化，或者需要同时运行多个活动。例如，您可能想要在另一个工作流程处于生产中时测试新执行的工作流程。您还可能想要用两种不同的功能集运行两个不同的执行，如基本和高级执行。您可以通过版本控制同时执行多个工作流程和活动，以满足您的要求。

工作流程和活动类型都有一个与之相关的版本，在注册时指定。版本是自由格式的字符串，您可以选择您自己的版本控制方案。为了能够创建某个已注册类型的新版本，您应该用同一名称和不同版本进行注册。前面所述的任务列表 (p. 33)可进一步帮助您执行版本控制。考虑这样一个情形，您的给定类型长期运行工作流程执行正在进程中，并考虑需要您修改工作流程（如增加新功能）的情况。您可以通过创建新版本的活动类型和工作人员以及新决策者。然后，您可以使用不同组的任务列表启动新工作流程版本的执行。通过这种方式，您可以同时运行不同版本工作流程的执行，而不会相互造成影响。

信号

信号可使您将信息插入正在运行的工作流程执行中。在某些应用场景中，建议您将信息增加到正在运行的工作流程执行中，使其了解有些情况发生了改变或向其通知外部事件。任何过程都能发送信号到开启的工作流程执行中。例如，一个工作流程执行可向另一个发送信号。

若要使用信号，请定义信号名称以及要传递至该信号的数据（如果有）。然后，对决策者进行编程以识别历史记录中的信号事件 (WorkflowExecutionSignaled)，并对其适当处理。当一个进程需要发出工作流程

执行的信号时，它会调用 Amazon SWF（使用 SignalWorkflowExecution 操作，或对于决策者而言，使用 SignalExternalWorkflowExecution 决策），后者会指定目标工作流程执行的标识符、信号名称和信号数据。然后，Amazon SWF 接收该信号，将其记录在目标工作流程执行的历史记录中，然后为其排定决策任务。当决策者收到决策任务时，它还会接收到工作流程执行历史内部的信号。然后，决策者可根据信号及其数据采取适当操作。

以下是信号的一些应用情况：

- 暂停工作流程执行的进展，直到收到信号（例如，等待库存发货）。
- 向工作流程执行提供可能会影响决策者做决策之逻辑的信息。这对受外部事件（例如，尝试在闭市后完成股票销售）影响的工作流程很有用。
- 在您期望发生更改时（例如，在下订单后发货之前更改订单数量）更新工作流程执行。

在应取消工作流程的情况下（例如，订单本身由客户取消），应使用 `RequestCancelWorkflowExecution` 操作而不是向该工作流程发送信号。

子工作流程

复杂的工作流程可使用子工作流程分解为更小、更可管理且有可能被重用的组件。子工作流程是被另一个（上层）工作流程执行启动的工作流程执行。为了启动子工作流程，父工作流程的决策者会使用 `StartChildWorkflowExecution` 决策。用此决策指定的输入数据可通过其历史记录提供给子工作流程。

`StartChildWorkflowExecution` 决策的属性还会指定*子策略*，即 Amazon SWF 对父工作流程执行在子工作流程执行之前终止这种情况的处理方式。可能存在以下三种值：

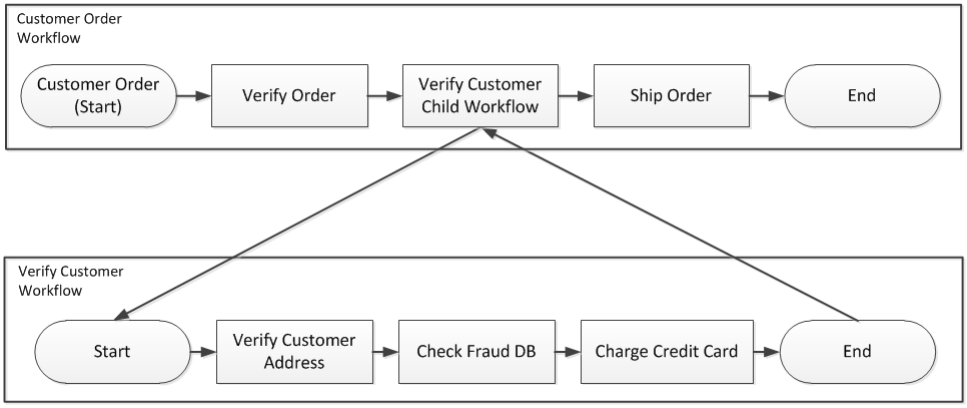
- TERMINATE：Amazon SWF 将终止子执行。
- REQUEST_CANCEL：Amazon SWF 将通过在子工作流程执行历史中放置一个 `WorkflowExecutionCancelRequested` 事件来尝试取消子执行。
- ABANDON：Amazon SWF 将不采取任何行动；子执行将继续运行。

子工作流程执行启动后，其运行方式类似于常规执行。执行完成时，Amazon SWF 会将完成情况及其结果记录在父工作流程执行的工作流程历史中。子工作流程的示例包括以下内容：

- 在不同网站中被工作流程所使用的处理子工作流程的信号卡
- 用于验证客户电子邮件地址、检查退出列表、发送电子邮件以及验证电子邮件未被退回或发送失败的电子邮件子工作流程。
- 将连接、设置、交易和验证进行组合的数据库存储和检索子工作流程。
- 将构建、包装和验证进行组合的源代码编译子工作流程。

在电子商务示例中，建议您将 Charge Credit Card 活动设为子工作流程。为了实现这一操作，您可以注册一个新的 Verify Customer 工作流程、注册 Verify Customer Address 和 Check Fraud DB 活动，并定义任务的协同逻辑。然后，Customer Order 工作流程中的决策者可通过排定 `StartChildWorkflowExecution` 决策来指定此工作流程类型，从而启动 Verify Customer 子工作流程。

下图所示为客户订单工作流程，其中包括一个新的 Verify Customer 子工作流程，用于检验客户地址、欺诈数据库和用信用卡付费。



多个工作流程可使用同一种工作流程类型创建子工作流程执行。例如，Verify Customer 子工作流程还能用在机构的其它部分。子工作流程的事件包含在其自己的工作流程历史中，而不包含在上层工作流程历史中。

由于子工作流程是由决策者启动的简单工作流程执行，它们还可作为正常的独立工作流程执行启动。

标记

有时，建议您将信息记录在您的使用案例之特定工作流程执行的工作流程历史中。您可以通过标记将信息记录在工作流程执行历史中，从而使您能将其用于任何自定义和适合特定情景的目的。

为了使用标记，决策者会使用 RecordMarker 决策，对标记进行命名，向决策附加所需的数据，并使用 `RespondDecisionTaskCompleted` 操作来通知 Amazon SWF。Amazon SWF 接收请求，将标记记录在工作流程历史中，并在请求中制定其他决策。从那一刻其，决策者就能在工作流程历史中查看标记并通过您编程所用任何方式使用该标记。

以下为标记示例：

- 用于对递归工作流程中的回路数进行计数的计数器。
- 根据活动结果进行的工作流程执行的进度。
- 从较早的工作流程历史事件总结的信息。

在电子商务示例中，您可以增加按日检查库存以及每次递增标记计数的活动。然后，您可以增加决策逻辑，以在计数超出五时发送电子邮件给客户或通知管理人员，无需审查整个历史记录。

标签

借助于 Amazon SWF，您可以将标签与工作流程执行相关联，随后基于这次标签来查询工作流程执行情况。您可以通过加标签在您使用可见性操作时筛选执行列表。通过仔细筛选您向执行分配的标签，您可以使用这些标签帮助提供有意义的列表。

举例来说，假设您在运行多个订单履行中心。适当加标签可使您列出特定订单履行中心内发生的流程。或者，另举一例，如果客户要转换不同类型的媒体文件，您可以通过加标签显示视频、音频和图像文件转换所用流程的差异。

Amazon SWF 支持为一个工作流程执行添加最多加五个标签。每个标签都是自由格式的字符串，长度最多为 256 个字符。如果您想使用标签，您必须在启动工作流程执行时分配标签。您不能再工作流程执行启动之后将标签增加到该执行中，您也不能编辑或删除已分配到工作流程执行中的标签。

使用 Amazon SWF API

本节介绍如何使用服务 API 在 Amazon Simple Workflow Service (Amazon SWF) 中开发工作流程。我们建议您在阅读本节内容的同时参考 *Amazon Simple Workflow Service API Reference*。本章节中的 API 调用示例采用 JavaScript Object Notation (JSON) 格式指定参数。

Topics

按类别列出 Amazon SWF 操作

本节列出 Amazon SWF 应用程序编程接口 (API) 中 Amazon SWF 操作的参考主题。其中按*功能类别* 列出这些主题。

有关操作的*字母顺序* 列表，请参阅 Amazon Simple Workflow Service API Reference。

Topics

与活动相关的操作

活动工作者使用 PollForActivityTask 获取新活动任务。工作者从 Amazon SWF 收到活动任务后即执行该任务，如果成功，则使用 RespondActivityTaskCompleted 进行响应，如果失败，则使用 RespondActivityTaskFailed 进行响应。

以下是活动工作者执行的操作。

- PollForActivityTask
- RespondActivityTaskCompleted
- RespondActivityTaskFailed
- RespondActivityTaskCanceled
- RecordActivityTaskHeartbeat

与决策者相关的操作

决策者使用 PollForDecisionTask 获取决策任务。决策者从 Amazon SWF 收到决策任务后，检查其工作流程执行历史记录并决定接下来要做什么。它调用 RespondDecisionTaskCompleted 以完成该决策任务，并提供零个或多个后续决策。

以下是由决策者执行的操作。

- PollForDecisionTask
- RespondDecisionTaskCompleted

与工作流程执行相关的操作

对工作流程可执行以下操作。

- RequestCancelWorkflowExecution
- StartWorkflowExecution
- SignalWorkflowExecution
- TerminateWorkflowExecution

与管理相关的操作

尽管可从 Amazon SWF 控制台中执行管理任务，但也可使用本节中的操作自动执行各种功能或构建您自己的管理工具。

活动管理

- RegisterActivityType
- DeprecateActivityType

工作流程管理

- RegisterWorkflowType
- DeprecateWorkflowType

域管理

通过以下这些操作，可注册和弃用 Amazon SWF 域。

- RegisterDomain
- DeprecateDomain

有关这些域管理操作的详细信息和示例，请参阅注册域 (p. 57)。

工作流程执行管理

- RequestCancelWorkflowExecution
- TerminateWorkflowExecution

可见性操作

尽管可从 Amazon SWF 控制台中执行可见性操作，但也可使用本节中的操作构建您自己的控制台或管理工具。

活动可见性

- ListActivityTypes
- DescribeActivityType

工作流程可见性

- ListWorkflowTypes
- DescribeWorkflowType

工作流程执行可见性

- DescribeWorkflowExecution
- ListOpenWorkflowExecutions
- ListClosedWorkflowExecutions
- CountOpenWorkflowExecutions
- CountClosedWorkflowExecutions
- GetWorkflowExecutionHistory

域可见性

- ListDomains
- DescribeDomain

任务列表可见性

- CountPendingActivityTasks
- CountPendingDecisionTasks

在 Amazon SWF 中创建基本工作流程

创建基本顺序工作流程涉及以下阶段。

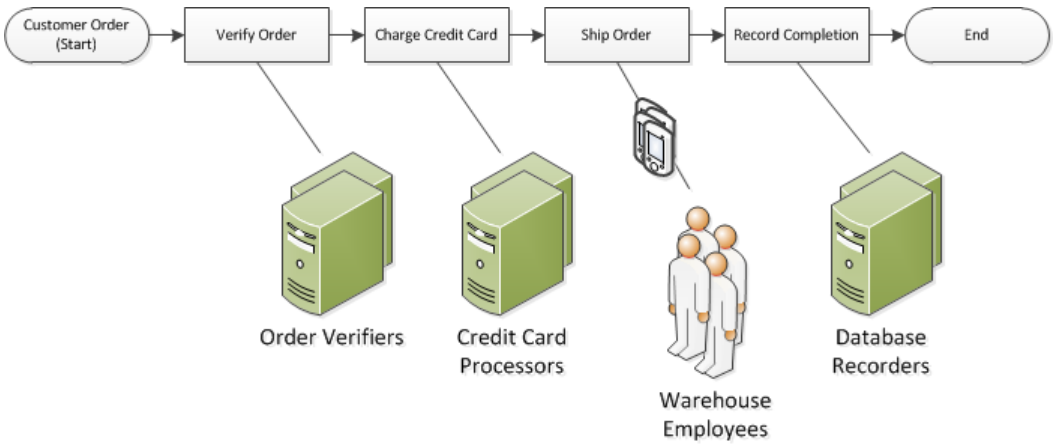
- 建立工作流程模型，注册其类型以及注册其活动类型
- 开发和启动执行活动任务的活动工作人员
- 开发和启动使用工作流程历史决定下一步操作的决策者
- 开始和启动工作流程启动者，即启动工作流程执行的应用程序

