Amazon SWF FAQs

### **General**

Q: What is Amazon SWF?  
Amazon Simple Workflow Service (SWF) is a web service that makes it easy to coordinate work across distributed application components. Amazon SWF enables applications for a range of use cases, including media processing, web application back-ends, business process workflows, and analytics pipelines, to be designed as a coordination of tasks. Tasks represent invocations of various processing steps in an application which can be performed by executable code, web service calls, human actions, and scripts.

The coordination of tasks involves managing execution dependencies, scheduling, and concurrency in accordance with the logical flow of the application. With Amazon SWF, developers get full control over implementing processing steps and coordinating the tasks that drive them, without worrying about underlying complexities such as tracking their progress and keeping their state. Amazon SWF also provides the AWS Flow Framework to help developers use asynchronous programming in the development of their applications. By using Amazon SWF, developers benefit from ease of programming and have the ability to improve their applications’ resource usage, latencies, and throughputs.

#### **Get Started with AWS for Free**

[Create a Free Account](https://portal.aws.amazon.com/gp/aws/developer/registration/index.html)  
[Or Sign In to the Console](https://console.aws.amazon.com/elasticache/home)

AWS Free Tier includes 750hrs of Micro Cache Node with Amazon ElastiCache.

[View AWS Free Tier Details »](https://aws.amazon.com/free/)

Q: What are the benefits of designing my application as a coordination of tasks? How does Amazon SWF help me with this?  
In Amazon SWF, tasks represent invocations of logical steps in applications. Tasks are processed by workers which are programs that interact with Amazon SWF to get tasks, process them, and return their results. A worker implements an application processing step. You can build workers in different programming languages and even reuse existing components to quickly create the worker. For example, you can use cloud services, enterprise applications, legacy systems, and even simple scripts to implement workers. By independently controlling the number of workers for processing each type of task, you can control the throughput of your application efficiently.

To coordinate the application execution across workers, you write a program called the decider in your choice of programming language. The separation of processing steps and their coordination makes it possible to manage your application in a controlled manner and give you the flexibility to deploy, run, scale and update them independently. You can choose to deploy workers and deciders either in [the cloud](https://aws.amazon.com/what-is-cloud-computing/) (e.g. Amazon EC2 or Lambda) or on machines behind corporate firewalls. Because of the decoupling of workers and deciders, your business logic can be dynamic and you application can be quickly updated to accommodate new requirements. For example, you can remove, skip, or retry tasks and create new application flows simply by changing the decider.

By implementing workers and deciders, you focus on your differentiated application logic as it pertains to performing the actual processing steps and coordinating them. Amazon SWF handles the underlying details such as storing tasks until they can be assigned, monitoring assigned tasks, and providing consistent information on their completion. Amazon SWF also provides ongoing visibility at the level of each task through APIs and a console.

Q: What can I do with Amazon SWF?  
Amazon SWF can be used to address many challenges that arise while building applications with distributed components. For example, you can use Amazon SWF and the accompanying AWS Flow Framework for:

* Writing your applications as asynchronous programs using simple programming constructs that abstract details such as initiating tasks to run remotely and tracking the program’s runtime state.
* Maintaining your application’s execution state (e.g. which steps have completed, which ones are running, etc.). You do not have to use databases, custom systems, or ad hoc solutions to keep execution state.
* Communicating and managing the flow of work between your application components. With Amazon SWF, you do not need to design a messaging protocol or worry about lost and duplicated tasks.
* Centralizing the coordination of steps in your application. Your coordination logic does not have to be scattered across different components, but can be encapsulated in a single program.
* Integrating a range of programs and components, including legacy systems and 3rd party cloud services, into your applications. By allowing your application flexibility in where and in what combination the application components are deployed, Amazon SWF helps you gradually migrate application components from private data centers to public cloud infrastructure without disrupting the application availability or performance.
* Automating workflows that include long-running human tasks (e.g. approvals, reviews, investigations, etc.) Amazon SWF reliably tracks the status of processing steps that run up to several days or months.
* Building an application layer on top of Amazon SWF to support domain specific languages for your end users. Since Amazon SWF gives you full flexibility in choosing your programming language, you can conveniently build interpreters for specialized languages (e.g. XPDL) and customized user-interfaces including modeling tools.
* Getting detailed audit trails and visibility into all running instances of your applications. You can also incorporate visibility capabilities provided by Amazon SWF into your own user interfaces using the APIs provided by Amazon SWF.

