User Guide for versions 1.12.0 - 1.15.3

AWS SimSpace Weaver

AWS SimSpace Weaver: User Guide for versions 1.12.0 - 1.15.3

Table of Contents

End of support notice: On May 20, 2026, AWS will end support for AWS SimSpace Weaver. After May 20, 2026, you will no longer be able to access the SimSpace Weaver console or SimSpace Weaver resources. For more information, see AWS SimSpace Weaver end of support.

What is AWS SimSpace Weaver?

AWS SimSpace Weaver is a service that you can use to build and run large-scale spatial simulations in the AWS Cloud. For example, you can create crowd simulations, large real-world environments, and immersive and interactive experiences.

With SimSpace Weaver, you can distribute simulation workloads across multiple Amazon Elastic Compute Cloud (Amazon EC2) instances. SimSpace Weaver deploys the underlying AWS infrastructure for you, and handles the simulation data management and network communication between the Amazon EC2 instances running your simulation.

Key concepts for SimSpace Weaver

A simulation or game is limited by the computer that runs it. As you grow the size and complexity of your virtual world, processing performance begins to degrade. Calculations take longer, systems run out of memory, and client frame rates drop. For simulations that don't need real-time performance, this might only be annoying. Or, it could be a business-critical situation in which increased processing delays result in increased costs. If your simulation or game needs real-time performance, then performance degradation is definitely a problem.

A common solution for a simulation that reaches a performance limit is to simplify the simulation. Online games with many users often address scaling problems by making copies of their virtual world on different servers and spreading users across them.

SimSpace Weaver solves the scaling problem by dividing your virtual world spatially and distributing the pieces across a cluster of compute instances that run in the AWS Cloud. The compute instances work together to process the entire simulation world in parallel. Your simulation world appears as a single integrated space to everything in it, and to all clients that connect to it. You no longer have to simplify a simulation because of a hardware performance limit. You can instead add more compute capacity in the cloud.

Topics

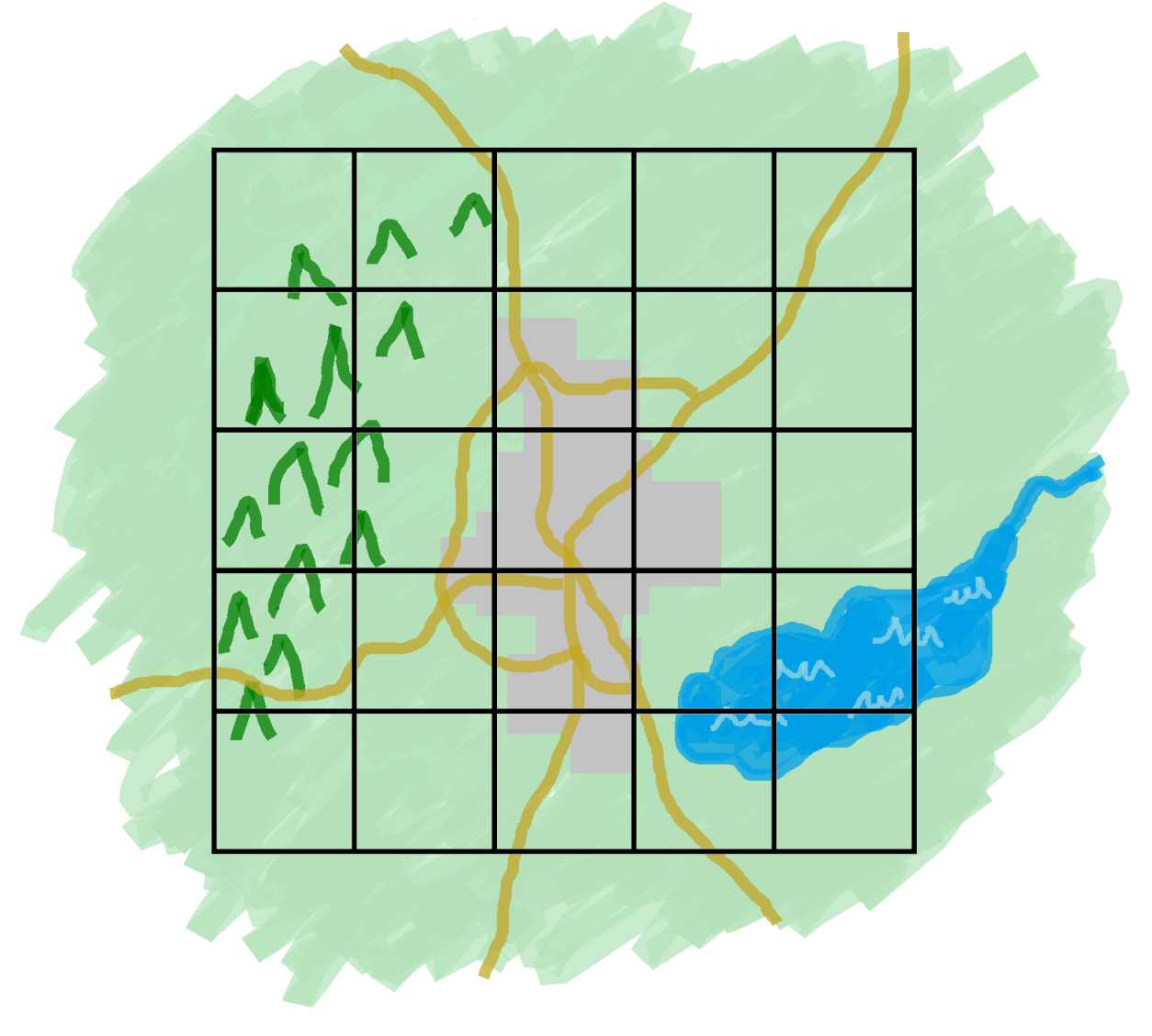
- Simulation clock
- Partitions
- State Fabric
- Entities
- Apps

How SimSpace Weaver works

Your simulation consists of a world with objects in it. Some of the objects (such as people and vehicles) move and do things. Other objects (such as trees and buildings) are static. In SimSpace Weaver, an *entity* is an object in your simulation world.

You define the boundaries of your simulation world and divide it into a grid. Instead of creating simulation logic that operates on the entire grid, you create simulation logic that operates on one cell of the grid. In SimSpace Weaver, a *spatial app* is a program that you write that implements the simulation logic for a cell of your grid. This includes the logic for all entities in that cell. A spatial app's *ownership area* is the grid cell that the spatial app controls.

> **Note**
>
> In SimSpace Weaver, the term "app" can refer to the code of an app or a running instance of that code.



Your simulation world divided into a grid

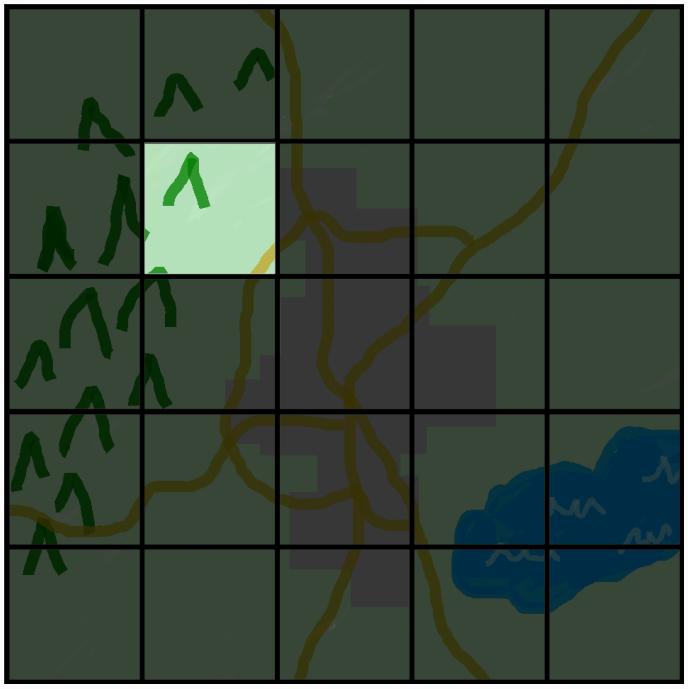
You divide your simulation world into a grid. Each spatial app implements simulation logic for a single cell in that grid.

SimSpace Weaver runs an instance of your spatial app code for each cell of your grid. All of the spatial app instances run in parallel. Essentially, SimSpace Weaver divides your overall simulation into multiple smaller simulations. Each of the smaller simulations handles a part of the overall simulation world. SimSpace Weaver can distribute and run these smaller simulations across multiple Amazon Elastic Compute Cloud (Amazon EC2) instances (called *workers*) in the AWS Cloud. A single worker can run multiple spatial apps.