建立工作流程及其活动的模型

若要使用 Amazon SWF，请在应用程序中将逻辑步骤作为活动进行建模。活动表示一个逻辑步骤或工作流程中的一个任务。例如，授权信用卡就是一项活动，其中涉及提供信号卡号码和其它信息、接收信用卡拒绝的授权代码或消息。

除了定义活动之外，您还需要定义处理决策点的协作逻辑。例如，协作逻辑可根据信用卡是被授权还是被拒绝来排定不同的后续活动。

下图显示的是顺序客户订单工作流程的示例，共有四个活动（Verify Order、Charge Credit Card、Ship Order 和 Record Completion）。



使用 Amazon SWF 注册域

您的工作流程和活动类型以及工作流程执行本身都囊括在*域*范围之内。域将一组类型、执行和任务列表与同一账户内的其它内容隔开。

您可以使用 AWS 管理控制台或 Amazon SWF API 中的 `RegisterDomain` 操作注册域。以下为使用 API 的示例。

```
https://swf.us-east-1.amazonaws.com RegisterDomain { "name" : "867530901",
"description" : "music", "workflowExecutionRetentionPeriodInDays" : "60" }
```

参数以 JavaScript Object Notation (JSON) 格式指定。此处，保留期设置为 60 天。保存期间，使用 AWS 管理控制台或 Amazon SWF API 通过可视性操作提供所有关于工作流程执行的信息。

注册域后，您应该注册工作流程类型和该工作流程所用的活动类型。您需要首先注册域，因为注册的域名是注册工作流程和活动类型时所需要信息的一部分。

另请参阅

- *Amazon Simple Workflow Service API Reference* 中的 RegisterDomain

设置 Amazon SWF 中的超时值

Topics

超时值限制

超时值始终以秒为单位来声明，并且可设置为长达一年（31536000 秒）内的任何秒数 — 一年是任何工作流程或活动的最大执行时限。特殊值“NONE”用于将超时参数设置为“没有超时”或无限，但仍适用一年的最大限制。

工作流程执行和决策任务超时

您可以在注册工作流程类型时设置工作流和决策任务的超时值。例如：

```
https://swf.us-east-1.amazonaws.com RegisterWorkflowType { "domain": "867530901",
 "name": "customerOrderWorkflow", "version": "1.0", "description": "Handle
customer orders", "defaultTaskStartToCloseTimeout": "600", "defaultExecution
StartToCloseTimeout": "3600", "defaultTaskList": { "name": "mainTaskList" },
"defaultChildPolicy": "TERMINATE" }
```

此工作流程类型注册会将 defaultTaskStartToCLoseTimeout 设置为 600 秒（10 分钟），并将 defaultExecutionStartToCloseTimeout 设置为 3600 秒（1 小时）。

有关工作流程类型注册的更多信息，请参阅 注册工作流程类型 (p. 59) 以及 *Amazon Simple Workflow Service API Reference* 中的 RegisterWorkflowType。

您可以覆盖为 `defaultExecutionStartToCloseTimeout` 设置的值，方法是在 StartWorkflowExecution 中指定 executionStartToCloseTimeout。

活动任务超时

您可以注册活动类型时设置活动任务的超时值。例如：

```
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" }, "defaultTaskSchedu
leToStartTimeout": "1800", "defaultTaskScheduleToCloseTimeout": "5400" }
```

此活动类型注册将 defaultTaskStartToCloseTimeout 设置为 600 秒（10 分钟），将 defaultTaskHeartbeatTimeout 设置为 120 秒（2 分钟），将 defaultTaskScheduleToStartTimeout 设置为 1800 秒（30 分钟），并将 defaultTaskScheduleToCloseTimeout 设置为 5400 秒（1.5 小时）。

有关活动类型注册的更多信息，请参阅 注册活动类型 (p. 59) 以及 *Amazon Simple Workflow Service API Reference* 中的 RegisterActivityType。

您可以覆盖为 `defaultTaskStartToCloseTimeout` 设置的值，方法是在 StartWorkflowExecution 中指定 taskStartToCloseTimeout。

另请参阅

- 超时类型 (p. 38)

使用 Amazon SWF 注册工作流程类型

本章节中讨论的示例是，使用 Amazon SWF API 注册工作流程类型。您在注册过程中指定的名称和版本会形成工作流程类型唯一的标识符。指定的域必须使用 RegisterDomain API 进行了注册。

下列示例中的超时参数为持续时间值，以秒为单位。对于 `defaultTaskStartToCloseTimeout` 参数，您可以使用持续时间说明符“NONE”指示没有超时。但是，您不能指定 `defaultExecutionStartToCloseTimeout` 的值为“NONE”；工作流程执行可运行的最大时长限制是一年。超出此限制都会导致工作流程执行超时。如果将 `defaultExecutionStartToCloseTimeout` 的值指定为大于一年，注册将会失败。

```
https://swf.us-east-1.amazonaws.com RegisterWorkflowType { "domain" :
"867530901", "name" : "customerOrderWorkflow", "version" : "1.0", "description"
 : "Handle customer orders", "defaultTaskStartToCloseTimeout" : "600", "defaultEx
ecutionStartToCloseTimeout" : "3600", "defaultTaskList" : { "name": "mainTask
List" }, "defaultChildPolicy" : "TERMINATE" }
```

另请参阅

- *Amazon Simple Workflow Service API Reference* 中的 RegisterWorkflowType

使用 Amazon SWF 注册活动类型

下面的示例使用 Amazon SWF API 注册活动类型。您在注册期间指定的名称和版本形成域内活动类型的唯一标识符。指定的域必须使用 `RegisterDomain` 操作进行了注册。

本示例中的超时参数为持续时间值，以秒为单位。您可以使用持续时间说明符 "NONE" 指示没有超时。

```
https://swf.us-east-1.amazonaws.com RegisterActivityType { "domain" :
"867530901", "name" : "activityVerify", "version" : "1.0", "description" :
"Verify the customer credit", "defaultTaskStartToCloseTimeout" : "600", "default
```

```
TaskHeartbeatTimeout" : "120", "defaultTaskList" : { "name" : "mainTaskList"
}, "defaultTaskScheduleToStartTimeout" : "1800", "defaultTaskScheduleToClose
Timeout" : "5400" }
```

另请参阅

- *Amazon Simple Workflow Service API Reference* 中的 RegisterActivityType

开发 Amazon SWF 中的活动工作人员

活动工作人员会执行一个或多个活动类型。活动工作人员与 Amazon SWF 通信以接收并执行活动任务。您可以有一队多个活动工作人员执行具有同一种活动类型的活动任务。

当决策者排定活动任务时，Amazon SWF 会将活动任务提供给活动工作人员。当决策者排定活动任务时，它会提供活动工作人员执行该活动任务所需的数据（此数据是由您来确定的）。Amazon SWF 会在将活动任务发送到活动工作人员之前，将这些数据插到活动任务中。

活动工作人员由您来管理。可以用任何语言编写活动工作人员。工作人员可在任何地点运行，只要它能通过 API 与 Amazon SWF 通信。由于 Amazon SWF 提供执行活动任务所需的所有信息，因此，所有活动工作人员都可以是无状态的。无状态性使您的工作流程可以高度扩展；处理增长的容量需求并简单增加更多活动工作人员。

本章节说明了如何执行活动工作人员。活动工作人员应重复执行下列操作。

1. 轮询 Amazon SWF 以获取活动任务。
2. 开始执行任务。
3. 如果任务长时间运行，请定期向 Amazon SWF 报告检测信号。
4. 报告任务已完成或失败并将结果返回到 Amazon SWF。

Topics

轮询活动任务

若要执行活动任务，每个活动工作人员必须轮询 Amazon SWF，方法是定期调用 `PollForActivityTask` 操作。

在下面的示例中，活动工作人员 **`ChargeCreditCardWorker01`** 轮询任务列表上的任务 **`ChargeCreditCard-v0.1`**。如果没有提供活动任务，则在 60 秒后，Amazon SWF 将返回空响应。空响应是一种 `Task` 结构，其中的 `taskToken` 值为空字符串。

```
https://swf.us-east-1.amazonaws.com PollForActivityTask { "domain" : "867530901",
 "taskList" : { "name": "ChargeCreditCard-v0.1" }, "identity" : "ChargeCredit
CardWorker01" }
```

如果活动任务变为可用，则 Amazon SWF 会将其返回给活动工作人员。任务中包含决策者在其排定活动时指定的数据。

活动工作人员接收活动任务后就已作好执行工作的准备。下一章提供了执行活动任务方面的信息。

执行活动任务

接收活动任务后，活动工作人员就做好了执行任何的准备。

要执行活动任务

1. 给您的活动工作人员编程以说明任务输入字段中的内容。该字段中包含决策者在排定任务时指定的数据。
2. 给活动工作人员编程以开始处理数据并执行您的逻辑。

下一节说明了如何对活动工作人员进行编程以将长时间运行活动的状态更新提供给 Amazon SWF。

报告活动任务检测信号

如果在活动类型中注册了检测信号超时，则活动工作人员必须在超出检测信号超时之前记录检测信号。如果活动任务没有在超时内提供检测信号，则任务会超时，Amazon SWF 关闭该任务并排定新决策任务以将超时情况通知决策者。然后，决策者可重新排定活动任务或采取另一个操作。

如果在超时后活动工作人员尝试访问 Amazon SWF（例如，通过调用 `RespondActivityTaskCompleted`），则 Amazon SWF 会返回 `UnknownResource` 错误。

本章描述了提供活动检测信号的方法。

若要记录活动任务检测信号，请对活动工作人员编程以调用 `RecordActivityTaskHeartbeat` 操作。此操作还会提供字符串字段，您可以在该字段中存储自由格式的数据，从而量化以任何方式作用于您的应用程序的过程。

在此示例中，活动工作人员将检测信号报告给 Amazon SWF，并使用详细信息字段来报告该活动任务已完成 40%。要报告检测信号，活动工作人员必须指定活动任务的任务令牌。

```
https://swf.us-east-1.amazonaws.com RecordActivityTaskHeartbeat { "taskToken"
: "12342e17-80f6-FAKE-TASK-TOKEN32f0223", "details" : "40" }
```

此操作本身不会在工作流程执行历史中创建事件；但是，如果任务超时，则工作流程执行历史将包含 `ActivityTaskTimedOut` 事件，该事件包含该活动工作人员生成的最后一个检测信号的信息。

完成活动任务或活动任务失败

执行完任务后，活动工作人员应报告活动任务完成或失败。

完成活动任务

若要完成活动任务，请对活动工作人员编程以便在其成功完成活动任务后调用 `RespondActivityTaskCompleted` 操作，同时指定任务令牌。

在此示例中，活动工作人员指示任务已成功完成。

```
https://swf.us-east-1.amazonaws.com RespondActivityTaskCompleted { "taskToken":
 "12342e17-80f6-FAKE-TASK-TOKEN32f0223", "results": "40" }
```