Customers have used Amazon SWF to build applications for video encoding, social commerce, infrastructure provisioning, MapReduce pipelines, business process management, and several other use cases. For more details on use cases, please see What are some use cases that can be solved with SWF?. To see how customers are using Amazon SWF today, please read our case studies.

Q: What are the benefits of Amazon SWF vs. homegrown solutions and existing workflow products?  
When building solutions to coordinate tasks in a distributed environment, developers have to account for several variables. Tasks that drive processing steps can be long-running and may fail, timeout, or require restarts. They often complete with varying throughputs and latencies. Tracking and visualizing tasks in all these cases is not only challenging, but is also undifferentiated work. As applications and tasks scale up, developers face difficult distributed systems’ problems. For example, they must ensure that a task is assigned only once and that its outcome is tracked reliably through unexpected failures and outages. By using Amazon SWF, developers can focus on their differentiated application logic, i.e. how to process tasks and how to coordinate them.

Existing workflow products often force developers to learn specialized languages, host expensive databases, and give up control over task execution. The specialized languages make it difficult to express complex applications and are not flexible enough for effecting changes quickly. Amazon SWF, on the other hand, is a cloud-based service, allows common programming languages to be used, and lets developers control where tasks are processed. By adopting a loosely coupled model for distributed applications, Amazon SWF enables changes to be made in an agile manner.

Q: What are workers and deciders?  
In Amazon SWF, an application is implemented by building workers and a decider which communicate directly with the service. Workers are programs that interact with Amazon SWF to get tasks, process received tasks, and return the results. The decider is a program that controls the coordination of tasks, i.e. their ordering, concurrency, and scheduling according to the application logic. The workers and the decider can run on cloud infrastructure, such as Amazon EC2, or on machines behind firewalls. Amazon SWF brokers the interactions between workers and the decider. It allows the decider to get consistent views into the progress of tasks and to initiate new tasks in an ongoing manner. At the same time, Amazon SWF stores tasks, assigns them to workers when they are ready, and monitors their progress. It ensures that a task is assigned only once and is never duplicated. Since Amazon SWF maintains the application’s state durably, workers and deciders don’t have to keep track of execution state. They can run independently, and scale quickly. Please see Functionality section of the Amazon SWF detail page to learn more about the steps in building applications with Amazon SWF.

You can have several concurrent runs of a workflow on Amazon SWF. Each run is referred to as a workflow execution or an execution. Executions are identified with unique names. You use the Amazon SWF Management Console (or the visibility APIs) to view your executions as a whole and to drill down on a given execution to see task-level details.

Q: What programming conveniences does Amazon SWF provide to write applications?  
Like other AWS services, Amazon SWF provides a core SDK for the web service APIs. Additionally, Amazon SWF offers an SDK called the AWS Flow Framework that enables you to develop Amazon SWF-based applications quickly and easily. AWS Flow Framework abstracts the details of task-level coordination with familiar programming constructs. While running your program, the framework makes calls to Amazon SWF, tracks your program’s execution state using the execution history kept by Amazon SWF, and invokes the relevant portions of your code at the right times. By offering an intuitive programming framework to access Amazon SWF, AWS Flow Framework enables developers to write entire applications as asynchronous interactions structured in a workflow. For more details, please see What is the AWS Flow Framework?

Q: When should I use Amazon SWF vs. AWS Step Functions?

AWS Step Functions is a fully managed service that makes it easy to coordinate the components of distributed applications and microservices using visual workflows. Instead of writing a Decider program, you define state machines in JSON. AWS customers should consider using Step Functions for new applications. If Step Functions does not fit your needs, then you should consider Amazon Simple Workflow (SWF). Amazon SWF provides you complete control over your orchestration logic, but increases the complexity of developing applications. You may write decider programs in the programming language of your choice, or you may use the Flow framework to use programming constructs that structure asynchronous interactions for you. AWS will continue to provide the Amazon SWF service, Flow framework, and support all Amazon SWF customers.