Entities can move through the simulation world. If an entity enters another spatial app's ownership area (another cell in the grid), then the spatial app owner of the new area takes over control of the entity. If your simulation runs on multiple workers, then an entity could move from the control of a spatial app on one worker to a spatial app on a different worker. When an entity moves to a different worker, SimSpace Weaver handles the underlying network communication.

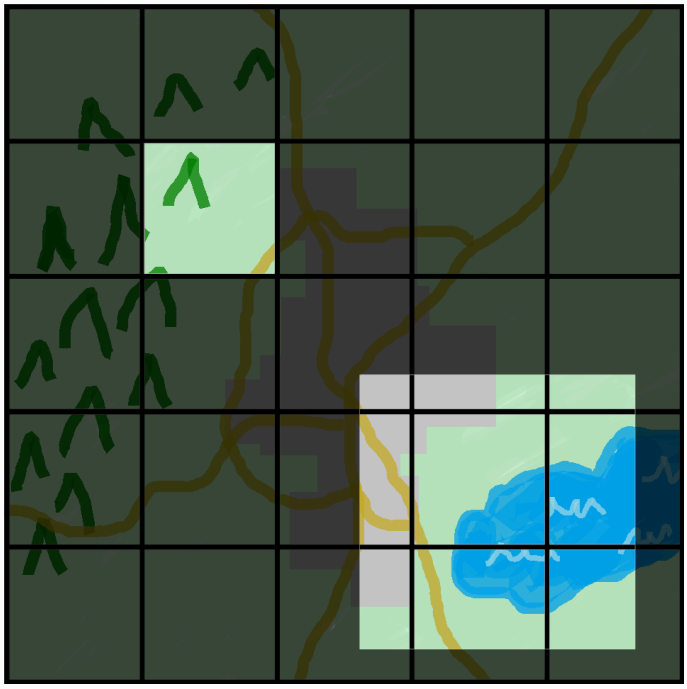
Subscriptions

A spatial app's view of the world is its own ownership area. To find out what's happening in another part of the simulation world, the spatial app creates a *subscription*. The *subscription area* is a subset of the overall simulation world area. A subscription area can include parts of multiple ownership areas, including the spatial app's own ownership area. SimSpace Weaver notifies the spatial app of all entity events (for example, enter, exit, create, update, and delete) that occur within the subscription area.



A spatial app's view of the world

A spatial app's view of the world is its ownership area, which is one cell in the world grid.



A spatial app's view with an added subscription area

A spatial app uses a subscription to find out what's happening in another part of the simulation world. The subscription area can contain multiple grid cells and parts of cells.

For example, an app that simulates entities interacting physically might need to know about entities just beyond the spatial boundaries of its ownership area. To accomplish this, the app can subscribe to areas that border its ownership area. After creating the subscription, the app receives notifications about entity events in those areas and it can read the entities. Another example is an autonomous vehicle that needs to see all entities 200 meters in front of it regardless of what app owns the area. The app for the vehicle can create a subscription with a filter as an axis-aligned bounding box (AABB) that covers the viewable area.

You can create simulation logic that isn't responsible for managing spatial aspects of your simulation. A *custom app* is an executable program that runs on a single worker. You control the lifecycle (start and stop) of a custom app. Simulation clients can connect to a custom app to view or interact with the simulation. You can also create a *service app* that runs on every worker. SimSpace Weaver starts an instance of your service app on every worker that runs your simulation.

Custom apps and service apps create subscriptions to learn about entity events and read entities. These apps don't have ownership areas because they aren't spatial. Using a subscription is the only way that they can find out what is happening in the simulation world.

How you use SimSpace Weaver

When you use SimSpace Weaver, these are the major steps that you follow:

1. Write and build C++ apps that integrate the SimSpace Weaver app SDK.
   a. Your apps make API calls to interact with the simulation state.
2. Write clients that view and interact with your simulation through some apps.
3. Configure your simulation in a text file.
4. Upload your app packages and simulation configuration to the service.
5. Start your simulation.
6. Start and stop your custom apps as needed.
7. Connect clients to your custom or service apps to view or interact with the simulation.
8. Check your simulation logs in Amazon CloudWatch Logs.
9. Stop your simulation.
10. Clean up your simulation.

Simulation schema

The *simulation schema* (or *schema*) is a YAML-formatted text file that contains configuration information for your simulation. SimSpace Weaver uses your schema when it starts a simulation. The SimSpace Weaver app SDK distributable package includes a schema for a sample project. You can use this as a starting point for your own schema. For more information about the simulation schema, see SimSpace Weaver simulation schema reference.

Workers and resource units

A *worker* is an Amazon EC2 instance that runs your simulation. You specify a worker type in your simulation schema. SimSpace Weaver maps your worker type to a specific Amazon EC2 instance type that the service uses. SimSpace Weaver starts and stops your workers for you, and manages network communication between the workers. SimSpace Weaver starts a set of workers for each simulation. Different simulations use different workers.

The available compute (processor and memory) capacity on a worker is divided into logical units called *compute resource units* (or *resource units*). A resource unit represents a fixed amount of processor and memory capacity.

> **Note**
>
> We previously referred to a compute resource unit as a *slot*. You might still see this previous term in our documentation.

Simulation clock

Each simulation has its own clock. You start and stop the clock using API calls or the SimSpace Weaver console. The simulation updates only when the clock is running. All operations in the simulation occur within time segments called *ticks*. The clock announces the start time of each tick to all workers.

The *clock rate* (or *tick rate*) is the number of ticks per second (hertz, or Hz) that the clock announces. The desired clock rate for a simulation is part of the simulation schema. All operations for a tick must complete before the next tick starts. For this reason, the effective clock rate can be lower than the desired clock rate. The effective clock rate won't be higher than the desired clock rate.

Partitions

A *partition* is a segment of the shared memory on a worker. Each partition holds part of the simulation state data.

A partition for a spatial app (also called a *spatial app partition* or *spatial partition*) contains all the entities in a spatial app's ownership area. SimSpace Weaver puts entities in spatial app partitions based on the spatial location of each entity. This means that SimSpace Weaver tries to place entities that are spatially near each other on the same worker. This minimizes the amount of knowledge that an app requires of entities that it doesn't own to simulate the entities that it does own.

State Fabric

The *State Fabric* is the system of shared memory (the collection of all partitions) on all workers. It holds all of the state data for your simulation.

The State Fabric uses a custom binary format that describes an entity as a set of initial data and an update log, for each data field of that entity. With this format, you can access the state of an entity at a previous point in simulation time and map it back to a point in real-world time. The buffer has a finite size, and it's not possible to go back in time beyond what's in the buffer. SimSpace Weaver uses a pointer to the current offset in the update log for each field, and it updates a pointer as part of a field update. SimSpace Weaver maps these update logs into an app’s process space using shared memory.

This object format results in low overhead and no serialization costs. SimSpace Weaver also uses this object format to parse and identify index fields (such as entity position).

Entities

An *entity* is the smallest building block of data in your simulation. Examples of entities include actors (such as people and vehicles) and static objects (such as buildings and obstacles). Entities have properties (such as position and orientation) that you can store as persistent data in SimSpace Weaver. Entities exist within partitions.

Apps

A SimSpace Weaver *app* is software that you write that contains custom logic that runs each simulation tick. The purpose of most apps is to update entities as the simulation runs. Your apps

call APIs in the SimSpace Weaver app SDK to perform actions (such as reading and updating) on entities in your simulation.

You package your apps and their required resources (such as libraries) as .zip files and upload them to SimSpace Weaver. An app runs in a Docker container on a worker. SimSpace Weaver allocates each app a fixed number of resource units on the worker.

SimSpace Weaver assigns ownership of one (and only one) partition to each app. An app and its partition are located on the same worker. Each partition has only one app owner. An app can create, read, update, and delete entities in its partition. An app owns all entities in its partition.

There are three types of apps: *spatial apps*, *custom apps*, and *service apps*. They differ by use cases and lifecycles.

> **Note**
>
> In SimSpace Weaver, the term "app" can refer to the code for an app or a running instance of that code.

Spatial apps

Spatial apps update the state of entities that exist spatially in your simulation. For example, you might define a `Physics` app that's responsible for moving and colliding entities for every tick based on their velocity, shape, and size. In this case, SimSpace Weaver runs multiple instances of the `Physics` app in parallel to handle the size of the workload.

SimSpace Weaver manages the lifecycle of spatial apps. You specify an arrangement of spatial app partitions in your simulation schema. When you launch your simulation, SimSpace Weaver starts a spatial app for each spatial app partition. When you stop the simulation, SimSpace Weaver shuts down your spatial apps.