活动完成时，Amazon SWF 会针对与该活动关联的工作流程执行排定新决策任务。

给活动工作人员编程，以在其即将完成任务后轮询另一个活动任务。此操作会创建一个循环，在该循环中，活动工作人员会继续轮询并完成任务。

如果该活动没有在 *StartToCloseTimeout* 期间产生响应，或者如果已超出 *ScheduleToCloseTimeout*，则 Amazon SWF 将该活动任务超时，并排定决策任务。这会使决策者采取适当操作，如重新排定任务。

例如，如果 Amazon EC2 实例正在执行活动任务且该实例在任务完成前失败，则决策者会将一个超时事件接收到工作流程执行历史中。如果该活动任务正在使用检测信号，则决策者会在该任务未在 Amazon EC2 实例失败后成功传输下一个检测信号时接收该事件。否则，决策者会在活动任务触及整个超时值中的一个前完成失败时最终接收事件。然后，由决策者决定重新分配任务或采取某些其它操作。

活动任务失败

如果活动工作人员出于某种原因无法执行活动任务，但仍能与 Amazon SWF 通信，则您可以对其进行编程以使该任务失败。

若要对活动工作人员编程以使某个活动任务失败，请对其进行编程以调用指定该任务的任务令牌的 `RespondActivityTaskFailed` 操作。

```
https://swf.us-east-1.amazonaws.com RespondActivityTaskFailed { "taskToken" :
"12342e17-80f6-FAKE-TASK-TOKEN32f0223", "reason" : "CC-Invalid", "details" :
"Credit Card Number Checksum Failed" }
```

作为开发人员，您要规定存储在原因和详情字段中的值。这些值是自由格式的字符串；您可以使用适用于您的应用程序的任何错误代码约定。Amazon SWF 不处理这些值。但是，Amazon SWF 可在控制台中显示这些值。

活动任务失败时，Amazon SWF 会排定与该活动任务关联的工作流程执行的决策任务，以通知决策者任务失败。给您的决策者编程以处理失败的活动，如通过重新排定活动或使工作流程执行失败，这取决于失败的性质。

启动活动工作人员

要启动活动工作人员，将您的逻辑打包成可执行，这样您就可以将其用于您的活动工作人员平台上。例如，您可以将您的活动代码打包成 Java 可执行，这样您就可以将其用于 Linux 和 Windows 服务器上。

启动后，您的工作人员即会开始轮询任务。在决策者排定活动任务之前，虽然这些轮询超时未轮询到任务，但您的工作人员只会继续轮询。

由于轮询是出站请求，活动工作人员可以在任何有权访问 Amazon SWF 终端节点的网站上运行。

您想要启动多少活动工作人员就可以启动多少。决策者排定活动任务时，Amazon SWF 会将活动任务自动分配给轮询活动工作人员。

开发 Amazon SWF 中的决策者

决策者指的是工作流程类型协作逻辑的执行，在您执行工作流程期间运行。您可以针对一种工作流程类型运行多个决策者。

由于工作流程执行的执行状态存储在其工作流程历史中，决策者可以是无状态的。Amazon SWF 会保留该工作流程历史，并随每个决策任务将其提供给一个决策者。这会使您视需要动态增加和删除决策者，从而使您的工作流程处理过程具有高度可扩展性。随着系统负载的增长，您只需增加更多决策者来处理增加的容量。但是，请注意，任何时候，对于给定的工作流程执行，都只能由一个决策任务处于开启状态。

每当工作流程执行的状态发生更改时，Amazon SWF 都会排定一个决策任务。每当决策者收到决策任务时，它都会执行以下操作：

- 解析与决策任务一并提供的工作流程执行历史
- 根据工作流程执行历史采用协作逻辑，并作出下一步操作的决策。每一个决策都由一个 Decision 结构表示
- 完成该决策任务并向 Amazon SWF 提供一个决策列表。

本章表述了开发决策者的方法，其中包括：

- 给您的决策者编程以轮询决策任务
- 给您的决策者编程以解析工作流程执行历史并作出决策
- 给您的决策者编程以响应决策任务。

本章节中的示例显示的是，您针对电子商务示例工作流程给决策者编程的方法。

您可以使用所需的任何语言来实现决策者并在任何位置来运行它，只要该决策者可通过其服务 API 与 Amazon SWF 通信。

Topics

定义协作逻辑

开发决策者的第一步是定义协作逻辑。在电子商务示例中，在前一个活动完成后排定每个活动的协作逻辑可能与以下内容相似：

```
IF lastEvent = "StartWorkflowInstance" addToDecisions ScheduleVerifyOrderActivity
 ELSIF lastEvent = "CompleteVerifyOrderActivity" addToDecisions Schedule
ChargeCreditCardActivity ELSIF lastEvent = "CompleteChargeCreditCardActivity"
addToDecisions ScheduleCompleteShipOrderActivity ELSIF lastEvent = "CompleteShi
pOrderActivity" addToDecisions ScheduleRecordOrderCompletion ELSIF lastEvent =
 "CompleteRecordOrderCompletion" addToDecisions CloseWorkflow ENDIF
```

决策者将协作逻辑应用于工作流程执行历史，并在使用 RespondDecisionTaskCompleted 操作完成决策任务时创建决策列表。

轮询决策任务

每个决策者都要轮询决策任务。决策任务中包含决策者用于生成决策（如排定活动任务）的信息。为了轮询决策任务，决策者会使用 PollForDecisionTask 操作。

在此示例中，决策者会轮询决策任务，从而指定 **customerOrderWorkflow-0.1** 任务列表。

```
https://swf.us-east-1.amazonaws.com PollForDecisionTask { "domain": "867530901",
 "taskList": {"name": "customerOrderWorkflow-v0.1"}, "identity": "Decider01",
"maximumPageSize": 500, "reverseOrder": true }
```

如果决策任务存在于指定的任务列表，则 Amazon SWF 会立即返回该任务。如果没有决策任务可用，Amazon SWF 会保持连接打开长达 60 秒，并且一旦任务可用，就会将其返回。如果没有任务可用，则 Amazon SWF 会返回一个空响应。空响应是一种 Task 结构，其中的 taskToken 值为空字符串。确保给您的决策者编程以在它收到空响应时轮询另一个任务。

如果决策任务可用，则 Amazon SWF 会返回一个包含决策任务的响应，以及工作流程执行历史的分页视图。

在此示例中，最近事件类型指的是启动的工作流程执行，输入元素中包含执行第一个任务所需的信息。

```
{ "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": "DecisionTaskSched
uled" }, { "eventId": 1, "eventTimestamp": 1326592619.474, "eventType": "Work
flowExecutionStarted", "workflowExecutionStartedEventAttributes": { "childPolicy"
 : "TERMINATE", "executionStartToCloseTimeout" : "3600", "input" : "data-used-
decider-for-first-task", "parentInitiatedEventId": 0, "tagList" : ["music pur
chase", "digital", "ricoh-the-dog"], "taskList": { "name": "specialTaskList"
}, "taskStartToCloseTimeout": "600", "workflowType": { "name": "customerOrder
Workflow", "version": "1.0" } } } ], ... }
```

收到工作流程执行历史后，决策者会对历史进行解析并根据其协作逻辑做出决策。

由于一个工作流程执行的工作流程历史事件数量可能会很大，返回的结果可能会被拆分跨页显示。若要检索随后的页面，请另外调用 PollForDecisionTask，调用中使用由初始调用返回的 *nextPageToken*。请注意，*请勿* 调用GetWorkflowExecutionHistory（使用此 *nextPageToken*）。而是要再次调用 PollForDecisionTask。

应用协作逻辑

在决策者收到决策任务后，给任务编程以解析工作流程执行历史，从而确定目前为止发生的情况。决策者应该基于这个情况生成决策列表。

在电子商务示例中，我们只关心工作流程历史中最近的事件，所有我们要定义以下逻辑。

```
IF lastEvent = "StartWorkflowInstance" addToDecisions ScheduleVerifyOrderActivity
```

```
 ELSIF lastEvent = "CompleteVerifyOrderActivity" addToDecisions Schedule
ChargeCreditCardActivity ELSIF lastEvent = "CompleteChargeCreditCardActivity"
addToDecisions ScheduleCompleteShipOrderActivity ELSIF lastEvent = "CompleteShi
pOrderActivity" addToDecisions ScheduleRecordOrderCompletion ELSIF lastEvent =
 "CompleteRecordOrderCompletion" addToDecisions CloseWorkflow ENDIF
```

如果 lastEvent 是 `CompleteVerifyOrderActivity`，则您需要将 `ScheduleChargeCreditCardActivity` 活动添加到该决策列表。

在决策者确定要做出的决策后，它可以用适当的决策来响应 Amazon SWF。

响应决策

在解释工作流程历史并生成决策列表后，决策者即做好将这些决策返回 Amazon SWF 的准备。

给您的决策者编程，以从工作流程执行历史中提取出它所需要的数据，然后创建决策以指定工作流程的下一步适当操作。决策者使用 `RespondDecisionTaskCompleted` 操作将这些决策发送回 Amazon SWF。有关可用决策类型的列表，请参见 *Amazon Simple Workflow Service API Reference*。

在电子商务示例中，当决策者回应其生成的决策集时，还会包括工作流程执行历史中的信用卡输入。然后，活动工作人员会获得其需要的信息来执行活动任务。

当工作流程执行中的所有活动均完成时，决策者会关闭工作流程执行。

```
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" :"ScheduleChargeCred
itCardActivity", "version" :"1.1" }, "activityId" :"3e2e6e55-e7c4-beef-feed-
aa815722b7be", "scheduleToCloseTimeout" :"360", "taskList" : { "name" :"CC_TASKS"
 }, "scheduleToStartTimeout" :"60", "startToCloseTimeout" :"300", "heartbeat
Timeout" :"60", "input" : "4321-0001-0002-1234: 0212 : 234" } } ] }
```

关闭工作流程执行

当决策者确定业务过程完成也就是没有更多要执行的活动时，决策者会生成关闭工作流程执行的决策。

若要关闭工作流程执行，请给您的决策者编程以解析工作流程历史中的事件，从而确定目前为止执行中发生的情况并查看工作流程执行是否应关闭。

如果工作流程成功完成，请通过调用 `RespondDecisionTaskCompleted` 并使用 `CompleteWorkflowExecution` 决策来关闭该工作流程执行。或者，您可以使用 `FailWorkflowExecution` 决策来放弃错误的执行。

在此电子商务示例中，决策者检查历史记录并根据协作逻辑将用于关闭工作流程执行的决策添加到其决策列表，然后通过一个关闭工作流程决策来启动 `RespondDecisionTaskCompleted` 操作。

Note

有些情况下关闭工作流程执行的操作会失败。例如，如果在决策者正关闭工作流程执行时收到信号，关闭决策会失败。要处理这种可能性，请确保决策者继续轮询决策任务。还要确保接收到下一个决策任务的决策者对该事件（在此情况下是一个信号）做出响应，从而阻止执行关闭。

您还可以支持取消工作流程执行。这尤其对长时间运行的工作流程有用。若要支持取消，决策者应处理历史记录中的 `WorkflowExecutionCancelRequested` 事件。此事件表示已请求取消执行。决策者应执行相应的清理操作，例如，取消进行中的活动任务以及关闭通过 `CancelWorkflowExecution` 决策调用 `RespondDecisionTaskCompleted` 操作的工作流程。

下面的示例调用 `RespondDecisionTaskCompleted` 以指定取消当前工作流程执行。

```
https://swf.us-east-1.amazonaws.com RespondDecisionTaskCompleted { "taskToken"
 : "12342e17-80f6-FAKE-TASK-TOKEN32f0223", "decisions" : [ { "decisionType":"Can
celWorkflowExecution", "CancelWorkflowExecutionAttributes":{ "Details": "Customer
 canceled order" } } ] }
```

Amazon SWF 进行检查以确保关闭和取消工作流程执行的决策是决策者发送的最后一个决策。也就是说，拥有在关闭工作流程之后还有决策的这样一组决策无效。

启动决策者

完成决策者开发之后，您就做好了启动一个或多个决策者的准备。

要启动决策者，请将您的协作逻辑打包成可执行，以使您能够将其用于您的决策者平台上。例如，您可以将您的决策者代码打包为 Java 可执行，以使您能够将其运行于 Linux 和 Windows 计算机上。