Q: How is Amazon SWF different from Amazon SQS?

Both Amazon SQS and Amazon SWF are services that facilitate the integration of applications or microservices:

* Amazon Simple Queue Service (Amazon SQS) offers reliable, highly-scalable hosted queues for storing messages while they travel between applications or microservices. Amazon SQS lets you move data between distributed application components and helps you decouple these components.
* Amazon Simple Workflow Service (Amazon SWF) is a web service that makes it easy to coordinate work across distributed application components.

The following are the main differences between Amazon SQS and Amazon SWF:

* Amazon SWF API actions are task-oriented. Amazon SQS API actions are message-oriented.
* Amazon SWF keeps track of all tasks and events in an application. Amazon SQS requires you to implement your own application-level tracking, especially if your application uses multiple queues.
* The Amazon SWF Console and visibility APIs provide an application-centric view that lets you search for executions, drill down into an execution’s details, and administer executions. Amazon SQS requires implementing such additional functionality.
* Amazon SWF offers several features that facilitate application development, such as passing data between tasks, signaling, and flexibility in distributing tasks. Amazon SQS requires you to implement some application-level functionality.
* In addition to a core SDK that calls service APIs, Amazon SWF provides the AWS Flow Framework with which you can write distributed applications using programming constructs that structure asynchronous interactions.

While you can use Amazon SQS to build basic workflows to coordinate your distributed application, you can get this facility out-of-the-box with Amazon SWF, alongside other application-level capabilities.

We recommend trying both Amazon SQS and Amazon SWF to determine which solution best fits your needs.

Q: What are some use cases that can be solved with Amazon SWF?

Amazon SWF has been applied to use cases in media processing, business process automation, data analytics, migration to the cloud, and batch processing. Some examples are:

Use case #1: Video encoding using Amazon S3 and Amazon EC2. In this use case, large videos are uploaded to Amazon S3 in chunks. The upload of chunks has to be monitored. After a chunk is uploaded, it is encoded by downloading it to an Amazon EC2 instance. The encoded chunk is stored to another Amazon S3 location. After all of the chunks have been encoded in this manner, they are combined into a complete encoded file which is stored back in its entirety to Amazon S3. Failures could occur during this process due to one or more chunks encountering encoding errors. Such failures need to be detected and handled.

With Amazon SWF: The entire application is built as a workflow where each video file is handled as one workflow execution. The tasks that are processed by different workers are: upload a chunk to Amazon S3, download a chunk from Amazon S3 to an Amazon EC2 instance and encode it, store a chunk back to Amazon S3, combine multiple chunks into a single file, and upload a complete file to Amazon S3. The decider initiates concurrent tasks to exploit the parallelism in the use case. It initiates a task to encode an uploaded chunk without waiting for other chunks to be uploaded. If a task for a chunk fails, the decider re-runs it for that chunk only. The application state kept by Amazon SWF helps the decider control the workflow. For example, the decider uses it to detect when all chunks have been encoded and to extract their Amazon S3 locations so that they can be combined. The execution’s progress is continuously tracked in the Amazon SWF Management Console. If there are failures, the specific tasks that failed are identified and used to pinpoint the failed chunks.

Use case #2: Processing large product catalogs using Amazon Mechanical Turk. While validating data in large catalogs, the products in the catalog are processed in batches. Different batches can be processed concurrently. For each batch, the product data is extracted from servers in the datacenter and transformed into CSV (Comma Separated Values) files required by Amazon Mechanical Turk’s Requester User Interface (RUI). The CSV is uploaded to populate and run the HITs (Human Intelligence Tasks). When HITs complete, the resulting CSV file is reverse transformed to get the data back into the original format. The results are then assessed and Amazon Mechanical Turk workers are paid for acceptable results. Failures are weeded out and reprocessed, while the acceptable HIT results are used to update the catalog. As batches are processed, the system needs to track the quality of the Amazon Mechanical Turk workers and adjust the payments accordingly. Failed HITs are re-batched and sent through the pipeline again.