Other types of apps can create entities, but only spatial apps can update entities. Other types of apps must transfer entities that they create to a spatial domain. SimSpace Weaver uses the spatial location of an entity to move the entity to the partition of a spatial app. This transfers ownership of the entity to the spatial app.

Custom apps

You use *custom apps* to interact with your simulation. A custom app reads entity data using subscriptions. A custom app can create entities. However, the app must transfer an entity to a spatial app to include the entity in the simulation and update it. You can have SimSpace Weaver assign a network endpoint to a custom app. Simulation clients can connect to the network endpoint to interact with the simulation. You define your custom apps in your simulation schema, but you are responsible to start and stop them (using SimSpace Weaver API calls). After you start a custom app instance on a worker, SimSpace Weaver doesn't transfer the instance to another worker.

Service apps

You can use a *service app* when you need a read-only process running on every worker. For example, you can use a service app if you have a large simulation and you need a viewing client that moves through the simulation and displays only the visible entities to the user. In this case, a single custom app instance can't process all the entities in the simulation. You can configure a service app to launch on every worker. Each of these service apps can then filter the entities on its assigned worker and send only the relevant entities to its connected clients. Your viewing client can then connect to different service apps as it moves through the simulation space. You configure service apps in your simulation schema. SimSpace Weaver starts and stops your service apps for you.

App summary

The following table summarizes the characteristics of the different types of SimSpace Weaver apps.

	Spatial apps	Custom apps	Service apps
Read entities	Yes	Yes	Yes
Update entities	Yes	No	No
Create entities	Yes	Yes*	Yes*
Lifecycle	Managed (SimSpace Weaver controls it.)	Unmanaged (You control it.)	Managed (SimSpace Weaver controls it.)
Start method	SimSpace Weaver starts one app	You start each app instance.	SimSpace Weaver starts one or more

	Spatial apps	Custom apps	Service apps
	instance for each spatial partition, as specified in your schema.		app instances on each worker, as specified in your schema.
Clients can connect	No	Yes	Yes

* When a custom app or service app creates an entity, the app must transfer ownership of the entity to a spatial app so that the spatial app can update the state of the entity.

Domains

A SimSpace Weaver *domain* is a collection of app instances that run the same executable app code and have the same launch options and commands. We refer to domains by the types of apps that they contain: spatial domains, custom domains, and service domains. You configure your apps within domains.

Subscriptions and replication

An app creates a *subscription* to a spatial region to learn entity events (for example, enter, exit, create, update, and delete) in that region. An app processes entity events from a subscription before it reads data for entities in partitions that it doesn't own.

A partition can exist on the same worker as the app (this is called a *local partition*), but another app can own the partition. A partition can also exist on a different worker (this is called a *remote partition*). If the subscription is to a remote partition, the worker creates a local copy of the remote partition through a process called *replication*. The worker then reads the local copy (replicated remote partition). If another app on the worker needs to read from that partition on the same tick, then the worker reads the same local copy.

Example use cases for SimSpace Weaver

You can use SimSpace Weaver for agent-based models and discrete time step simulations with a spatial component.

Create large crowd simulations

You can use SimSpace Weaver to simulate crowds in real-world environments. SimSpace Weaver enables you to scale your simulations to millions of dynamic objects with their own behaviors.

Create city-scale environments

Use SimSpace Weaver to create a digital twin of an entire city. Create simulations for city planning, to design traffic routing, and to plan environmental hazard response. You can use your own geospatial data sources as the building blocks for your environments.

Create immersive and interactive experiences

Create simulation experiences that multiple users can participate and interact in. Use popular development tools such as Unreal Engine and Unity to build 3-dimensional (3D) virtual worlds. Customize your 3D experience with your own content and behaviors.

AWS SimSpace Weaver end of support

After careful consideration, we decided to end support for AWS SimSpace Weaver, effective May 20, 2026. AWS SimSpace Weaver will no longer accept new customers beginning May 20, 2025. As an existing customer with an account signed up for the service before May 20, 2025, you can continue to use AWS SimSpace Weaver features. After May 20, 2026, you will no longer be able to use AWS SimSpace Weaver.

For more information about transitioning to AWS Batch to help run containerized simulations, see this blog post.

Setting up for SimSpace Weaver

To get set up for using SimSpace Weaver for the first time, you must set up your AWS account and your local environment. When you complete these tasks, you will be ready for the getting started tutorials.

Setup tasks

1. Set up your AWS account to use SimSpace Weaver.
2. Set up your local environment for SimSpace Weaver.

Set up your AWS account to use SimSpace Weaver

Complete the following tasks to set up your AWS account to use SimSpace Weaver.

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.
2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call or text message and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

   For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.
2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

Add permissions to use SimSpace Weaver

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.
- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Create a role for a third-party identity provider (federation) in the *IAM User Guide*.
- IAM users:
  - Create a role that your user can assume. Follow the instructions in Create a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

Example IAM policy to grant permissions to use SimSpace Weaver

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateAndRunSimulations",
            "Effect": "Allow",
            "Action": [
                "simspaceweaver:*",
                "iam:GetRole",
```

```
                "iam:ListRoles",
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:UpdateRole",
  "iam:CreatePolicy",
  "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "iam:GetRolePolicy",
                "iam:DeleteRolePolicy",
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListAllMyBuckets",
                "s3:PutBucketPolicy",
                "s3:CreateBucket",
                "s3:ListBucket",
                "s3:PutEncryptionConfiguration",
                "s3:DeleteBucket",
                "cloudformation:CreateStack",
                "cloudformation:UpdateStack",
                "cloudformation:DescribeStacks"
            ],
            "Resource": "*"
        },
        {
            "Sid": "PassAppRoleToSimSpaceWeaver",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "simspaceweaver.amazonaws.com"
                }
            }
        }
    ]
}
```

Set up your local environment for SimSpace Weaver

SimSpace Weaver simulations run in containerized Amazon Linux 2 (AL2) environments. You must have an AL2 environment to compile and link your apps with the SimSpace Weaver app SDK. The standard local development environment is an AL2 container in Docker. If you choose not to

use Docker, we provide alternate instructions to run an AL2 environment in Windows Subsystem for Linux (WSL). You can also use your own method to create a local AL2 environment. For some additional ways to run AL2 locally, see the Amazon EC2 documentation.

> **Important**
>
> **Docker on Microsoft Windows is the standard development environment.** For your convenience, we suggest other ways to set up your local development environment, but they are not standard and are unsupported.

Topics

Set up Amazon Linux 2 (AL2) in Docker

This section provides instructions for setting up your local AL2 environment in Docker. For instructions to set up AL2 in Windows Subsystem for Linux (WSL), see Set up Amazon Linux 2 (AL2) in Windows Subsystem for Linux (WSL).

Requirements

- Microsoft Windows 10 or higher
- Microsoft Visual Studio 2019 or later, with the *Desktop development with C++* workload installed
- CMake3
- Git
- Docker Desktop
- AWS CLI

To set up AL2 in Docker

1. If you have not already configured your AWS credentials for the AWS CLI, follow these instructions: Configuring the AWS CLI. If you are just using SimSpace Weaver then you can configure the AWS CLI to use your SimSpace Weaver credentials by default.
2. Download the SimSpace Weaver app SDK distributable package. It contains the following:

- Binaries and libraries for SimSpace Weaver app development
- Helper scripts that automate parts of the development workflow
- Sample applications that demonstrate SimSpace Weaver concepts

3. Unzip the file to an *sdk-folder* of your choice.
4. Go to the *sdk-folder*.
5. Enter the following command to create a Docker image.

```
docker-create-image.bat
```

> **Note**
>
> If you get an error during this step, make sure that Docker is running.

Set up Amazon Linux 2 (AL2) in Windows Subsystem for Linux (WSL)

This section provides instructions for setting up your local AL2 environment in Windows Subsystem for Linux (WSL). For instructions to set up AL2 in Docker, see Set up Amazon Linux 2 (AL2) in Docker.

> **Important**
>
> This section describes a solution that uses a version of AL2 that is not owned, developed, or supported by Amazon. This solution is provided for your convenience only, if you choose not to use Docker. Amazon and AWS assume no liability if you choose to use this solution.

Requirements

- Hyper-V on Windows 10
- Windows Subsystem for Linux (WSL)
- Third-party open source AL2 distribution for WSL (download version 2.0.20200722.0-update.2) (see the instructions)

> **Important**
>
> Our WSL instructions use the *2.0.20200722.0-update.2* version of the AL2 distribution for WSL. You might experience errors if you use any other version.

To set up AL2 in WSL

1. At a **Windows command prompt**, start your AL2 environment in WSL.