启动后，决策者应开始轮询 Amazon SWF 以获取任务。在您启动工作流程执行且 Amazon SWF 排定决策任务之前，这些轮询将会超时并获得空响应。空响应是一种 `Task` 结构，其中的 `taskToken` 值为空字符串。您的决策者应只需继续轮询。

Amazon SWF 可确保在任何时候，一个工作流程执行只有一个决策任务处于活动状态。这会防止决策冲突之类的问题。此外，Amazon SWF 还可确保只将一个决策任务分配给一个决策者，无论正在运行的决策者有多少。

如果在决策者处理另一个决策任务时有生成决策任务的情况发生，则 Amazon SWF 会对该新任务进行排队，直到当前任务完成。当前任务完成后，Amazon SWF 会使这个新的决策任务可用。另外，还将对决策任务进行批处理，在这个意义上，如果在决策者处理决策任务的同时有多个活动完成，则 Amazon SWF 只会创建一个新决策任务以应对多个任务完成。但是，每个任务完成都会收到工作流程执行历史中的一个单独事件。

由于轮询是出站请求，决策者可以在任何可以访问 Amazon SWF 终端节点的网络上运行。

为了推进工作流程执行，必须运行一个或多个决策者。您可以启动任意多个决策者。Amazon SWF 支持多个决策者对同一任务列表进行轮询。

利用 Amazon SWF 启动工作流程执行

您可以利用 `StartWorkflowExecution` 操作从任何应用程序中启动已注册工作流程类型的工作流程执行。启动执行时，您将被称为 `workflowId` 的标识符与执行相关联。`workflowId` 可以是适用于您的应用程序的任何字符串，如订单处理应用程序中的订单号。您不能将同一个 `workflowId` 用于同一个域中的多个已开启工作流程执行。例如，如果您用 `workflowId` **`Customer Order 01`** 启动两个工作流程执行，则第二个工作流程执行不会启动且请求会失败。但您可以再次利用已关闭执行的 `workflowId`。Amazon SWF 还将一个称为 `runId` 的系统生成的唯一标识符与每个工作流程执行相关联。

注册完工作流程和活动类型之后，通过调用 `StartWorkflowExecution` 操作从任何应用程序中启动已注册工作流程类型的工作流程执行。`input` 参数的值可以是启动工作流程所用应用程序指定的任何字符串。`executionStartToCloseTimeout` 是工作流程执行从启动到关闭所消耗的时长，以秒为单位。超出这个范围会导致工作流程执行超时。与 Amazon SWF 中其他一些超时参数不同的是，您不能将此超时

值指定为“NONE”；工作流程执行可运行的最大时长限制是一年。同样，*taskStartToCloseTimeout*指的是与此工作流程执行相关的决策任务在超时之前可花费的时间，以秒为单位。

```
https://swf.us-east-1.amazonaws.com StartWorkflowExecution { "domain" :
"867530901", "workflowId" : "20110927-T-1", "workflowType" : { "name" : "custom
erOrderWorkflow", "version" : "1.1" }, "taskList" : { "name" : "specialTaskList"
 }, "input" : "arbitrary-string-that-is-meaningful-to-the-workflow", "execution
StartToCloseTimeout" : "1800", "tagList" : [ "music purchase", "digital", "ricoh-
the-dog" ], "taskStartToCloseTimeout" : "1800", "childPolicy" : "TERMINATE" }
```

如果 `StartWorkflowExecution` 操作成功，Amazon SWF 会返回工作流程执行的 `runId`。`runId` 唯一标识此工作流程执行，用以将其与目前在 Amazon SWF 下运行的任何其他工作流程执行区分开来。请保存 `runId`，以便日后需要在 Amazon SWF 调用中指定此工作流程执行时使用。例如，如果您以后需要发送信号至工作流程执行中，将要用到 `runId`。

```
{"runId": "9ba33198-4b18-4792-9c15-7181fb3a8852"}
```

处理 Amazon SWF 中的错误

工作流程执行过程中会发生很多不同类型的错误。

Topics

验证错误

当发送至 Amazon SWF 的请求因格式不正确或包含无效数据而失败时，就会发生验证错误。在此情况下，请求可以是操作（如 `DescribeDomain`）或决策（如 `StartTimer`.如果该请求是一个操作，则 Amazon SWF 会在响应中返回错误代码。检查此错误代码，因为它可能会提供有关请求的哪方面导致失败的信息。例如，请求传输的一个或多个参数可能无效。有关常见错误代码的列表，请转到 *Amazon Simple Workflow Service API Reference* 中的操作主题。

如果失败的请求是一个决策，则将会在工作流程执行历史中列出适当的事件。例如，如果 `StartTimer` 决策失败，则您会在历史记录中看到 `StartTimerFailed` 事件。当决策者在 `PollForDecisionTask` 或 `GetWorkflowExecutionHistory` 的响应中接收到历史记录时，它应该检查这些事件。以下是在决策格式不适当或包含无效数据时可能会发生决策失败的事件之列表。

制定操作或决策中发生的错误

即使该请求的格式正确，也可能会在 Amazon SWF 尝试执行请求时发生错误。在这种情况下，历史中的以下事件中会有一个指示错误已发生。请查看该事件的 `reason` 字段以确定失败原因。

- CancelTimerFailed
- RequestCancelActivityTaskFailed

- RequestCancelExternalWorkflowExecutionFailed
- ScheduleActivityTaskFailed
- SignalExternalWorkflowExecutionFailed
- StartChildWorkflowExecutionFailed
- StartTimerFailed

超时

决策者、活动工作人员 和工作流程执行全部在超时时间限制范围内运行。在这种类型的错误中，任务或子工作流程超时。描述超时的事件会出现在历史记录中。决策者应通过重新排定任务或重新启动子工作流程等方法处理此事件。有关超时的更多信息，请参阅 超时类型 (p. 38)

- ActivityTaskTimedOut
- ChildWorkflowExecutionTimedOut
- DecisionTaskTimedOut
- WorkflowExecutionTimedOut

用户代码导致的错误

这种错误条件类型的示例为活动任务失败和子工作流程失败。与超时错误一样，Amazon SWF 会将相应事件添加到工作流程执行历史。决策者应合理地重新排定任务或重新启动子工作流程以处理此事件。

- ActivityTaskFailed
- ChildWorkflowExecutionFailed

与关闭工作流程执行相关的错误。

如果决策者尝试关闭具有待办决策任务的工作流程，它们还可以查看以下事件。

- FailWorkflowExecutionFailed
- CompleteWorkFlowExecutionFailed
- ContinueAsNewWorkflowExecutionFailed
- CancelWorkflowExecutionFailed

有关上面所列的任一事件的详细信息，请参阅 Amazon SWF API 参考中的历史事件。

使用 Amazon SWF 控制台

Amazon Simple Workflow Service (Amazon SWF) 控制台提供了配置、启动和管理工作流程执行的一种替代方法。

通过 Amazon SWF 控制台，可以：

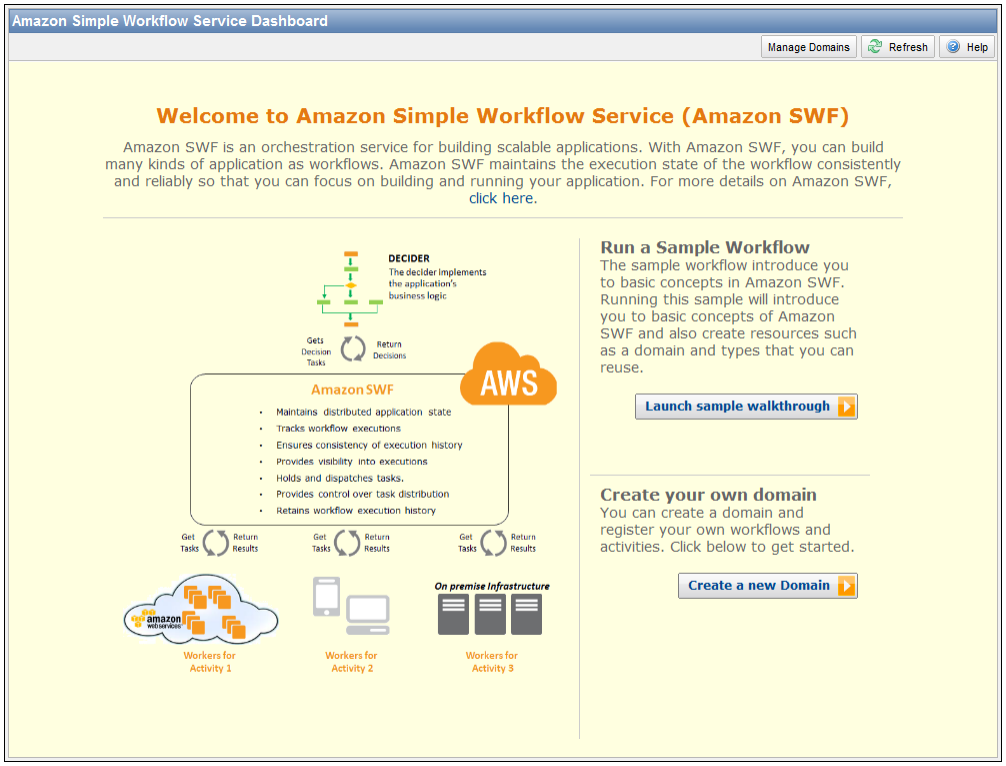
- 注册工作流程域。
- 注册工作流程类型。
- 注册活动类型。
- 启动工作流程执行。
- 查看待办任务的信息。
- 查看正在运行的工作流程执行。
- 取消、终止并发送信号至正在运行的工作流程执行中。
- 重新启动已关闭的工作流程执行。