With Amazon SWF: The use case above is implemented as a set of workflows. A BatchProcess workflow handles the processing for a single batch. It has workers that extract the data, transform it and send it through Amazon Mechanical Turk. The BatchProcess workflow outputs the acceptable HITs and the failed ones. This is used as the input for three other workflows: MTurkManager, UpdateCatalogWorkflow, and RerunProducts. The MTurkManager workflow makes payments for acceptable HITs, responds to the human workers who produced failed HITs, and updates its own database for tracking results quality. The UpdateCatalogWorkflow updates the master catalog based on acceptable HITs. The RerunProducts workflow waits until there is a large enough batch of products with failed HITs. It then creates a batch and sends it back to the BatchProcess workflow. The entire end-to-end catalog processing is performed by a CleanupCatalog workflow that initiates child executions of the above workflows. Having a system of well-defined workflows enables this use case to be architected, audited, and run systematically for catalogs with several million products.

Use case #3: Migrating components from the datacenter to the cloud. Business critical operations are hosted in a private datacenter but need to be moved entirely to the cloud without causing disruptions.

With Amazon SWF: Amazon SWF-based applications can combine workers that wrap components running in the datacenter with workers that run in the cloud. To transition a datacenter worker seamlessly, new workers of the same type are first deployed in the cloud. The workers in the datacenter continue to run as usual, along with the new cloud-based workers. The cloud-based workers are tested and validated by routing a portion of the load through them. During this testing, the application is not disrupted because the workers in the datacenter continue to run. After successful testing, the workers in the datacenter are gradually stopped and those in the cloud are scaled up, so that the workers are eventually run entirely in the cloud. This process can be repeated for all other workers in the datacenter so that the application moves entirely to the cloud. If for some business reason, certain processing steps must continue to be performed in the private data center, those workers can continue to run in the private data center and still participate in the application.

See our case studies for more exciting applications and systems that developers and enterprises are building with Amazon SWF.

Q: Does Amazon use Amazon SWF for its own applications?  
Yes. Developers within Amazon use Amazon SWF for a wide variety of projects and run millions of workflow executions every day. Their use cases include key business processes behind the Amazon.com and AWS web sites, implementations for several AWS web services and their APIs, MapReduce analytics for operational decision making, and management of user-facing content such as web pages, videos and Kindle books.

### **Getting Started**

Q: How can I get started with Amazon SWF?   
To sign up for Amazon SWF, go to the Amazon SWF detail page and click the “Sign Up Now” button. If you do not have an Amazon Web Service account, you will be prompted to create one. After signing up, you can run a sample walkthrough in the AWS Management Console which takes you through the steps of running a simple image conversion application with Amazon SWF. You can also download the AWS Flow Framework samples to learn about the various features of the service. To start using Amazon SWF in your applications, please refer to the Amazon SWF documentation.

Q: Are there sample workflows that I can use to try out Amazon SWF?   
Yes. When you get started with Amazon SWF, you can try the sample walkthrough in the AWS Management Console which takes you through registering a domain and types, deploying workers and deciders and starting workflow executions. You can download the code for the workers and deciders used in this walkthrough, run them on your infrastructure and even modify them to build your own applications. You can also download the AWS Flow Framework samples, which illustrate the use of Amazon SWF for various use cases such as distributed data processing, Cron jobs and application stack deployment. By looking at the included source code, you can learn more about the features of Amazon SWF and how to use the AWS Flow Framework to build your distributed applications.

Q: What are the different ways to access SWF?   
You can access SWF in any of the following ways:

* AWS SDK for Java, Ruby, .NET, and PHP
* AWS Flow Framework for Java (Included in the AWS SDK for Java)
* Amazon SWF web service APIs
* AWS Management Console

[AWS FREE TIER](https://aws.amazon.com/free/)

[Gain free, hands-on experience with AWS for 12 months](https://aws.amazon.com/free/)

[AWS MARKETPLACE](https://aws.amazon.com/marketplace/cp/freetrial/ref=mkt_ste_ha_2up_p)

[Launch popular software on AWS in minutes](https://aws.amazon.com/marketplace/cp/freetrial/ref=mkt_ste_ha_2up_p)

### **Functionality**

Q: What is registration?  
Registration is a one-time step that you perform for each different types of workflows and activities. You can register either programmatically or through the Amazon SWF Management Console. During registration, you provide unique type-ids for each activity and workflow type. You also provide default information that is used while running a workflow, such as timeout values and task distribution parameters.