```
wsl -d Amazon2
```

> **Important**
>
> While you are running in WSL, use the Linux instructions instead of the Windows instructions in this guide (when you have a choice). If there are no Linux instructions, use the following substitutions:
>
> - Use `tools/linux` instead of `tools\windows`
> - Use `.sh` scripts instead of `.bat` scripts

2. At a **Linux shell prompt**, update your yum package manager.

```
yum update -y
```

> **Important**
>
> If this step times-out, you might need to switch to WSL1 and retry these procedures. Exit your WSL AL2 session and enter the following at your **Windows command prompt**:
>
> ```
> wsl --set-version Amazon2 1
> ```

3. Install the unzip tool.

```
yum install -y unzip
```

4. Remove any AWS CLI that `yum` installed. Try both of the following commands if you are unsure if `yum` installed an AWS CLI.

```
yum remove awscli
```

```
yum remove aws-cli
```

5. Make a temporary directory and go to it.

```
mkdir ~/temp
cd ~/temp
```

6. Download and install the AWS CLI:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
./aws/install
```

7. You can remove the temporary directory.

```
cd ~
rm -rf temp
```

8. Restart the shell session to update the path in the environment.

```
exec
```

9. Configure your AWS credentials for the AWS CLI in your AL2 environment. For more information, see Configuring the AWS CLI. If you use AWS IAM Identity Center, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*.

```
aws configure
```

10. Install Git.

```
yum install -y git
```

11. Install `wget`.

```
yum install -y wget
```

12. Create a folder for the SimSpace Weaver app SDK.

```
mkdir sdk-folder
```

13. Go to your SDK folder.

```
cd sdk-folder
```

14. Download the SimSpace Weaver app SDK distributable file. It contains the following:

- Binaries and libraries for SimSpace Weaver app development
- Helper scripts that automate parts of the development workflow
- Sample applications that demonstrate SimSpace Weaver concepts

```
wget https://artifacts.simspaceweaver.us-east-2.amazonaws.com/latest/
SimSpaceWeaverAppSdkDistributable.zip
```

15. Unzip the file.

```
unzip *.zip
```

16. Run the additional setup script.

```
source ./setup-wsl-distro.sh
```

> **Note**
>
> You only need to do this one time for your AL2 environment in WSL.

Use of licensed software with AWS SimSpace Weaver

AWS SimSpace Weaver allows you to build simulations with your choice of simulation engine and content. In connection with your use of SimSpace Weaver, you are responsible for obtaining, maintaining, and adhering to the license terms of any software or content you use in your

simulations. Verify your licensing agreement allows you to deploy your software and content in a virtual hosted environment.

Getting started with SimSpace Weaver

This section provides tutorials to help you get started with SimSpace Weaver. These tutorials introduce you to the general workflow for building simulations with SimSpace Weaver. In both tutorials, you will learn how to create, deploy, and run simulations in SimSpace Weaver. We recommend that you begin with the quick start tutorial to get a simulation running in minutes. Then, go through the detailed tutorial to learn more about each step in the process.

These tutorials use a sample application (`PathfindingSample`) included in the SimSpace Weaver app SDK .zip file that you downloaded during the setup procedures. The sample application demonstrates the concepts that all SimSpace Weaver simulations share, including spatial partitioning, cross-partition entity handoff, apps, and subscriptions.

In the tutorials, you will create a simulation with four spatial partitions. A separate instance of the `PathfindingSample` spatial app manages each individual partition. The spatial apps create entities in their own partitions. The entities move to a particular position in the simulation world, avoiding obstacles as they move. You can use a separate client application (included in the SimSpace Weaver app SDK) to view the simulation.

Topics

Quick start tutorial: Build and run a simulation in minutes

This tutorial guides you through the process to build and run a simulation on SimSpace Weaver in minutes. We recommend that you begin with this tutorial and then go through the detailed tutorial afterwards.

Requirements

Before you begin, be sure that you completed the steps in Setting up for SimSpace Weaver.

Steps

Step 1: Create a project

The SimSpace Weaver app SDK distributable contains a script that creates a project from the bundled `PathfindingSample` project. You must run the script from its location in your file system. The script creates a *`project-name`* inside a *`path`* using the values that you provide at the command line.

Docker

To create a project

1. At a **Windows command prompt**, change to your project folder.

   ```
   cd sdk-folder
   ```

2. Run the `create-project.bat` script.

   ```
   .\create-project.bat --name project-name --path path
   ```

   The full command and parameter list follow.

   ```
   .\create-project.bat --name project-name --path path --app-sdk-version version-
   number --template template-name --overwriteproject
   ```

   > **Important**
   >
   > Don't use a project name longer than 20 characters. Exceeding this limit might result in errors.

> **Note**
>
> If you receive a prompt to share a drive with Docker, choose **Yes** to share.

> **Important**
>
> If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

To create a project

1. At a **Linux shell prompt**, change to your project folder.

```
cd sdk-folder
```

2. Run the `create-project.sh` script.

```
./create-project.sh --name project-name --path path
```

The full command and parameter list follow.

```
./create-project.sh --name project-name --path path --profile cli-profile-name
 --app-sdk-version version-number --template template-name --overwriteproject
```

Important

Don't use a project name longer than 20 characters. Exceeding this limit might result in errors.

Important

If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

Parameters

name

The name of your project.

path

The location of the project in your file system. The project structure can sometimes have long file paths that can exceed the path length limit in your operating system. We recommend that you use a path name that is as short as possible.

profile

The name of the AWS CLI profile that the script should use for authentication. For more information, see Configuration and credential file settings in the *AWS Command Line Interface User Guide*. This parameter is only available for the SimSpace Weaver app SDK version 1.12.1 or higher. For more information about SimSpace Weaver versions, see SimSpace Weaver versions.

app-sdk-version

(optional) The version of the SimSpace Weaver app SDK that your project uses. You will build and link your apps using this version. If the script doesn't find the version in the location of the distributable, or if you don't provide the version number, the script will automatically download the latest version.

template

(optional) A project template that the script will use to create your project. If you don't provide a template, the script will use the `PathfindingSample`. Valid values:

- **`PathfindingSample`** – a sample application that uses a single worker.
- **`MultiWorkerPathfindingSample`** – a version of the sample application that uses multiple workers.

overwriteproject

(optional) Use this option to overwrite an existing project folder with the same project name and path.

Step 2: Turn on logging (optional)

Logging is off by default for the `PathfindingSample` project. This tutorial assumes that logging is on. You can choose to turn on logging, but it is optional.

> **Important**
>
> The `PathfindingSample` generates large amounts of log data. If you choose to turn on logging, you will receive billing charges for the log data. You will continue to receive

billing charges for the log data as long as it exists. We highly recommend that you stop this simulation and perform the cleanup steps at the end of the tutorial as soon as possible if you turn on logging.

To turn on logging

1. Open the following file in a text editor:

   Docker

   project-folder\tools*project-name*-schema.yaml

   > **Note**
   >
   > Your *project-folder* is *path**project-name* using the values that you provided when you created the project.

   WSL

   > **Important**
   >
   > We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

   project-folder/tools/*project-name*-schema.yaml

   > **Note**
   >
   > Your *project-folder* is *path*/*project-name* using the values that you provided when you created the project.

2. Find the `simulation_properties:` section at the beginning of the file:

```
simulation_properties:
```

```
  default_entity_index_key_type: "Vector3<f32>"
```

3. Insert the following 2 lines after the line `simulation_properties:`:

```
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
```

4. Confirm that your `simulation_properties:` section is the same as the following:

```
simulation_properties:
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
  default_entity_index_key_type: "Vector3<f32>"
```

5. Save the file and exit your text editor.

Step 3: Run the quick start script

The sample application includes a quick start script. The script will create, build, upload, and start your simulation and its apps.

Docker

To run the quick start script

1. If you aren't there already, go to the tools folder for your project and platform. Your *project-folder* is *path**project-name* using the values that you provided when you created the project.

   At a **Windows command prompt**, enter:

   ```
   cd project-folder\tools\windows
   ```

2. Run the quick start script for this project.

   ```
   .\quick-start-project-name-cli.bat
   ```

> **Important**
>
> Starting with version 1.12.3, the `quick-start` script starts your simulation with a maximum duration of 1 hour. You can use the `--maximum-duration` parameter to specify a different maximum duration. In version 1.12.2 or lower, you can't provide a maximum duration to the script and simulations have a maximum duration of 14 days. For more information about the maximum duration of a simulation, see Maximum duration of a simulation.

> **Important**
>
> If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

To run the quick start script

1. If you aren't there already, go to the tools folder for your project and platform.Your *project-folder* is *path*/*project-name* using the values that you provided when you created the project.