Amazon SWF 控制台是更大型的 AWS 控制台的一部分，您可通过登录 http://aws.amazon.com 方位该控制台。登录链接位于页面右上角。

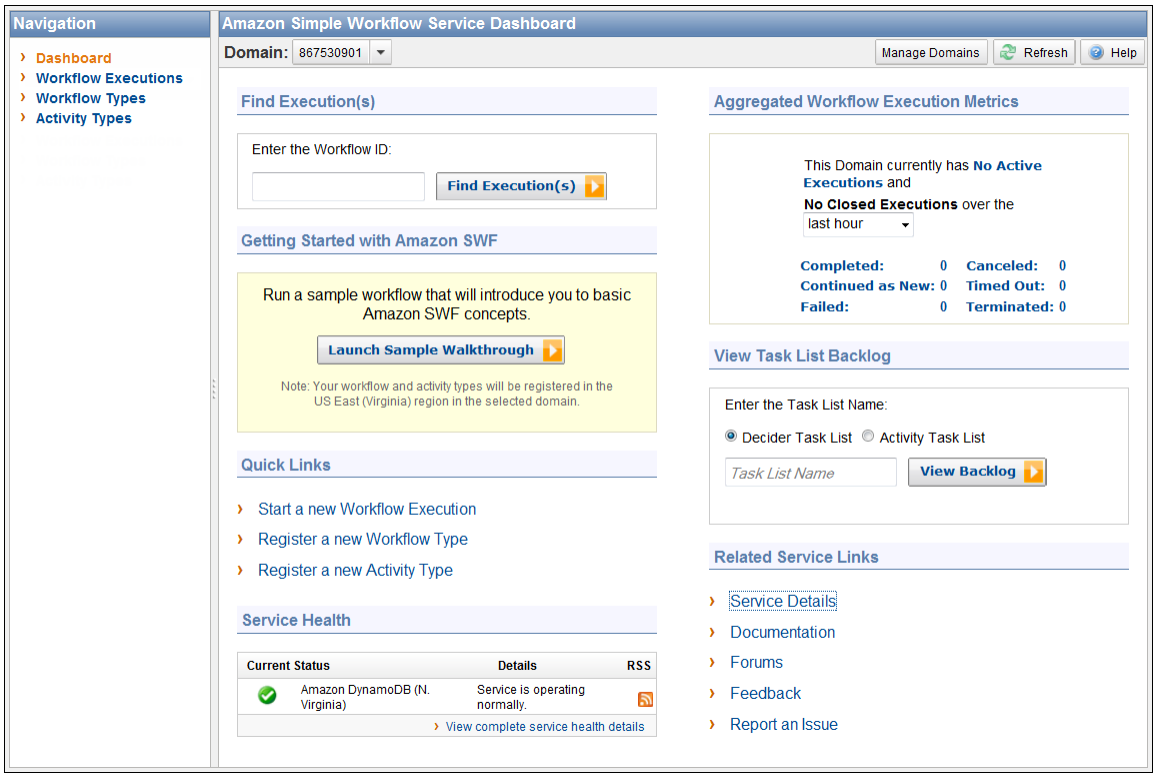
Topics

Amazon Simple Workflow Service 控制面板

下图显示了 Amazon SWF 控制台的 Amazon Simple Workflow Service 控制面板，这是没有注册域时的外观。



当注册了至少一个域时，Amazon Simple Workflow Service 控制面板将显示全部功能。



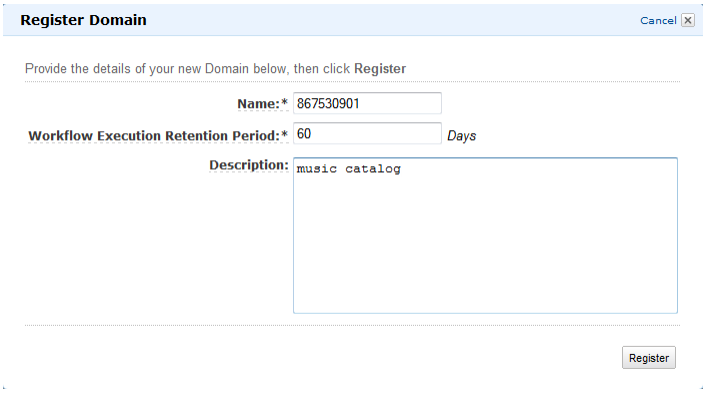
注册 Amazon SWF 域

在至少注册一个域之前，域注册是控制台提供的唯一功能。

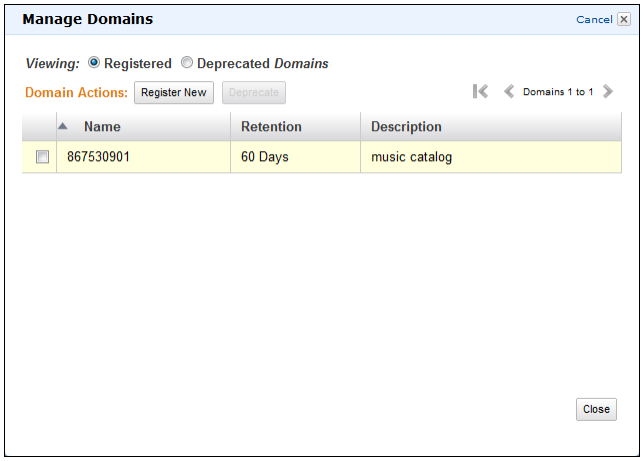
使用控制台注册 Amazon SWF 域

1. 如果没有注册域，请单击主窗格中央的注册新域。

   如果注册了至少一个域，则在控制面板视图中，单击管理域按钮，然后在管理域对话框中，单击注册新。
2. 在注册域对话框中，输入名称、保留期和描述。这些值对应于 RegisterDomain 操作中名称类似的参数。



3. 单击注册。
4. 注册域后，控制台会显示管理域对话框。



注册工作流程类型

您可以使用 Amazon Simple Workflow 控制台注册工作流程类型。在至少有一个域被注册之前，您不能注册工作流程类型。

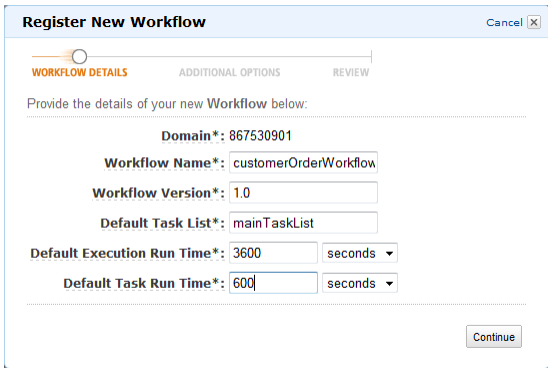
使用控制台注册 Amazon SWF 工作流程类型

1. 在Amazon Simple Workflow Service控制面板的快速链接下，单击注册新工作流程类型。

   在工作流程详细信息对话框中，输入以下信息。

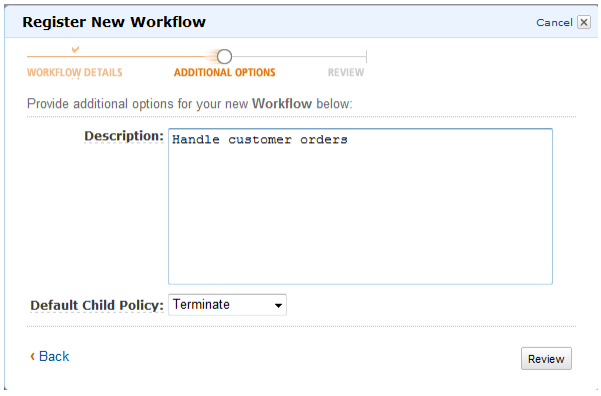
   - 域
   - 工作流程名称
   - 工作流程版本
   - 默认任务列表
   - 默认执行运行时间
   - 默认任务运行时间

   标有星号 ("*") 的字段为必填字段。

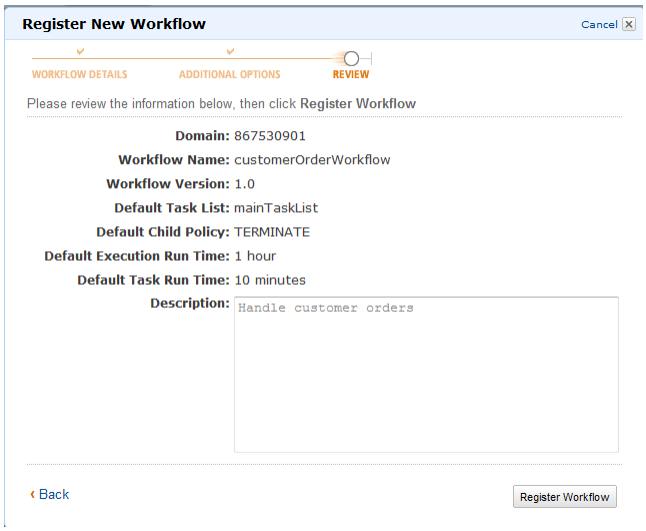


   单击继续。

2. 在 其他选项 对话框中，输入描述（描述）并指定默认子策略（默认子策略）。单击审核。



3. 在审核对话框中，检查在前面对话框中输入的信息。如果信息正确，请单击注册工作流程。否则，单击返回以更改信息。



注册活动类型

您可以使用 Amazon Simple Workflow Service 控制台来注册活动类型。在至少有一个域被注册之前，您不能注册活动类型。

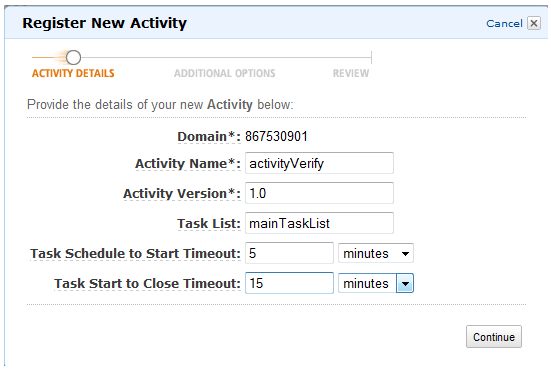
使用控制台注册 Amazon SWF 活动类型

1. 在Amazon Simple Workflow Service 控制面板的快速链接下，单击注册新活动类型。

   在活动详细信息对话框中，输入以下信息。

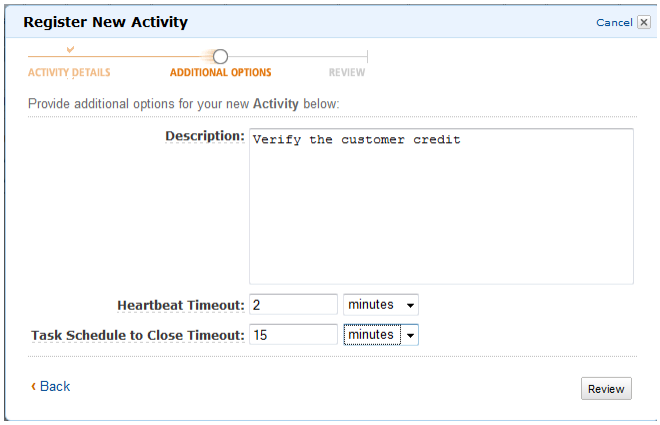
   - 域
   - 活动名称
   - 活动版本
   - 默认任务列表
   - 任务排定到启动超时
   - 任务启动到关闭超时

   标有星号 ("*") 的字段为必填字段。

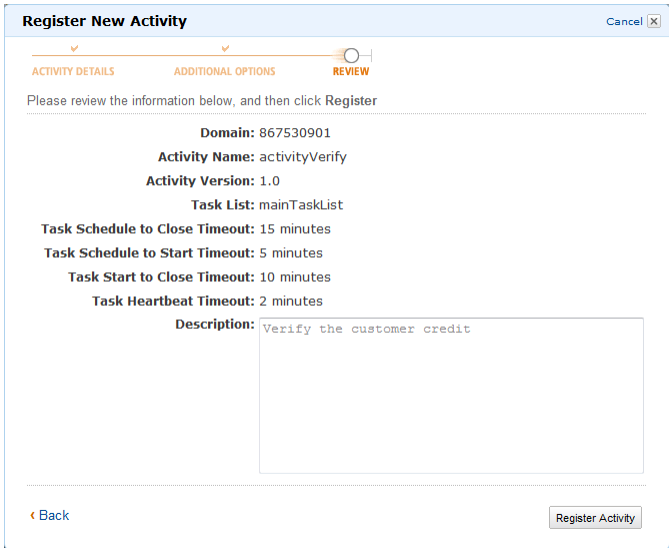


   单击继续。

2. 在其他选项对话框中，输入描述（描述）并指定心跳超时（心跳超时）和任务计划到关闭超时（任务安排到关闭超时）。单击审核。



3. 在审核对话框中，检查在前面对话框中输入的信息。如果信息正确，请单击注册活动。否则，单击返回以更改信息。



启动工作流程执行

您可以从 Amazon Simple Workflow Service 控制台来启动工作流程执行。在至少有一个工作流程类型被注册之前，您不能启动工作流程执行。

要使用控制台启动工作流程执行

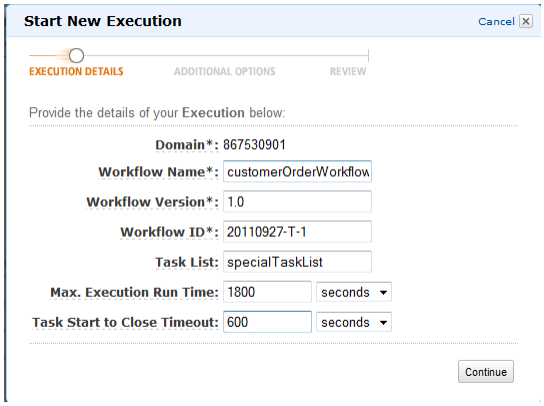
1. 在Amazon Simple Workflow Service 控制面板的快速链接下，单击启动新工作流程执行。