Q: What are domains?  
In SWF, you define logical containers called domains for your application resources. Domains can only be created at the level of your AWS account and may not be nested. A domain can have any user-defined name. Each application resource, such as a workflow type, an activity type, or an execution, belongs to exactly one domain. During registration, you specify the domain under which a workflow or activity type should be registered. When you start an execution, it is automatically created in the same domain as its workflow type. The uniqueness of resource identifiers (e.g. type-ids, execution ID) is scoped to a domain, i.e. you may reuse identifiers across different domains.

Q: How can I manage my application resources across different environments and groupings?  
You can use domains to organize your application resources so that they are easier to manage and do not inadvertently affect each other. For example, you can create different domains for your development, test, and production environments, and create the appropriate resources in each of them. Although you may register the same workflow type in each of these domains, it will be treated as a separate resource in each domain. You can change its settings in the development domain or administer executions in the test domain, without affecting the corresponding resources in the production domain.

Q: How does a decider coordinate a workflow in Amazon SWF?  
The decider can be viewed as a special type of worker. Like workers, it can be written in any language and asks Amazon SWF for tasks. However, it handles special tasks called decision tasks. Amazon SWF issues decision tasks whenever a workflow execution has transitions such as an activity task completing or timing out. A decision task contains information on the inputs, outputs, and current state of previously initiated activity tasks. Your decider uses this data to decide the next steps, including any new activity tasks, and returns those to Amazon SWF. Amazon SWF in turn enacts these decisions, initiating new activity tasks where appropriate and monitoring them. By responding to decision tasks in an ongoing manner, the decider controls the order, timing, and concurrency of activity tasks and consequently the execution of processing steps in the application. SWF issues the first decision task when an execution starts. From there on, Amazon SWF enacts the decisions made by your decider to drive your execution. The execution continues until your decider makes a decision to complete it.

To help the decider in making decisions, SWF maintains an ongoing record on the details of all tasks in an execution. This record is called the history and is unique to each execution. A new history is initiated when an execution begins. At that time, the history contains initial information such as the execution’s input data. Later, as workers process activity tasks, Amazon SWF updates the history with their input and output data, and their latest state. When a decider gets a decision task, it can inspect the execution’s history. Amazon SWF ensures that the history accurately reflects the execution state at the time the decision task is issued. Thus, the decider can use the history to determine what has occurred in the execution and decide the appropriate next steps.

Q: How do I ensure that a worker or decider only gets tasks that it understands?  
You use task lists to determine how tasks are assigned. Task lists are Amazon SWF resources into which initiated tasks are added and from which tasks are requested. Task lists are identified by user-defined names. A task list may have tasks of different type-ids, but they must all be either activity tasks or decision tasks. During registration, you specify a default task list for each activity and workflow type. Amazon SWF also lets you create task lists at run time. You create a task list simply by naming it and starting to use it. You use task lists as follows:

* While initiating an activity task, a decider can add it into a specific task list or request Amazon SWF to add it into the default task list for its activity type.
* While starting an execution, you can request Amazon SWF to add all of its decision tasks to a specific task list or to the default task list for the workflow type.
* While requesting tasks, deciders and workers specify which task list they want to receive tasks from. If a task is available in the list, SWF sends it in the response and also includes its type-id.

Based on the above, you control which task list a task gets added into and who asks for tasks from each list. Thus, you can ensure that workers and deciders only get the tasks that they understand.

Q: What is the AWS Flow Framework? How does it help me with coordinating my workflow?  
AWS Flow Framework is a programming framework that enables you to develop Amazon SWF-based applications quickly and easily. It abstracts the details of task-level coordination and asynchronous interaction with simple programming constructs. Coordinating workflows in Amazon SWF involves initiating remote actions that take variable times to complete (e.g. activity tasks) and implementing the dependencies between them correctly.