   At a **Linux shell prompt**, enter:

   ```
   cd project-folder/tools/linux
   ```

2. Run the quick start script for this project.

   ```
   ./quick-start-project-name-cli.sh
   ```

   > **Important**
   >
   > Starting with version 1.12.3, the `quick-start` script starts your simulation with a maximum duration of 1 hour. You can use the `--maximum-duration` parameter to specify a different maximum duration. In version 1.12.2 or lower, you can't provide a maximum duration to the script and simulations have a maximum duration of 14 days. For more information about the maximum duration of a simulation, see Maximum duration of a simulation.

   > **Important**
   >
   > If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

The script will start a loop and stop automatically after all components are STARTED. Look for output similar to the following:

```
[2022-10-04T22:15:28] [INFO] Describe Simulation Results:
[2022-10-04T22:15:28] [INFO] {
    "Status": "STARTED",
    "Name": "MyProjectSimulation_22-10-04_22_10_15",
    "RoleArn": "arn:aws:iam::111122223333:role/weaver-MyProject-app-role",
    "CreationTime": 1664921418.09,
```

Step 4: Get your IP address and port number

You must get the IP address and port number of your view (custom) app so that you can connect to the simulation. The following procedure assumes that you don't know anything about your simulation (such as the simulation name). You can use this procedure at any time to find the IP address and port number for a custom app or service app. The following example output is for a project named `MyProject`.

Docker

To get your IP address and port number

1. If you aren't there already, go to the tools folder for your project and platform. Your *`project-folder`* is *`path`**`project-name`* using the values that you provided when you created the project.

   At a **Windows command prompt**, enter:

   ```
   cd project-folder\tools\windows
   ```

2. Use the **ListSimulations** API to get the name of your simulation.

   ```
   .\weaver-project-name-cli.bat list-simulations
   ```

> **Important**
>
> If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

Example output:

```
{
    "Simulations": [
        {
            "Status": "STARTED",
            "CreationTime": 1664921418.09,
            "Name": "MyProjectSimulation_22-10-04_22_10_15",
            "Arn": "arn:aws:simspaceweaver:us-west-2: 111122223333:simulation/
MyProjectSimulation_22-10-04_22_10_15",
            "TargetStatus": "STARTED"
        }
    ]

}
```

3. Use the **DescribeSimulation** API to get a list of domains in your simulation.

```
.\weaver-project-name-cli.bat describe-simulation --simulation simulation-name
```

Look for the `Domains` section in the `LiveSimulationState` section of the output.

Example output:

```
    "LiveSimulationState": {
        "Domains": [
            {
                "Type": "",
                "Name": "MySpatialSimulation",
                "Lifecycle": "Unknown"
            },
            {
                "Type": "",
                "Name": "MyViewDomain",
                "Lifecycle": "ByRequest"
            }
        ],
```

4. Use the **ListApps** API to get a list of custom apps in a domain. The domain name for the view (custom) app in the sample project is `MyViewDomain`. Look for the app name in the output.

```
.\weaver-project-name-cli.bat list-apps --simulation simulation-name --
domain domain-name
```

Example output:

```
{
    "Apps": [
        {
            "Status": "STARTED",
            "Domain": "MyViewDomain",
            "TargetStatus": "STARTED",
            "Name": "ViewApp",
            "Simulation": "MyProjectSimulation_22-10-04_22_10_15"
        }
    ]
}
```

5. Use the **DescribeApp** API to get the IP address and port number. For the sample project, the domain name is `MyViewDomain` and the app name is `ViewApp`.

```
.\weaver-project-name-cli.bat describe-app --simulation simulation-name --
domain domain-name --app app-name
```

The IP address and port number are in the `EndpointInfo` block in the output. The IP address is the value of `Address` and the port number is the value of `Actual`.

Example output:

```
{
    "Status": "STARTED",
    "Domain": "MyViewDomain",
    "TargetStatus": "STARTED",
    "Simulation": "MyProjectSimulation_22-10-04_22_10_15",
    "LaunchOverrides": {
        "LaunchCommands": []
    },
    "EndpointInfo": {
        "IngressPortMappings": [
            {
                "Declared": 7000,
                "Actual": 4321
            }
        ],
        "Address": "198.51.100.135"
    },
    "Name": "ViewApp"
}
```

Note

The value of `Declared` is the port number that your app code should bind to. The value of `Actual` is the port number that SimSpace Weaver exposes to clients to

connect to your app. SimSpace Weaver maps the `Declared` port to the `Actual` port.

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

To get your IP address and port number

1. If you aren't there already, go to the tools folder for your project and platform.Your *`project-folder`* is *`path`*`/`*`project-name`* using the values that you provided when you created the project.

   At a **Linux shell prompt**, enter:

   ```
   cd project-folder/tools/linux
   ```

2. Use the **ListSimulations** API to get the name of your simulation.

   ```
   ./weaver-project-name-cli.sh list-simulations
   ```

   > **Important**
   >
   > If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface*

User Guide and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

Example output:

```
{
    "Simulations": [
        {
            "Status": "STARTED",
            "CreationTime": 1664921418.09,
            "Name": "MyProjectSimulation_22-10-04_22_10_15",
            "Arn": "arn:aws:simspaceweaver:us-west-2: 111122223333:simulation/
MyProjectSimulation_22-10-04_22_10_15",
            "TargetStatus": "STARTED"
        }
    ]

}
```

3. Use the **DescribeSimulation** API to get a list of domains in your simulation.

```
./weaver-project-name-cli.sh describe-simulation --simulation simulation-name
```

Look for the `Domains` section in the `LiveSimulationState` section of the output.

Example output:

```
    "LiveSimulationState": {
        "Domains": [
            {
                "Type": "",
                "Name": "MySpatialSimulation",
                "Lifecycle": "Unknown"
            },
            {
                "Type": "",
                "Name": "MyViewDomain",
```

```
                "Lifecycle": "ByRequest"
            }
        ],
```

4. Use the **ListApps** API to get a list of custom apps in a domain. The domain name for the view (custom) app in the sample project is `MyViewDomain`. Look for the app name in the output.

```
./weaver-project-name-cli.sh list-apps --simulation simulation-name --
domain domain-name
```

Example output:

```
{
    "Apps": [
        {
            "Status": "STARTED",
            "Domain": "MyViewDomain",
            "TargetStatus": "STARTED",
            "Name": "ViewApp",
            "Simulation": "MyProjectSimulation_22-10-04_22_10_15"
        }
    ]
}
```

5. Use the **DescribeApp** API to get the IP address and port number. For the sample project, the domain name is `MyViewDomain` and the app name is `ViewApp`.

```
./weaver-project-name-cli.sh describe-app --simulation simulation-name --
domain domain-name --app app-name
```

The IP address and port number are in the `EndpointInfo` block in the output. The IP address is the value of `Address` and the port number is the value of `Actual`.

Example output:

```
{
    "Status": "STARTED",
    "Domain": "MyViewDomain",
    "TargetStatus": "STARTED",
    "Simulation": "MyProjectSimulation_22-10-04_22_10_15",
    "LaunchOverrides": {
        "LaunchCommands": []
    },
    "EndpointInfo": {
        "IngressPortMappings": [
            {
                "Declared": 7000,
                "Actual": 4321
            }
        ],
        "Address": "198.51.100.135"
    },
    "Name": "ViewApp"
}
```

Note

The value of `Declared` is the port number that your app code should bind to. The value of `Actual` is the port number that SimSpace Weaver exposes to clients to connect to your app. SimSpace Weaver maps the `Declared` port to the `Actual` port.

Step 5: View your simulation

The SimSpace Weaver app SDK provides different options to view the sample application. You can use the sample console client if you don't have any local support for Unreal Engine development. The instructions for the Unreal Engine client assume that you are using Windows.

The console client displays a list of entity events as they occur. The client gets the entity event information from the `ViewApp`. If your console client displays the list of events, then it confirms network connectivity with the `ViewApp` and activity in your simulation.

The `PathfindingSample` simulation creates stationary and moving entities on a 2-dimensional plane. The moving entities move around the stationary entities. The Unreal Engine client provides a visualization of the entity events.

Windows console client

Requirements

- Microsoft Windows 10 or higher
- Microsoft Visual Studio 2019 or later, with the *Desktop development with C++* workload installed
- CMake3
- Git

To connect to the sample application with the sample console client

1. In a **command prompt window**, go to the folder for the console client (in the app SDK folder).

   ```
   cd sdk-folder\packaging-tools\clients\PathfindingSampleClients\ConsoleClient
   ```

2. Use CMake3 to create a Visual Studio solution in this folder.

   ```
   cmake .
   ```

   Note

   Make sure to include the space and period at the end.