   在执行详细信息对话框中，输入以下信息。

   - 域
   - 工作流程名称
   - 工作流程版本

- 工作流程 ID
- 任务列表
- 最大执行运行时间
- 任务启动到关闭超时

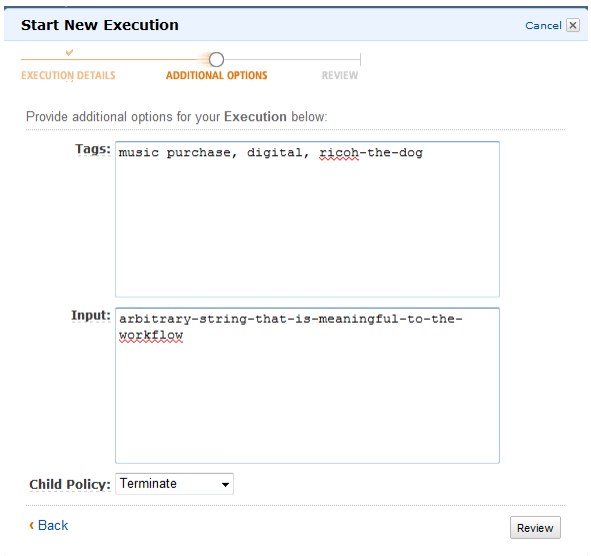
标有星号 ("*") 的字段为必填字段。



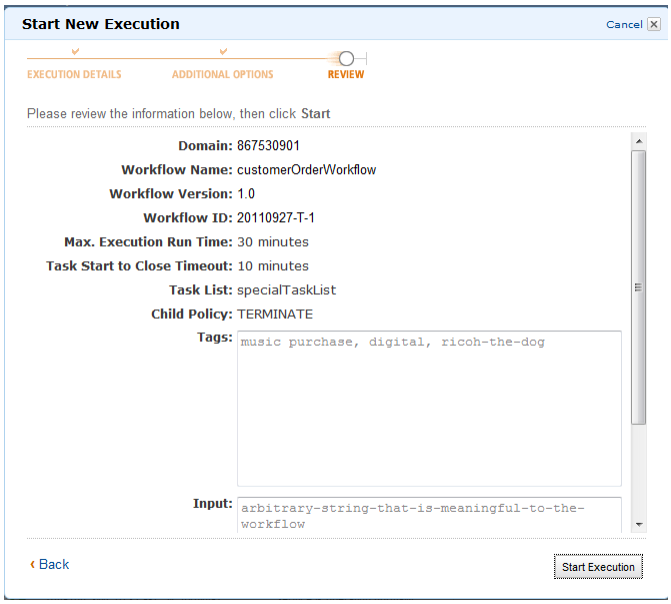
单击继续。

2. 在其他选项对话框中，指定：

   - 一组与该工作流程执行相关联的标签。您可以使用这些标签查询有关工作流程执行的信息。
   - 对该执行有意义的输入字符串。该字符串不由 Amazon SWF 解释。
   - 一个子策略。



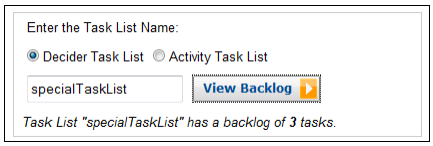
3. 在审核对话框中，检查在前面对话框中输入的信息。如果信息正确，请单击启动执行。否则，单击返回以更改信息。



查看待办任务

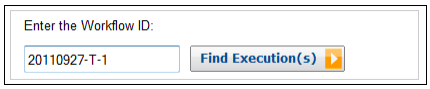
通过Amazon Simple Workflow Service 控制面板，您可以查看与特定任务列表相关的待办任务的数量。

1. 选择任务列表是决策者任务列表还是活动任务列表。
2. 在文本框中输入任务列表名称。
3. 单击查看剩余部分。



管理您的工作流程执行

Amazon SWF 控制台中的 My Workflow Executions 视图提供了用于管理正在运行以及已关闭工作流程执行的功能。若要访问此视图，请单击 Amazon SWF 控制面板中的查找执行按钮。



如果您首先输入了一个工作流程 ID，则控制台会显示带该工作流程 ID 的执行。否则，如果您只单击 查找执行，则可通过 我的工作流程执行 视图，根据工作流程执行的启动时间、它们是否仍在运行以及它们的关联元数据来查询工作流程执行。对于给定查询，您可以从以下类型的元数据中的任何一个中进行选择：

- 工作流程 ID
- 工作流程类型
- 标签

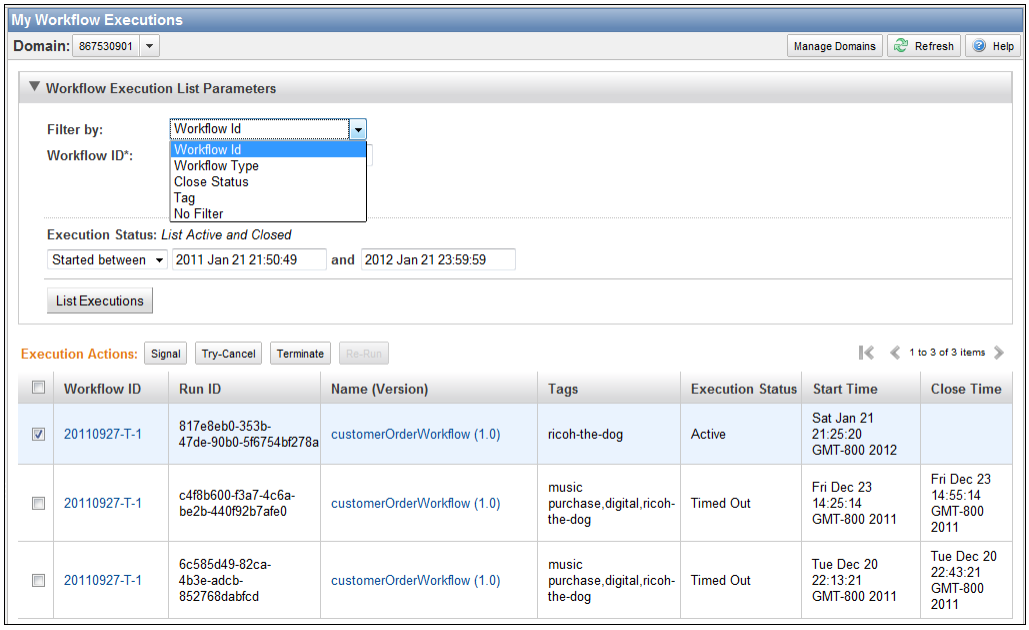
- 关闭状态

如果工作流程执行已关闭，则关闭状态是以下值中的一个，这些值指示工作流程关闭时的状态。

- 已完成
- 已失败
- 已取消
- 超时
- 作为新域继续

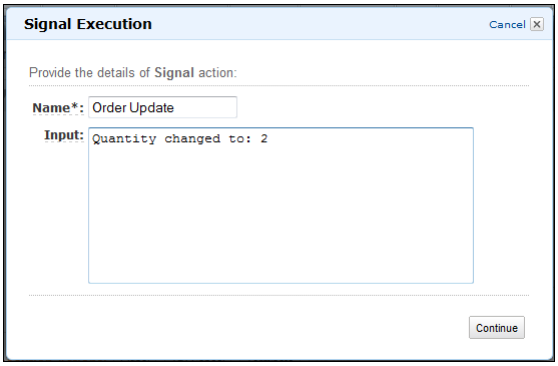
Note

在列举工作流程执行之前，您必须从域下拉列表选择域。

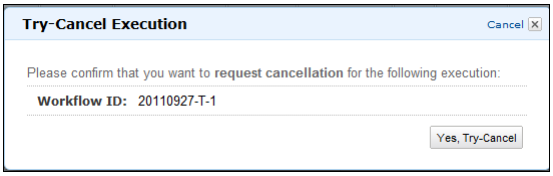


列举工作流程执行列表后，您可以执行下列操作。

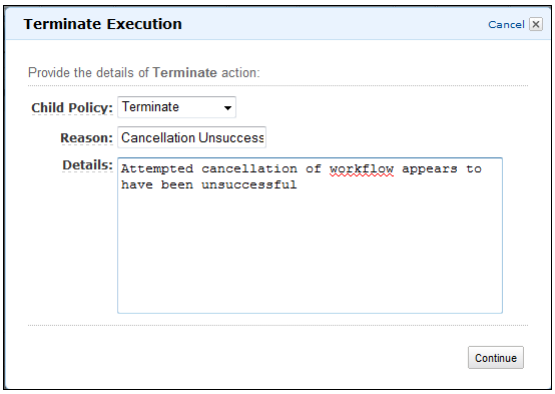
- 发送工作流程执行信号 — 即发送正在运行的工作流程附加数据。



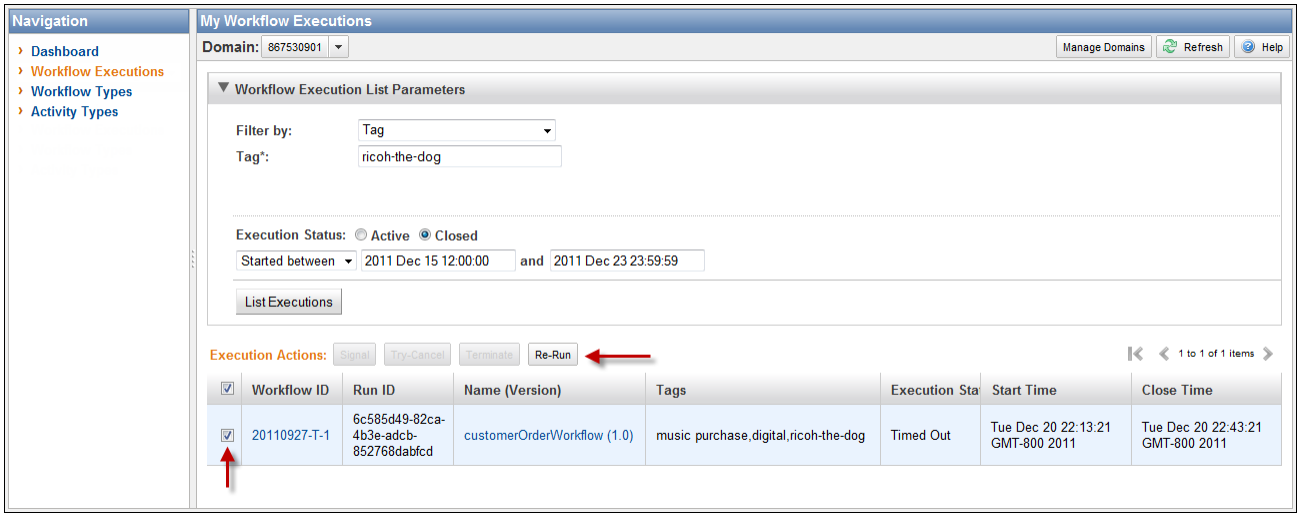
- 尝试取消工作流程执行。最好的做法是取消工作流程执行而不是终止它。取消为工作流程执行提供了执行任何清除任务以及适当关闭任务的机会。



- 终止工作流程执行。请注意，最好的做法是取消工作流程执行而不是终止它。



- 重新运行已关闭的工作流程执行。



若要重新运行已关闭的工作流程执行

1. 在工作流程执行列表中，选择已关闭的执行进行重新运行。当您选择已关闭的执行时，重新运行按钮会启用。单击重新运行。

   此时出现对话框的重新运行执行序列。
2. 在执行详细信息对话框中，请指定以下信息。对话框中已填入来自原始执行的信息。

   - 域
   - 工作流程名称

- 工作流程版本
- 工作流程 ID

通过单击高级选项链接，可指定以下附加选项。

- 任务列表
- 最大执行运行时间
- 任务启动到关闭超时

单击继续。

3. 在其他选项对话框中，指定执行的输入字符串。通过单击高级选项链接，可指定标签以便与此运行或工作流执行相关联，并可以更改执行子策略。如前述对话框一样，原始执行的信息已填入。

   单击审核。

4. 在审核对话框中，验证所有信息是否正确。如果信息正确，请单击重新运行执行。否则，单击返回以更改信息。

在 Amazon Simple Workflow Service 中使用 AWS CLI

Amazon Simple Workflow Service 的许多功能可从 AWS CLI 中访问。AWS CLI 提供一种在 AWS Management Console 中使用 Amazon SWF 或者在某些情况下使用 Amazon SWF API 和 AWS Flow Framework 进行编程的替代方法。

