Express vs Standard, Callback URL and Task Tokens

Express vs Standard State Machines

Standard Workflows are ideal for long-running, durable, and auditable workflows. They can run for up to a year and you can retrieve the full execution history using the Step Functions API, up to 90 days after your execution completes. Standard Workflows employ an at-most-once model, where your tasks and states are never executed more than once unless you have specified Retry behavior in ASL. This makes them suited to orchestrating non-idempotent actions.
Express Workflows are ideal for high-volume, event-processing workloads such as IoT data ingestion, streaming data processing and transformation, and mobile application backends. They can run for up to five minutes. Express Workflows employ an at-least-once model, where there is a possibility that an execution might be run more than once. This makes them ideal for orchestrating idempotent actions.

Activities

In AWS Step Functions, you can create activities. Activities are used as a way to associate code running somewhere like Amazon EC2 or Amazon ECS, or any external compute (known as an activity worker) with a specific task in a state machine.
Since this code lives in an environment that is not as tightly integrated with AWS Step Functions, like AWS Lambda for example, there is a specific way in which you need to setup your communications between your activity worker and your state machine.
When you create an activity in Step Function you will get a generated ARN for that state. You will use this ARN for your activity worker to poll for events.
When Step Functions reaches an activity task state, the workflow waits for an activity worker to poll for a task.

Unlike with AWS Lambda, activities need to **poll** for a task. This means that the code running acting as the activity worker must include code for this polling.
The activity worker polls Step Functions by using the GetActivityTask API, and sending the ARN for the related activity in the request. GetActivityTask returns a response including a string of JSON input for the task and a **taskToken**.
A task token is a unique identifier for the task.

After the activity worker completes its work, it can provide a report of its success or failure by using SendTaskSuccess or SendTaskFailure. These two calls use the taskToken provided by GetActivityTask to associate the result with that task.

Callback Pattern Examples

Some applications will require a callback pattern to be implemented. **Callback tasks** provide a way to pause a workflow until a task token is returned. A task might need to wait for a human approval, integrate with a third party, or call legacy systems. For tasks like these, you can pause Step Functions indefinitely, and wait for an external process or workflow to complete.
Read some examples using callback patterns here:
https://docs.aws.amazon.com/step-functions/latest/dg/callback-task-sample-sqs.html
Read about an example using Task Tokens here:
https://docs.aws.amazon.com/step-functions/latest/dg/connect-to-resource.html#connect-wait-example

AWS Step Functions Best Practices

Use Timeouts to Avoid Stuck Executions

By default, the Amazon States Language doesn't set timeouts in state machine definitions. Without a timeout set, the state will wait on a response to move on. If something goes wrong and TimeoutSeconds isn't specified, an execution is stuck waiting for a response that will never come.
To avoid this issue, specify a reasonable timeout when you create a task in your state machine.

Use ARNs Instead of Passing Large Payloads

There is a size limit on the payload you can pass between states. Currently, that limit it set at 32 KB. If you need to pass data between states that exceeds the size limit, use Amazon Simple Storage Service (Amazon S3) to store the data, and pass the Amazon Resource Name (ARN) instead of the raw data.

Avoid Reaching the History Quota

AWS Step Functions has a hard quota of 25,000 entries in the execution history. To avoid reaching this quota for long-running executions, implement a pattern that uses an AWS Lambda function that can start a new execution of your state machine to split ongoing work across multiple workflow executions.

Handle Lambda Service Exceptions

Sometimes, AWS Lambda might return service exceptions to a state. You should proactively handle these exceptions so if they ever occur, you have accounted for which path to take and how to recover or retry after encountering the error.

Avoid Latency When Polling for Activity Tasks

If you only have a small number of polls waiting for a response, it's possible that all requests will queue up behind the blocked request and stop. However, if you have a large number of outstanding polls for each activity Amazon Resource Name (ARN), and some percentage of your requests are stuck waiting, there will be many more that can still get a taskToken and begin to process work.
For production systems, we recommend at least 100 open polls per activity ARN's at each point in time. If one poll gets blocked, and a portion of those polls queue up behind it, there are still many more requests that will receive a taskToken to process work while the GetActivityTask request is blocked.

Choosing Standard or Express Workflows

You can choose Standard Workflows when you need long-running, durable, and auditable workflows, or Express Workflows for high-volume, event processing workloads.
Read more about Step Functions Best Practices here:
https://docs.aws.amazon.com/step-functions/latest/dg/bp-express.html