   Important

   Keep the command prompt window open for further steps.

3. In **Visual Studio**, open the `PathfindingSampleConsoleClient.sln` that you created in the previous step.
4. Select the **RelWithDebInfo** build configuration.

5. Choose **Build** > **Build Solution**.
6. In your previous **command prompt window**, go to the build output folder in the folder for the console client.

```
cd RelWithDebInfo
```

7. Run the client with the IP address and port number of your `ViewApp`.

```
.\ConsoleClient.exe --url tcp://ip-address:port-number
```

Your command prompt window should display numbers for entity update, delete, and create events, similar to the following example output.

> **Note**
>
> The IP addresses and port numbers in the following example output are placeholders. Provide the IP address and port number of your `ViewApp` to the console client. Provide the `Actual` port number if you want to connect to a `ViewApp` that runs in the AWS Cloud. Provide the IP address and port number 127.0.0.1:7000 when you connect to a `ViewApp` that runs on your local system. For more information, see Local development.

```
##PathfindingSample#ViewApp Message Reader##

Added argument url:tcp://198.51.100.135:4321
Some subscription arguments are missing, restoring defaults.

*****
Sample usage without a MoveStrategy:
ConsoleClient --url tcp://198.51.100.135:4321 --subs-center-x 600 --subs-center-
y 500 --subs-radius 50
Sample usage with CircleMoveStrategy:
ConsoleClient --url tcp://198.51.100.135:4321 --subs-center-x 600 --subs-center-
y 500 --subs-radius 50 --subs-move-strategy circle --circle-center-x 500 --
circle-center-y 500 --circle-speed 0.001
```

```
*****
Starting NNG client. NNG version: 1.2.4
Creating socket ...done.
Connecting to View App ... done.
Initiating connection to tcp:// 198.51.100.135:4321 ... done.

Receiving messages ...
[2022-10-04 19:13:00.710] CreateEntity Count: 72
[2022-10-04 19:13:00.756] UpdateEntity Count: 42
[2022-10-04 19:13:00.794] DeleteEntity Count: 72
[2022-10-04 19:13:03.690] CreateEntity Count: 11
[2022-10-04 19:13:03.725] UpdateEntity Count: 2
[2022-10-04 19:13:03.757] UpdateEntity Count: 2
[2022-10-04 19:13:03.790] UpdateEntity Count: 2
```

> **Note**
>
> For troubleshooting guidance, see PathfindingSample console client fails to connect.

8. Press **CTRL+C** to quit the console client.

Linux console client

> **Important**
>
> We provide these instructions for your convenience. They might not work in some Linux environments. These procedures are unsupported.

This procedure assumes that you are working entirely within a Linux environment. You can also view your simulation using clients built in Windows.

Requirements

- CMake3
- C compiler (already included in Amazon Linux 2)
- Git

To connect to the sample application with the sample console client

1. At a **Linux shell prompt**, go to the folder for the console client (in the app SDK folder).

```
cd sdk-folder/packaging-tools/clients/PathfindingSampleClients/ConsoleClient
```

2. Create a build folder.

```
mkdir build
```

3. Go to the build folder.

```
cd build
```

4. Use CMake3 to build the client.

```
cmake3 ../ && cmake3 --build .
```

> **Note**
>
> Make sure to include the space and period at the end.

5. Run the client with the IP address and port number of your `ViewApp`.

```
./ConsoleClient --url tcp://ip-address:port-number
```

Your command prompt window should display numbers for entity update, delete, and create events, similar to the following example output.

> **Note**
>
> The IP addresses and port numbers in the following example output are placeholders. Provide the IP address and port number of your `ViewApp` to the console client. Provide the `Actual` port number if you want to connect to a `ViewApp` that runs in the AWS Cloud. Provide the IP address and port number 127.0.0.1:7000 when you connect to a `ViewApp` that runs on your local system. For more information, see Local development.

```
##PathfindingSample#ViewApp Message Reader##

Added argument url:tcp://198.51.100.135:4321
Some subscription arguments are missing, restoring defaults.

*****
Sample usage without a MoveStrategy:
ConsoleClient --url tcp://198.51.100.135:4321 --subs-center-x 600 --subs-center-
y 500 --subs-radius 50
Sample usage with CircleMoveStrategy:
ConsoleClient --url tcp://198.51.100.135:4321 --subs-center-x 600 --subs-center-
y 500 --subs-radius 50 --subs-move-strategy circle --circle-center-x 500 --
circle-center-y 500 --circle-speed 0.001

*****
Starting NNG client. NNG version: 1.2.4
Creating socket ...done.
Connecting to View App ... done.
Initiating connection to tcp:// 198.51.100.135:4321 ... done.

Receiving messages ...
[2022-10-04 19:13:00.710] CreateEntity Count: 72
[2022-10-04 19:13:00.756] UpdateEntity Count: 42
[2022-10-04 19:13:00.794] DeleteEntity Count: 72
[2022-10-04 19:13:03.690] CreateEntity Count: 11
[2022-10-04 19:13:03.725] UpdateEntity Count: 2
[2022-10-04 19:13:03.757] UpdateEntity Count: 2
[2022-10-04 19:13:03.790] UpdateEntity Count: 2
```

> **Note**
>
> For troubleshooting guidance, see PathfindingSample console client fails to connect.

6. Press **CTRL+C** to quit the console client.

Unreal Engine on Windows

Requirements

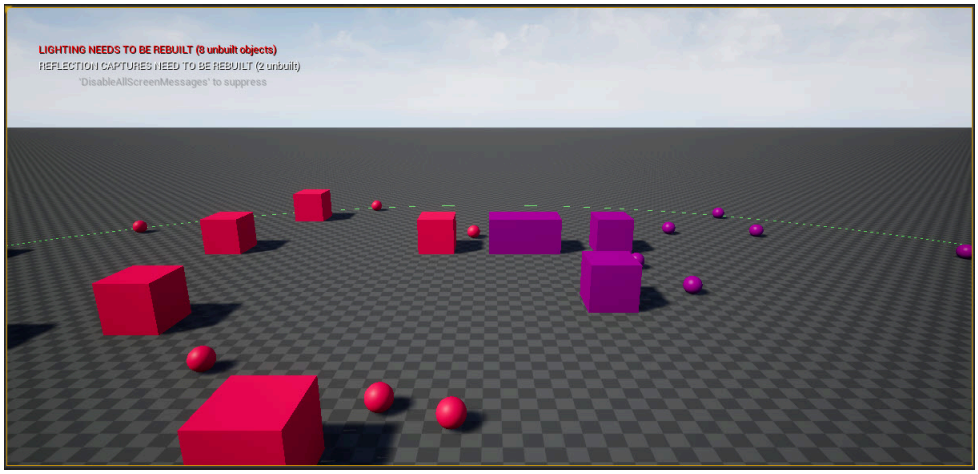
- Unreal Engine 5 development environment
- Microsoft .NET Framework 4.8 Developer Pack
- Windows console client (see the **Windows console client** tab on this page)

> **Important**
>
> Other versions of Unreal Engine and .NET are not supported and might cause problems.

To connect to the sample application with the sample Unreal client

1. The Unreal Engine client uses the NNG library from the console client. You must build the console client for Windows if you haven't already built it. For more information, see the **Windows console client** tab on this page.
2. In a **file manager window**, go to *sdk-folder*\packaging-tools\clients\PathfindingSampleClients\UnrealClient.
3. Open UnrealClient.uproject.
4. If the editor asks you if you want to rebuild the UnrealClient modules, choose **yes**.
5. In a **text editor**, open *sdk-folder*\packaging-tools\clients\PathfindingSampleClients\UnrealClient\view_app_url.txt.
6. Update the URL with the IP address and port number for your view app: tcp://*ip-address*:*port-number* (it should look like tcp://198.51.100.135:1234).
7. In **Unreal editor**, choose **play**.

Your Unreal editor should display a visualization of the simulation, similar to the following screenshot.



> **Note**
>
> Depending on the power of your local development system, it could take a few minutes for the Unreal editor to display the simulation. During this time, the system might appear to freeze.

Use the **W**, **A**, **S**, **D** keys to move in the Unreal client. Hold the mouse button and drag the mouse to turn.

You can press the **[** (left square bracket) key to decrease the size of the subscription area. You can press the **]** (right square bracket) key to increase the size of the subscription area. The size of the subscription area determines the number of entities that appear in the client.

You can press the **C** key to create an entity in the simulation. The client sends a `CreateEntity` command to the view app. The view app will then create the entity and transfer it to the spatial domain.

You can examine the code for `ViewAppDriver::HandleEntityCreationRequests` in *`project-folder`*`\src\PathfindingSample\ViewApp\Driver\ViewAppDriver.cpp` to see how the app implements this process.