例如，您可以使用 AWS CLI 注册新的工作流程类型：

```
aws swf register-workflow-type --domain DataFrobtzz --name "MySimpleWorkflow"
--workflow-version "v1"
```

您还可以列出已注册的工作流程类型：

```
aws swf list-workflow-types --domain DataFrobtzz --registration-status REGISTERED
```

以下显示了 JSON 中默认输出的示例：

```
{ "typeInfos": [ { "status": "REGISTERED", "creationDate": 1377471607.752,
"workflowType": { "version": "v1", "name": "MySimpleWorkflow" } }, { "status":
 "REGISTERED", "creationDate": 1371454149.598, "description": "DataFrobtzz
subscribe workflow", "workflowType": { "version": "v3", "name": "subscribe" }
} ] }
```

AWS CLI 中的 Amazon SWF 命令能够启动和管理工作流程执行、轮询活动任务、记录任务检测信号等！有关 Amazon SWF 命令的完整列表、可用参数的描述以及显示它们使用方法的示例，请参阅 *AWS Command Line Interface Reference* 中的 Amazon SWF 命令。

AWS CLI 命令与 Amazon SWF API 密切相关，因此，您可以使用 AWS CLI 来了解基本的 Amazon SWF API。您还可以使用现有 API 知识创建代码原型，或者在命令行上执行 Amazon SWF 操作。

如需了解有关 AWS CLI 的更多信息，请参阅 AWS Command Line Interface User Guide。

使用 Amazon SWF 中的高级工作流程功能

Topics

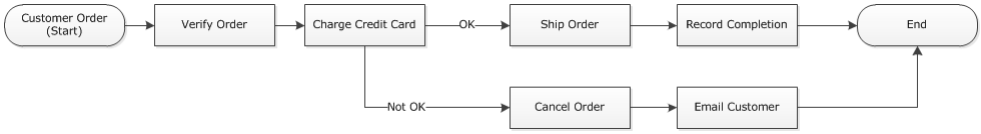
执行排他选择

在某些应用场景中，建议您根据前一个活动的结果排定不同组活动。您可以通过排他选择模式创建灵活的工作流程，以满足应用程序的复杂要求。

Amazon Simple Workflow Service (Amazon SWF) 没有特定的排他选择操作。要使用排他选择，您只需写入您的决策者逻辑以根据前一个活动的结果作出不同决策。以下为适用于排他选择的某些应用程序：

- 如果前一个活动的结果不成功，执行清除活动
- 根据客户购买的是基本还是高级计划来排定不同的活动
- 根据客户订单历史执行不同的客户身份验证活动

在电子商务示例中，您可以使用排他选择根据信用卡付费结果给订单发货或取消订单。在下图中，如果信用卡付费成功，决策者将排定 Ship Order 和 Record Completion 活动任务。否则，它将排定 Cancel Order 和 Email Customer 活动任务。



如果信用卡成功付费，则决策者将排定 `ShipOrder` 活动。否则，决策者将排定 `CancelOrder` 活动。

在这种情况下，给决策者编程以解析历史记录并决定信用卡是否付费成功。要实现此操作，您要有与下列内容相类似的逻辑

```
IF lastEvent = "WorkflowExecutionStarted" addToDecisions ScheduleActiv
ityTask(ActivityType = "VerifyOrderActivity") ELSIF lastEvent = "Activ
ityTaskCompleted" AND ActivityType = "VerifyOrderActivity" addToDecisions
ScheduleActivityTask(ActivityType = "ChargeCreditCardActivity") #Successful
Credit Card Charge Activities ELSIF lastEvent = "ActivityTaskCompleted" AND
ActivityType = "ChargeCreditCardActivity" addToDecisions ScheduleActiv
ityTask(ActivityType = "ShipOrderActivity") ELSIF lastEvent = "Activ
ityTaskCompleted" AND ActivityType = "ShipOrderActivity" addToDecisions
ScheduleActivityTask(ActivityType = "RecordOrderCompletionActivity") ELSIF
lastEvent = "ActivityTaskCompleted" AND ActivityType = "RecordOrderCompletion
Activity" 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 Activity
Type = "EmailCustomerActivity" addToDecisions CompleteWorkflowExecution ENDIF
```

如果信用卡付费成功，则决策者应通过 RespondDecisionTaskCompleted 做出响应以排定 ShipOrder 活动。

```
https://swf.us-east-1.amazonaws.com RespondDecisionTaskCompleted { "taskToken":
 "12342e17-80f6-FAKE-TASK-TOKEN32f0223", "decisions":[ { "decisionType":"Schedule
ActivityTask", "scheduleActivityTaskDecisionAttributes":{ "control":"OPTION
AL_DATA_FOR_DECIDER", "activityType":{ "name":"ShipOrder", "version":"2.4" },
"activityId":"3e2e6e55-e7c4-fee-deed-aa815722b7be", "scheduleToClose
Timeout":"3600", "taskList":{ "name":"SHIPPING" }, "scheduleToStart
Timeout":"600", "startToCloseTimeout":"3600", "heartbeatTimeout":"300", "input":
 "123 Main Street, Anytown, United States" } } ] }
```

如果信用卡付费未成功，则决策者应通过 RespondDecisionTaskCompleted 做出响应以排定 CancelOrder 活动。

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

如果 Amazon SWF 能够通过 RespondDecisionTaskCompleted 操作验证数据，则 Amazon SWF 会返回与下面类似的成功 HTTP 响应。

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

计时器

您可以使用计时器在一定量的时间过去后通知决策者。在响应决策任务时，决策者可选择通过 `StartTimer` 决策做出响应。此决策将指定时间量，在此之后，计时器应启动。指定时间过去之后，Amazon SWF 将向工作流程执行历史添加一个 `TimerFired` 事件并排定决策任务。然后，决策者将使用此信息通知进一步决策。计时器的一项常见应用为延迟活动任务的执行。例如，客户可能想延迟交付项目。

信号

信号可使您通知外部事件的工作流程执行并将信息插入正在运行的工作流程执行中。任何程序都可通过调用 `SignalWorkflowExecution` API 将信号发送到正在运行的工作流程执行。 接收到信号后，Amazon SWF 会在工作流程执行历史中将其记录为 `WorkflowExecutionSignaled` 事件，并通过排定决策任务来提醒决策者。

Note

尝试将信号发送到未打开的工作流程执行会导致带有 `UnknownResourceFault` 的 `SignalWorkflowExecution` 错误。

在这个示例中，将会发送取消订单信号到工作流程执行。

```
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"}
```

如果工作流程执行接收到信号，则 Amazon SWF 会返回与下面类似的成功 HTTP 响应。Amazon SWF 将生成一个决策任务以通知决策者处理该信号。

```
HTTP/1.1 200 OK Content-Length: 0 Content-Type: application/json x-amzn-Request
Id: bf78ae15-3f0c-11e1-9914-a356b6ea8bdf
```

有时，您可能需要等待信号。例如，用户可能会通过发送信号来取消订单，但只能在发出订单后一小时内取消。Amazon SWF 没有可以让决策者等待来自服务的信号的基元。暂停功能需要决策者自行执行。决策者应使用 `StartTimer` 决策来启动计时器以执行暂停，该计时器指定了决策者在继续轮询决策任务时等待信号的持续时间。当决策者收到决策任务时，它应该检查历史记录以查看信号是否已收到或计时器是否已启动。如果已收到信号，则决策者应取消计时器。然而，如果未收到信号，计时器已启动，则表示信号未在指定时间内到达。简而言之，请执行以下操作来等待特定信号。

1. 为决策者应等待的时间量创建计时器。
2. 收到决策任务时，检查历史记录以查看信号是否已到达或计时器是否已启动。
3. 如果信号已到达，请使用 `CancelTimer` 决策来取消计时器并处理该信号。根据具体定时情况，历史记录中可能包含 `TimerFired` 和 `WorkflowExecutionSignaled` 事件。在这种情况下，您可以依据历史记录中事件的相对顺序来确定首先发生的事件。
4. 收到信号前，如果计时器已启动，则决策者等待信号超时。您可以放弃执行或执行适合您的使用案例的其它任何逻辑。

活动任务取消

通过活动任务取消，决策者可以结束不再需要执行的活动。Amazon SWF 使用一种协作取消机制，不会强制中断运行中的活动任务。您必须给您的活动工作人员编程，以处理取消请求。

决策者可决定在活动任务正在处理决策任务时取消该活动任务。若要取消活动任务，决策者可使用带有 RequestCancelActivityTask 决策的RespondDecisionTaskCompleted 操作。

如果活动工作人员尚未获得活动任务，则服务将取消任务。请注意，存在一种潜在的竟态条件，活动工作人员可在此条件中随时获取任务。如果任务已分配给活动工作人员，则该活动工作人员将被请求取消任务。

在这个示例中，将会发送取消订单信号到工作流程执行。

```
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"}
```

如果工作流程执行接收到信号，则 Amazon SWF 会返回与下面类似的成功 HTTP 响应。Amazon SWF 将生成一个决策任务以通知决策者处理该信号。

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

当决策者处理决策任务并查看历史记录中的信号时，它会尝试取消具有 ShipOrderActivity0001 活动 ID 的未完成活动。活动 ID 在排定活动任务事件的工作流程历史中有提供。

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

如果 Amazon SWF 成功接收取消请求，则它会返回与下面类似的成功 HTTP 响应：

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

取消尝试以 ActivityTaskCancelRequested 事件的形式记录在历史中。

如果任务如 ActivityTaskCanceled 事件所示被成功取消，则对该决策者进行编程，以便在任务取消（如关闭工作流程执行）之后采取适当步骤。

如果活动任务无法取消（例如，如果任务完成、失败或超时而不能取消），则您的决策者应接受活动结果或执行您的使用案例所需的任何清除或缓解操作。

如果活动工作人员已获得活动任务，则取消请求将通过任务检测信号机制发送。活动工作人员可定期使用 RecordActivityTaskHeartbeat 向 Amazon SWF 报告任务仍在进行中。

请注意，虽然建议执行长期运行的任务，但不需要活动工作人员检测信号。任务取消需要待记录的定期检测信号；如果工作人员不检测信号，则不能取消任务。

如果决策者请求取消任务，则 Amazon SWF 会将 cancelRequest 数据元的值设置为 true。cancelRequest 数据元是由该服务在响应中返回到 RecordActivityTaskHeartbeat 的 ActivityTaskStatus 数据元的一部分。

Amazon SWF 不会阻止已请求取消的活动任务的成功完成；但取消请求的处理方式由该活动决定。根据您的要求，对活动工作人员进行编程，以取消活动任务或忽略取消请求。

如果您希望活动工作人员指示该活动任务的工作已取消，请对其进行编程以通过 RespondActivityTaskCanceled 做出响应。如果您希望活动工作人员完成任务，请对其进行编程以通过标准 RespondActivityTaskCompleted 做出响应。

当 Amazon SWF 接收 RespondActivityTaskCompleted 或 RespondActivityTaskCanceled 请求时，它会更新工作流程执行历史并排定决策任务以通知决策者。

对决策者进行编程以处理决策任务和返回任何附加决策。如果活动任务被成功取消，请对决策者进行编程以执行需要继续的任务或关闭工作流程执行。如果活动任务未成功取消，请对决策者进行编程以接受结果、忽略结果或排定任何需要的清除。

标记

您可以使用标记在工作流程执行历史中记录事件，以用于特定应用目的。标记在您想记录自定义信息以帮助执行决策者逻辑时很有用。举例来说，您可以使用标记来计数循环工作流程中的回路数量。

在下面的示例中，决策者将完成一个决策任务，并通过包含 RecordMarker 决策的 RespondDecisionTaskCompleted 操作做出响应。

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

如果 Amazon SWF 成功记录了该标记，它会返回一个与下面类似的成功 HTTP 响应。

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

记录标记本身不会启动决策任务。为了防止工作流程执行被卡住，必须执行一些继续工作流程执行的操作。举例来说，这些操作可能包括决策者排定另一个活动任务、接收信号的工作流程执行或之前排定的活动任务完成。