AWS Flow Framework makes it convenient to express both facets of coordination through familiar programming concepts. For example, initiating an activity task is as simple as making a call to a method. AWS Flow Framework automatically translates the call into a decision to initiate the activity task and lets Amazon SWF assign the task to a worker, monitor it, and report back on its completion. The framework makes the outcome of the task, including its output data, available to you in the code as the return values from the method call. To express the dependency on a task, you simply use the return values in your code, as you would for typical method calls. The framework’s runtime will automatically wait for the task to complete and continue your execution only when the results are available. Behind the scenes, the framework’s runtime receives worker and decision tasks from Amazon SWF, invokes the relevant methods in your program at the right times, and formulates decisions to send back to Amazon SWF. By offering access to Amazon SWF through an intuitive programming framework, the AWS Flow Framework makes it possible to easily incorporate asynchronous and event driven programming in the development of your applications.

Q: How do workers and deciders communicate with Amazon SWF? Isn’t a poll protocol resource-intensive?  
Typically poll based protocols require developers to find an optimal polling frequency. If developers poll too often, it is possible that many of the polls will be returned with empty results. This leads to a situation where much of the application and network resources are spent on polling without any meaningful outcome to drive the execution forward. If developers don’t poll often enough, then messages may be held for longer increasing application latencies.

To overcome the inefficiencies inherent in polling, Amazon SWF provides long-polling. Long-polling significantly reduces the number of polls that return without any tasks. When workers and deciders poll Amazon SWF for tasks, the connection is retained for a minute if no task is available. If a task does become available during that period, it is returned in response to the long-poll request. By retaining the connection for a period of time, additional polls that would also return empty during that period are avoided. With long-polling, your applications benefit with the security and flow control advantages of polling without sacrificing the latency and efficiency benefits offered by push-based web services.

Q: Can I use an existing web service as a worker?  
Workers use standard HTTP GET requests to get tasks from Amazon SWF and to return the results. To use an existing web service as a worker, you can write a wrapper that gets tasks from Amazon SWF, invokes your web service’s APIs as appropriate, and returns the results back to Amazon SWF. In the wrapper, you translate input data provided in a task into the parameters for your web service’s API. Similarly, you also translate the output data from the web service APIs into results for the task and return those to Amazon SWF.

Q: Does Amazon SWF restrict me to use specific programming languages?  
No, you can use any programming language to write a worker or a decider, as long as you can communicate with Amazon SWF using web service APIs. The AWS SDK is currently available in Java, .NET, PHP and Ruby. The AWS SDK for Java includes the AWS Flow Framework.

Q: I want to ensure that there is only one execution for each activation of my business process (e.g. a transaction, a submission, or an assignment). How do I accomplish this?   
When you start new workflow executions you provide an ID for that workflow execution. This enables you to associate an execution with a business entity or action (e.g. customer ID, filename, serial number). Amazon SWF ensures that an execution’s ID is unique while it runs. During this time, an attempt to start another execution with the same ID will fail. This makes it convenient for you to satisfy business needs where no more than one execution can be running for a given business action, such as a transaction, submission or assignment. Consider a workflow that registers a new user on a website. When a user clicks the submit button, the user’s unique email address can be used to name the execution. If the execution already exists, the call to start the execution will fail. No additional code is needed to prevent conflicts as a result of the user clicking the button more than one when the registration is in progress.

Once the workflow execution is complete (either successfully or not), you can start another workflow execution with the same ID. This causes a new run of the workflow execution with the same execution ID but a different run ID. The run ID is generated by Amazon SWF and multiple executions that have the same workflow execution ID can be differentiated by the run ID. By allowing you to reuse workflow execution IDs in such a manner, Amazon SWF allows you to address use cases such as retries. For example, in the above user registration example, assume that the workflow execution failed when creating a database record for the user. You can start the workflow execution again with the same execution ID (user’s email address) and do not have to create a new ID for retrying the registration.

Q: How does Amazon SWF help with scaling my applications?  
Amazon SWF lets you scale your applications by giving you full control over the number of workers that you run for each activity type and the number of instances that you run for a decider. By increasing the number of workers or decider instances, you increase the compute resources allocated for the corresponding processing steps and, thereby, the throughput for those steps. To auto-scale, you can use run-time data that Amazon SWF provides through its APIs. For example, Amazon SWF provides the number of tasks in a task list. Since an increase in this number implies that the workers are not keeping up with the load, you can spin up new workers automatically whenever the backlog of tasks crosses a threshold.