Step 6: Stop and clean up your simulation

It's important to clean up your simulations when you don't need them anymore. SimSpace Weaver simulation resources count towards your service quotas (limits), even if your simulation is stopped.

You will continue to get billing charges for simulations that are running. You might also get billing charges for data storage in supporting services, such as Amazon CloudWatch Logs and Amazon Simple Storage Service For more information about SimSpace Weaver service quotas, see SimSpace Weaver endpoints and quotas.

Follow the procedures in this section when you are ready to clean up your simulation.

> **Important**
>
> You can't restart a stopped simulation.

> **Important**
>
> You can't recover a deleted simulation.

Clean up simulation resources in SimSpace Weaver

You must stop your simulation before you can delete it. Deleting a simulation removes resources in SimSpace Weaver only. You must perform separate steps to delete resources that your simulation created or uses in other services (see the following section).

Docker

To clean up your simulation

1. If you aren't there already, go to the tools folder for your project and platform. Your *project-folder* is *path**project-name* using the values that you provided when you created the project.

   At a **Windows command prompt**, enter:

   ```
   cd project-folder\tools\windows
   ```

2. Find the names of your simulations.

   ```
   .\weaver-project-name-cli.bat list-simulations
   ```

> **Important**
>
> If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

3. Stop a simulation.

```
.\weaver-project-name-cli.bat stop-simulation --simulation simulation-name
```

4. Delete a stopped simulation.

```
.\weaver-project-name-cli.bat delete-simulation --simulation simulation-name
```

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

To clean up your simulation

1. If you aren't there already, go to the tools folder for your project and platform.Your *`project-folder`* is *`path`*`/`*`project-name`* using the values that you provided when you created the project.

At a **Linux shell prompt**, enter:

```
cd project-folder/tools/linux
```

2. Find the names of your simulations.

```
./weaver-project-name-cli.sh list-simulations
```

> **Important**
>
> If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile cli-profile-name` parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

3. Stop a simulation.

```
./weaver-project-name-cli.sh stop-simulation --simulation simulation-name
```

4. Delete a stopped simulation.

```
./weaver-project-name-cli.sh delete-simulation --simulation simulation-name
```

AWS Management Console

To clean up your simulation

1. Open the SimSpace Weaver console at SimSpace Weaver console.
2. From the navigation pane, choose **Simulations**.

3. From the **Simulations** list, select the option next to the name of the simulation that you want to delete.
4. If the **Status** of the simulation that you selected is **STARTED**:
   a. Choose the **Actions** drop-down menu.
   b. Choose **Stop**.
   c. To confirm, enter your simulation name.
   d. Choose **Stop**.
   e. Wait until the **Status** of your simulation is **STOPPED**.
5. Choose the **Actions** drop-down menu.
6. Choose **Delete**.
7. To confirm, choose **Delete**.

Clean up simulation resources in supporting services

To support your simulation, SimSpace Weaver creates resources in other services. SimSpace Weaver doesn't delete these resources when you delete your simulation. You can delete these additional resources if you don't need them.

> **Important**
>
> You might get billing charges for any of these resources that you don't delete.

To delete support resources for your project

1. If you are done with your **project**, delete its CloudFormation stack. For more information about working with AWS CloudFormation, see Deleting a stack on the AWS CloudFormation console in the *AWS CloudFormation User Guide*.
   - weaver-*project-name*-stack

   > **Important**
   >
   > Simulations that you started from the same project share resources such as the app role. When you delete the CloudFormation stack, you delete the app role. Don't

delete your CloudFormation stack if you have other simulations that share the same resources.

> **Note**
>
> Your CloudFormation stack will probably report `DELETE_FAILED` because it can't delete Amazon S3 buckets that aren't empty. You will delete your Amazon S3 buckets in the following step.

2. If you are done with your **project**, delete its Amazon S3 bucket(s). For more information on working with Amazon S3 buckets, see Deleting a bucket in the *Amazon Simple Storage Service User Guide*.

   - `weaver-`*`lowercase-project-name`*`-`*`account-number`*`-`*`region`*

   For example, the project named `MyProject` in the account `111122223333` in the `us-west-2` Region has the following bucket:

   - `weaver-myproject-111122223333-us-west-2`

   > **Note**
   >
   > You must delete the contents of an Amazon S3 bucket before you can delete the bucket.

   > **Note**
   >
   > SimSpace Weaver app SDK version 1.12.x projects use separate buckets for the app .zip files and the schema:
   >
   > - weaver-*lowercase-project-name*-*account-number*-app-zips-*region*
   > - weaver-*lowercase-project-name*-*account-number*-schemas-*region*

3. If you turned on logging for your simulation, delete the CloudWatch Logs log group. For more information about working with CloudWatch Logs, see Working with log groups and log streams in the *Amazon CloudWatch Logs User Guide*.

   The name of the log group for your simulation is specified in its schema (configuration file): *`project-folder`*`\tools\`*`project-name`*`.yaml`

   The name of the log group is the value of `log_destination_resource_name`.
   The following schema snippet shows that the log group for the sample application is `MySimulationLogs`.

```
simulation_properties:
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
  default_entity_index_key_type: "Vector3<f32>"
```

   Warning

   If you start multiple simulations that specify the same log group then the log data for all of those simulations will go to the same log group. If you delete the log group then you delete the log data for all simulations that use that log group. If you delete a log group for a running simulation then the simulation will fail.

   Important

   If your simulation's schema specifies `log_destination_service: "logs"` and a `log_destination_resource_name` but you can't find the log group in CloudWatch Logs, make sure that you check the same AWS Region that your simulation ran in.

Detailed tutorial: Learn the details while building the sample application

This tutorial guides you through the same overall procedure that the quick start tutorial presented, but in more detail. The quick start tutorial simplified many of the steps and hid details within automation. This tutorial exposes and explains those details.

We recommend that you complete the quick start tutorial before going through this tutorial.

Requirements

Before you begin, be sure that you completed the steps in Setting up for SimSpace Weaver.

Steps

Step 1: Create a project

The SimSpace Weaver app SDK distributable contains a script that creates a project from the bundled `PathfindingSample` project. You must run the script from its location in your file system. The script creates a *`project-name`* inside a *`path`* using the values that you provide at the command line.

Docker

To create a project

1. At a **Windows command prompt**, change to your project folder.

```
cd sdk-folder
```

2. Run the `create-project.bat` script.

```
.\create-project.bat --name project-name --path path
```

The full command and parameter list follow.

```
.\create-project.bat --name project-name --path path --app-sdk-version version-number --template template-name --overwriteproject
```

> **Important**
>
> Don't use a project name longer than 20 characters. Exceeding this limit might result in errors.

> **Note**
>
> If you receive a prompt to share a drive with Docker, choose **Yes** to share.

> **Important**
>
> If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile cli-profile-name` parameter. For more information, see Configuring

the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

To create a project

1. At a **Linux shell prompt**, change to your project folder.

```
cd sdk-folder
```

2. Run the `create-project.sh` script.

```
./create-project.sh --name project-name --path path
```

The full command and parameter list follow.

```
./create-project.sh --name project-name --path path --profile cli-profile-name
 --app-sdk-version version-number --template template-name --overwriteproject
```

> **Important**
>
> Don't use a project name longer than 20 characters. Exceeding this limit might result in errors.

> **Important**
>
> If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

Parameters

name

The name of your project.

path

The location of the project in your file system. The project structure can sometimes have long file paths that can exceed the path length limit in your operating system. We recommend that you use a path name that is as short as possible.

profile

The name of the AWS CLI profile that the script should use for authentication. For more information, see Configuration and credential file settings in the *AWS Command Line Interface User Guide*. This parameter is only available for the SimSpace Weaver app SDK version 1.12.1 or higher. For more information about SimSpace Weaver versions, see SimSpace Weaver versions.

app-sdk-version

(optional) The version of the SimSpace Weaver app SDK that your project uses. You will build and link your apps using this version. If the script doesn't find the version in the location of the distributable, or if you don't provide the version number, the script will automatically download the latest version.

template

(optional) A project template that the script will use to create your project. If you don't provide a template, the script will use the `PathfindingSample`. Valid values:

- **`PathfindingSample`** – a sample application that uses a single worker.
- **`MultiWorkerPathfindingSample`** – a version of the sample application that uses multiple workers.

overwriteproject

(optional) Use this option to overwrite an existing project folder with the same project name and path.

Step 2: Turn on logging (optional)

Logging is off by default for the `PathfindingSample` project. This tutorial assumes that logging is on. You can choose to turn on logging, but it is optional.