标记

如标签 (p. 53)一节所述，在您使用 StartWorkflowExecution 操作、StartChildWorkflowExecution 决策或 ContinueAsNewWorkflowExecution 决策开始执行时，您最多可以将五个标签与工作流程执行相关联。您可以通过加标签在您使用可视性操作列出或计数工作流程执行时筛选您的结果。

若要使用加标签

1. 修改加标签策略。考虑您的业务要求并创建一列对您有意义的标签。决定哪种执行获得哪种标签。虽然一种执行最多能分配五个标签，但您的标签库中可以有任何数量的标签。由于每种标签可以是长度最多为 256 个字符的任何字符串值，标签几乎可以描述所有的业务理念。
2. 在您创建执行时，给执行最多加上五个标签。
3. 通过为 ListOpenWorkflowExecutions、ListClosedWorkflowExecutions、CountOpenWorkflowExecutions 和 CountClosedWorkflowExecutions 操作指定 *tagFilter* 参数，可以列出带有特定标签的执行或对其计数。该操作将根据指定的标签筛选执行。

当您将标签关联到工作流程执行上时，标签会与该执行永久关联，且不能被删除。

您可以在 ListWorkflowExecutions 的 *tagFilter* 参数中仅指定一个标签。同时，标签匹配区分大小写，只有完全匹配才能返回结果。

假设您已经建立了两个加有以下标签的执行。

执行名称	分配的标签
第一次执行	消费者，2011 年 2 月
第二次执行	批发，2011 年 3 月

您可以根据消费者标签筛选 `ListOpenWorkflowExecutions` 返回的执行列表。*`oldestDate`* 和 *`latestDate`* 值是作为 Unix Time 值指定的。

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

向 Amazon SWF 发出 HTTP 请求

Topics

如果您不使用任何 AWS 开发工具包，则可以使用 POST 请求方法，通过 HTTP 来执行 Amazon Simple Workflow Service (Amazon SWF) 操作。POST 方法要求您在请求标头中指定操作并在请求正文中以 JSON 格式提供操作数据。

HTTP 标头内容

Amazon SWF 要求在 HTTP 请求标头中包含以下信息：

- *`host`* Amazon SWF 终端节点。
- *`x-amz-date`* 您必须在 HTTP *`Date`* 标头或 AWS *`x-amz-date header`*（某些 HTTP 客户端库不能设置 *`Date`* 标头）中提供时间戳。当存在 *`x-amz-date`* 标头时，系统会在对请求进行身份验证时忽略任何 *`Date`* 标头。

  必须利用以下三种格式中的一种来指定数据，如 HTTP/1.1 RFC 中所规定：
  - 格林威治时间 1994 年 11 月 6 日，星期日 08:49:37（RFC 822，由 RFC 1123 更新）
  - 格林威治时间 1994 年 11 月 6 日，星期日 08:49:37（RFC 850，由 RFC 1036 废弃）
  - 1994 年 11 月 6 日 08:49:37，星期日（ANSI C 的 asctime() 格式）
- *`x-amzn-authorization`*已签名的请求参数格式如下：

```
AWS3 AWSAccessKeyId=####,Algorithm=HmacSHA256, [,SignedHeaders=Header1;Head
er2;...] Signature=S(StringToSign)
```

  AWS3 - 这是一种与 AWS 实现相关的标签，它表示用于对该请求签名的身份验证版本（目前，对于 Amazon SWF 来说，此值始终是 *`AWS3`*）。

  AWSAccessKeyId - 您的 AWS 访问密钥 ID。

  Algorithm - 用于创建待签字符串的 HMAC-SHA 值的算法，如 *`HmacSHA256`* 或 *`HmacSHA1`*。

Signature - Base64(Algorithm(StringToSign, SigningKey))。有关详细信息，请参阅 计算 HMAC-SHA 签名值 (p. 88)

SignedHeaders - 可选。如果有这一项，其中必须包含标准化 HttpHeaders 计算中所用的所有 HTTP 标头的列表。必须用一个分号字符 (;) (ASCII 字符 59) 分隔列表值。

- *x-amz-target*：请求的目标服务和数据操作，格式如下：
  com.amazonaws.swf.service.model.SimpleWorkflowService. + *<action>*
  例如，*com.amazonaws.swf.service.model.SimpleWorkflowService.RegisterDomain*
- *content-type* 类型需要将 JSON 和字符集指定为*application/json; charset=UTF-8*

以下示例为创建域所用的 HTTP 请求的标头。

```
POST http://swf.us-east-1.amazonaws.com/ HTTP/1.1 Host: swf.us-east-1.amazon
aws.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: applica
tion/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=AKIAIOSFODNN7EX
AMPLE,Algorithm=HmacSHA256,SignedHeaders=Host;X-Amz-Date;X-Amz-Target;Content-
Encoding,Signature=tzjkF55lxAxPhzp/BRGFYQRQRq6CqrM254dTDE/EncI= Referer: ht
tp://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"}
```

此处为对应 HTTP 响应的示例。

```
HTTP/1.1 200 OK Content-Length: 0 Content-Type: application/json x-amzn-Request
Id: 4ec4ac3f-3e16-11e1-9b11-7182192d0b57
```

HTTP 正文内容

HTTP 请求的正文包含 HTTP 请求标头中指定的操作数据。使用 JSON 数据格式可以同时传递数据值和数据结构。元素可通过括号嵌套在其它元素内。例如，下面显示了一个请求，用于列出在使用 Unix 时间表示法的两个指定时间点之间启动的所有工作流程执行。

```
{ "domain": "867530901", "startTimeFilter": { "oldestDate": 1325376070,
"latestDate": 1356998399 }, "tagFilter": { "tag": "music purchase" } }
```

Amazon SWF JSON 请求和响应示例

下面的示例显示了一个对 Amazon SWF 的请求，用于获取对我们在前面创建的域的描述。然后，它显示了 Amazon SWF 响应。

HTTP POST 请求：

```
POST http://swf.us-east-1.amazonaws.com/ HTTP/1.1 Host: swf.us-east-1.amazon
aws.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: applica
```

```
tion/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=AKIAIOSFODNN7EX
AMPLE,Algorithm=HmacSHA256,SignedHeaders=Host;X-Amz-Date;X-Amz-Target;Content-
Encoding,Signature=IFJtq3M366CHqMlTpyqYqd9z0ChCoKDC5SCJBsLifu4= Referer: ht
tp://swf.us-east-1.amazonaws.com/explorer/index.html Content-Length: 21 Pragma:
 no-cache Cache-Control: no-cache {"name": "867530901"}
```

Amazon SWF 响应：

```
HTTP/1.1 200 OK Content-Length: 137 Content-Type: application/json x-amzn-Re
questId: e86a6779-3f26-11e1-9a27-0760db01a4a8 {"configuration": {"workflowExe
cutionRetentionPeriodInDays": "60"}, "domainInfo": {"description": "music",
"name": "867530901", "status": "REGISTERED"} }
```

请注意，该协议 (*HTTP/1.1*) 的后面跟有一个状态代码 (*200*)。代码值 *200* 表示操作成功。

Amazon SWF 不会对空值执行序列化操作。如果您的 JSON 分析程序设置为对请求的空值执行序列化，则 Amazon SWF 会忽略这些值。

计算 Amazon SWF 的 HMAC-SHA 签名

所需的身份验证信息

必须对发给 Amazon Simple Workflow Service (Amazon SWF) 的每个请求进行身份验证。AWS SDK 自动对请求进行签名，并根据 Amazon SWF 的需要来管理基于令牌的身份验证。但是，如果您想写入自己的 HTTP POST 请求，则需要在对您的请求进行身份验证的过程中为 HTTP POST 标头内容创建一个 *x-amzn-authorization* 值。有关对标头进行格式设置的详细信息，请参阅 HTTP 标头内容 (p. 86)。

签名过程

下面的一系列任务用于创建HMAC-SHA（基于哈希的消息验证码 - 安全哈希算法）请求签名。假设您已接收到 AWS 访问密钥（访问密钥 ID 和私有密钥）。

Note

您可以使用 SHA1 或 SHA256 方法签名。在整个签名过程中应使用同一个签名，并且签名必须与 HTTP 标头中提供的 *Algorithm* 名称匹配。

您可以执行以下任务对请求进行签名并将其提交到 Amazon SWF。

签名过程

1. 创建一个标准化的 HTTP 请求标头。HTTP 标头规范形式包括以下内容：
   - *host*
   - 以*x-amz-* 为开头的任何标头元素

   有关所包含的标头的详细信息，请参阅 HTTP 标头内容 (p. 86)。
   a. 对于每一个标头名值对，将标头名称转换为小写字母（而不是标头值）。
   b. 按照 RFC 2616 第 4.2 节的规定，构建标头名称到逗号分隔标头值的映射。

```
x-amz-example: value1 x-amz-example: value2 => x-amz-example:value1,value2
```

c. 对于每个标头名称-值对，将其转换成 `headerName:headerValue` 格式的字符串。从 `headerName` 和 `headerValue` 的开头和结尾部分剪切掉任何空白，且冒号两边不留空格。

```
x-amz-example1:value1,value2 x-amz-example2:value3
```

d. 在转换好的每一个字符串（包括最后一个字符串）后插入一个换行符 (U+000A)。

e. 按照字母顺序用标头名称给转换好的字符串集合分类。

2. 创建一个包含以下内容的 *`string-to-sign`* 值。
   - 第 1 行：HTTP 方法 (*`POST`*)，后跟一个换行符。
   - 第 2 行：请求 URI (*/*)，后跟一个换行符。
   - 第 3 行：空字符串。通常，这里是一个查询字符串，但 Amazon SWF 不使用查询字符串。跟在换行符之后
   - 第 4–*n* 行：表示您在步骤 1 中计算的规范请求标头的字符串，后跟换行符。此换行符将根据 RFC 2616 在 HTTP 请求的标头与正文之间创建一个空白行。
   - 请求正文。请不要在请求正文后插入换行符。
3. 计算 *`string-to-sign`* 值的 SHA256 或 SHA1 摘要。在整个过程中采用同一种 SHA 方法。
4. 使用您通过 GetSessionToken API 从 AWS 安全令牌服务收到的临时私有访问密钥，通过上一个步骤中的结果值的 SHA256 或 SHA1 摘要（取决于您的选择）来计算 HMAC-SHA 并对其进行 Base64 编码。有关将临时安全证书用于 Amazon SWF 和其他 Amazon Web Services 的详细信息，请转到标识和访问管理文档。

   Note

   Amazon SWF 预期 Base64 编码 HMAC-SHA 值的结尾处有一个等号 (=)。如果您的 Base64 编码程序不包括附加等号，请在结尾附加一个等号。

5. 放置所得的值以作为发给 Amazon SWF 的 HTTP 请求的 *`x-amzn-authorization`* 标头中 *`Signature`* 名称的值。
6. Amazon SWF 验证该请求并执行指定操作。

关于适用于 AWS 版本 3 签名的 Java 实现的 AWS 开发工具包，请参阅 AWSSigner.java 类。

文档历史记录

下表说明了自 *Amazon Simple Workflow Service 开发人员指南*上次发布以来对该文档所做的重要更改。

- API 版本：2012-01-25
- 文档最近更新时间：2013 年 10 月 25 日

变更	描述	修改日期
更新	添加了工作流程教程。请参阅 教程：Amazon SWF 和 Amazon SNS 的订阅工作流程 (p. 7)。	2013 年 10 月 25 日
更新	添加了 AWS CLI 信息和示例 (p. 80)。	2013 年 8 月 26 日
更新	更新和补丁。	2013 年 8 月 1 日
更新	更新了该文档以说明如何使用 IAM 进行访问控制。	2013 年 2 月 22 日
首次发行	这是 *Amazon Simple Workflow Service 开发人员指南*的首次发行。	2012 年 10 月 16 日