Q: I run a large number of mission critical application executions. How can I monitor and scale them?   
In addition to a Management Console, Amazon SWF provides a comprehensive set of visibility APIs. You can use these to get run-time information to monitor all your executions and to auto-scale your executions depending on load. You can get detailed data on each workflow type, such as the count of open and closed executions in a specified time range. Using the visibility APIs, you can also build your own custom monitoring applications.

Q: I have numerous executions running at any time, but a handful of them often fail or stall. How can I detect and troubleshoot these problematic executions?  
Amazon SWF lets you search for executions through its Management Console and visibility APIs. You can search by various criteria, including the time intervals during which executions started or completed, current state (i.e. open or closed), and standard failure modes (e.g. timed out, terminated). To group workflow executions together, you can use upto 5 tags to associate custom text with workflow executions when you start them. In the AWS Management Console, you can use tags when searching workflow executions.

To find executions that may be stalled, you can start with a time-based search to hone in on executions that are running longer than expected. Next, you can inspect them to see task level details and determine if certain tasks have been running too long or have failed, or whether the decider has simply not initiated tasks. This can help you pinpoint the problem at a task-level.

Q: I have an activity type that can be used in multiple applications. Can I share it across these applications?  
Yes. Multiple applications can share a given activity type provided the applications and the activity are all registered within the same domain. To implement this, you can have different deciders initiate tasks for the activity type and add it to the task list that the workers for that activity poll on. The workers of that activity type will then get activity tasks from all the different applications. If you want to tell which application an activity task came from or if you want to deploy different sets of workers for different applications, you can use multiple task lists. Refer to [How do I ensure that a worker or decider only gets tasks that it understands](http://aws.amazon.com/swf/faqs/#gets_tasks_it_understands)?

### **Security and Reliability**

Q: Can I use AWS Identity and Access Management (IAM) to manage access to Amazon SWF?   
Yes. You can grant IAM users permission to access Amazon SWF. IAM users can only access the SWF domains and APIs that you specify.

Q: Can I run my workers behind a firewall?   
Yes. Workers use standard HTTP GET requests to ask Amazon SWF for tasks and to return the computed results. Since workers always initiate requests to Amazon SWF, you do not have to configure your firewall to allow inbound requests.

Q: Isn’t it a security risk to expose my business logic as workers and deciders?   
Workers use standard HTTP GET requests to ask Amazon SWF for tasks and to return the computed results. Thus, you do not have to expose any endpoint for your workers. Furthermore, Amazon SWF only gives tasks to workers when the decider initiates those tasks. Since you write the decider, you have full control over when and how tasks are initiated, including the input data that gets sent with them to the workers.

Q: How does Amazon SWF help in coordinating tasks reliably in my application?   
Amazon SWF provides useful guarantees around task assignment. It ensures that a task is never duplicated and is assigned only once. Thus, even though you may have multiple workers for a particular activity type (or a number of instances of a decider), Amazon SWF will give a specific task to only one worker (or one decider instance). Additionally, Amazon SWF keeps at most one decision task outstanding at a time for a workflow execution. Thus, you can run multiple decider instances without worrying about two instances operating on the same execution simultaneously. These facilities enable you to coordinate your workflow without worrying about duplicate, lost, or conflicting tasks.

### **Limits**

Q: How many workflow types, activity types, and domains can I register with Amazon SWF?  
You can have a maximum of 10,000 workflow and activity types (in total) that are either registered or deprecated in each domain. You can have a maximum of 100 Amazon SWF domains (including registered and deprecated domains) in your AWS account. If you think you will exceed the above limits, please use [this form](http://www.amazon.com/gp/html-forms-controller/AmazonSWF_Limit_Increase_Form) to contact the Amazon SWF team to discuss your scenario and request higher limits.

Q: Are there limits on the number of workflow executions that I can run simultaneously?  
At any given time, you can have a maximum of 100,000 open executions in a domain. There is no other limit on the cumulative number of executions that you run or on the number of executions retained by Amazon SWF. If you think you will exceed the above limits, please use [this form](http://www.amazon.com/gp/html-forms-controller/AmazonSWF_Limit_Increase_Form) to contact the Amazon SWF team to discuss your scenario and request higher limits.

Q: How long can workflow executions run?  
Each workflow execution can run for a maximum of 1 year. Each workflow execution history can grow up to 25,000 events. If your use case requires you to go beyond these limits, you can use features Amazon SWF provides to continue executions and structure your applications using child workflow executions.

Q: What happens if my workflow execution is idle for an extended period of time?  
Amazon SWF does not take any special action if a workflow execution is idle for an extended period of time. Idle executions are subject to the timeouts that you configure. For example, if you have set the maximum duration for an execution to be 1 day, then an idle execution will be timed out if it exceeds the 1 day limit. Idle executions are also subject to the Amazon SWF limit on how long an execution can run (1 year).

Q: How long can a worker take to process a task?  
Amazon SWF does not impose a specific limit on how long a worker can take to process a task. It enforces the timeout that you specify for the maximum duration for the activity task. Note that since Amazon SWF limits an execution to run for a maximum of 1 year, a worker cannot take longer than that to process a task.

Q: How long can Amazon SWF keep a task before a worker asks for it?  
Amazon SWF does not impose a specific limit on how long a task is kept before a worker polls for it. However, when registering the activity type, you can set a default timeout for how long Amazon SWF will hold on to activity tasks of that type. You can also specify this timeout or override the default timeout through your decider code when you schedule an activity task. Since Amazon SWF limits the time that a workflow execution can run to a maximum of 1 year, if a timeout is not specified, the task will not be kept longer than 1 year.

Q: Can I schedule several activity tasks by issuing one decision?   
Yes, you can schedule up to 100 activity tasks in one decision and also issue several decisions one after the other.

Q: How many worker tasks, signals, and markers can I have in a workflow execution and across executions?   
There is no limit on the total number of activity tasks, signals, and timers used during a workflow execution. However at this time, you can only have a maximum of 1,000 open activity tasks per workflow execution. This includes activity tasks that have been initiated and activity tasks that are being processed by workers. Similarly there can be up to 1,000 open timers per workflow execution and up to 1,000 open child executions per workflow execution.

Q: How much data can I transfer within a workflow execution?  
There is no limit on the total amount of data that is transferred during a workflow execution. However, Amazon SWF APIs impose specific maximum limits on parameters that are used to pass data within an execution. For example, the input data that is passed into a activity task and the input data that is sent with a signal can each be a maximum of 32,000 characters.

Q: Does Amazon SWF retain completed executions? If so, for how long?

Amazon SWF retains the history of a completed execution for any number of days that you specify, up to a maximum of 90 days (i.e. approximately 3 months). During retention, you can access the history and search for the execution programmatically or through the console.

Q: When are API calls throttled?   
Beyond infrequent spikes, You may be throttled if you make a very large number of API calls in a very short period of time. If you find that you are frequently throttled or your application encounters frequent spikes, please use [this form](http://www.amazon.com/gp/html-forms-controller/AmazonSWF_Limit_Increase_Form) to contact the Amazon SWF team to discuss your usage scenario and request different throttle settings for your account.

### **Access and Availability**

Q: Which regions is Amazon SWF available in?   
Amazon SWF (SWF) is available in each of the following regions: US East (Northern Virginia), US West (Oregon), US West (Northern California), EU (Ireland), EU (Frankfurt), Asia Pacific (Singapore), Asia Pacific (Tokyo), Asia Pacific (Sydney), South America (Sao Paulo), and AWS GovCloud (US).

Q: Is Amazon SWF available across availability zones?  
Yes, Amazon SWF manages your workflow execution history and other details of your workflows across 3 availability zones so that your applications can continue to rely on Amazon SWF even if there are failures in one availability zone.

Q: What are the Amazon SWF service access points?   
Please visit the [AWS General Reference documentation](http://docs.aws.amazon.com/general/latest/gr/rande.html#swf_region) for more information on access endpoints.

### **Billing**

Q: Do your prices include taxes?

Except as otherwise noted, our prices are exclusive of applicable taxes and duties, including VAT and applicable sales tax. For customers with a Japanese billing address, use of AWS services is subject to Japanese Consumption Tax. [Learn more](https://aws.amazon.com/c-tax-faqs/).