> **Important**
>
> The `PathfindingSample` generates large amounts of log data. If you choose to turn on logging, you will receive billing charges for the log data. You will continue to receive billing charges for the log data as long as it exists. We highly recommend that you stop this simulation and perform the cleanup steps at the end of the tutorial as soon as possible if you turn on logging.

To turn on logging

1. Open the following file in a text editor:

   Docker

   `project-folder``\tools\`*`project-name`*`-schema.yaml`

   > **Note**
   >
   > Your *`project-folder`* is *`path`*`\`*`project-name`* using the values that you provided when you created the project.

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

project-folder/tools/*project-name*-schema.yaml

> **Note**
>
> Your *project-folder* is *path*/*project-name* using the values that you provided when you created the project.

2. Find the `simulation_properties:` section at the beginning of the file:

```
simulation_properties:
  default_entity_index_key_type: "Vector3<f32>"
```

3. Insert the following 2 lines after the line `simulation_properties:`:

```
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
```

4. Confirm that your `simulation_properties:` section is the same as the following:

```
simulation_properties:
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
  default_entity_index_key_type: "Vector3<f32>"
```

5. Save the file and exit your text editor.

Step 3: Upload your simulation schema

SimSpace Weaver uses a **schema** to configure your simulation. The schema is a YAML-formatted plain text file. For more information, see Configuring your simulation.

Docker

The sample application comes with a preconfigured schema. You can find the sample application's schema file in the tools folder for your project:

```
project-folder\tools\project-name-schema.yaml
```

To upload your schema

1. If you aren't there already, go to the tools folder for your project and platform. Your *project-folder* is *path**project-name* using the values that you provided when you created the project.

   At a **Windows command prompt**, enter:

   ```
   cd project-folder\tools\windows
   ```

2. Use the helper script to upload the schema.

   ```
   .\upload-schema-project-name.bat
   ```

   Important

   If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface*

User Guide and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

The sample application comes with a preconfigured schema. You can find the sample application's schema file in the tools folder for your project:

```
project-folder/tools/project-name-schema.yaml
```

To upload your schema

1. If you aren't there already, go to the tools folder for your project and platform.Your *project-folder* is *path*/*project-name* using the values that you provided when you created the project.

   At a **Linux shell prompt**, enter:

   ```
   cd project-folder/tools/linux
   ```

2. Use the helper script to upload the schema.

   ```
   ./upload-schema-project-name.sh
   ```

   > **Important**
   >
   > If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use

the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

Step 4: Build your project

You're now ready to build the spatial and custom apps for the sample project. The sample project includes a helper script that builds these apps for you.

Docker

The script will launch a Docker container using the Docker image that you created when you set up your local environment. The script runs your build in an Amazon Linux environment in the Docker container. It writes the build artifacts and their dependencies to your *`project-folder`*`\build` folder in Windows.

To build your project

1. If you aren't there already, go to the tools folder for your project and platform. Your *`project-folder`* is *`path`*`\`*`project-name`* using the values that you provided when you created the project.

   At a **Windows command prompt**, enter:

   ```
   cd project-folder\tools\windows
   ```

2. Use the helper script to build the project.

   ```
   .\build-project-name.bat
   ```

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

The script write the build artifacts and their dependencies to your *project-folder*/build folder.

To build your project

1. If you aren't there already, go to the tools folder for your project and platform.Your *project-folder* is *path*/*project-name* using the values that you provided when you created the project.

   At a **Linux shell prompt**, enter:

   ```
   cd project-folder/tools/linux
   ```

2. Use the helper script to build the project.

   ```
   ./build-project-name.sh
   ```

Step 5: Upload apps

The build script packaged your apps as zip files. You must upload these zip files to specific buckets in Amazon Simple Storage Service in order to run your SimSpace Weaver simulation in the cloud. The SimSpace Weaver app SDK provides a helper script to handle the upload.

Docker

To upload your apps

1. If you aren't there already, go to the tools folder for your project and platform. Your *project-folder* is *path**project-name* using the values that you provided when you created the project.

At a **Windows command prompt**, enter:

```
cd project-folder\tools\windows
```

2. Use the helper script to upload your apps.

```
.\upload-app-project-name.bat
```

> **Important**
>
> If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile cli-profile-name` parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

To upload your apps

1. If you aren't there already, go to the tools folder for your project and platform.Your *project-folder* is *path*/*project-name* using the values that you provided when you created the project.

At a **Linux shell prompt**, enter:

```
cd project-folder/tools/linux
```

2. Use the helper script to upload your apps.

```
./upload-app-project-name.sh
```

> **Important**
>
> If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

Check your Amazon S3 resources

You can check your Amazon S3 buckets to make sure that all of the uploads succeeded. For more information on using Amazon S3, see Creating, configuring, and working with Amazon S3 buckets in the *Amazon Simple Storage Service User Guide*.

For the sample application, your schema (that you uploaded in a previous step) and app resources use the following name formats:

- **Schema bucket**: `simspaceweaver-`*`project-name-lowercase`*`-`*`account-number`*`-schemas-`*`region`*
  - **Schema file**: *`project-name`*`-schema.yaml`
- **App bucket**: `simspaceweaver-`*`project-name-lowercase`*`-`*`account-number`*`-app-zips-`*`region`*

- **Spatial app**: *project-name*`Spatial.zip`
- **View (custom) app**: *project-name*`View.zip`

For example, given the following project properties:

- **Project name**: `MyProject`
- **AWS account number**: `111122223333`
- **AWS Region**: `us-west-2`

The schema and app resources would have the following names:

- **Schema bucket**: `simspaceweaver-myproject-111122223333-schemas-us-west-2`
  - **Schema file**: `MyProject-schema.yaml`
- **App bucket**: `simspaceweaver-myproject-111122223333-apps-zips-us-west-2`
  - **Spatial app**: `MyProjectSpatial.zip`
  - **View (custom) app**: `MyProjectView.zip`

Step 6: Start your simulation

The SimSpace Weaver app SDK provides a helper script to start your simulation. The script is a wrapper around a **StartSimulation** API call. It provides the following parameters to the API call:

- The name of the simulation schema (that you uploaded in an earlier step)
- The simulation name
- The SimSpace Weaver service endpoint

Docker

To start your simulation

1. If you aren't there already, go to the tools folder for your project and platform. Your *project-folder* is *path**project-name* using the values that you provided when you created the project.

   At a **Windows command prompt**, enter:

```
cd project-folder\tools\windows
```

2. Use the helper script to upload start your simulation.

```
.\start-simulation-project-name.bat
```

Important

Starting with version 1.12.3, the `start-simulation` script starts your simulation with a maximum duration of 1 hour. You can use the `--maximum-duration` parameter to specify a different maximum duration. In version 1.12.2 or lower, you can't provide a maximum duration to the script and simulations have a maximum duration of 14 days. For more information about the maximum duration of a simulation, see Maximum duration of a simulation.

Important

If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile cli-profile-name` parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

Note

`run-project-name.bat` is an alternate helper script that will also start the simulation clock. For this tutorial, you will start the clock separately in a later step.

Starting with version 1.12.3, the `run` script starts your simulation with a maximum duration of 1 hour. You can use the `--maximum-duration` parameter to specify a different maximum duration. In version 1.12.2 or lower, you can't provide a maximum duration to the script and simulations have a maximum duration of 14 days. For more information about the maximum duration of a simulation, see Maximum duration of a simulation.

WSL

Important

We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

To start your simulation

1. If you aren't there already, go to the tools folder for your project and platform.Your *`project-folder`* is *`path`*`/`*`project-name`* using the values that you provided when you created the project.

   At a **Linux shell prompt**, enter:

   ```
   cd project-folder/tools/linux
   ```

2. Use the helper script to upload start your simulation.

   ```
   ./start-simulation-project-name.sh
   ```

   Important

   Starting with version 1.12.3, the `start-simulation` script starts your simulation with a maximum duration of 1 hour. You can use the `--maximum-duration` parameter to specify a different maximum duration. In version 1.12.2 or lower, you can't provide a maximum duration to the script and simulations have a maximum

duration of 14 days. For more information about the maximum duration of a simulation, see Maximum duration of a simulation.

Important

If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

Note

`run-`*`project-name`*`.sh` is an alternate helper script that will also start the simulation clock. For this tutorial, you will start the clock separately in a later step. Starting with version 1.12.3, the `run` script starts your simulation with a maximum duration of 1 hour. You can use the `--maximum-duration` parameter to specify a different maximum duration. In version 1.12.2 or lower, you can't provide a maximum duration to the script and simulations have a maximum duration of 14 days. For more information about the maximum duration of a simulation, see Maximum duration of a simulation.

The script will loop until the simulation status is either `STARTED` or `FAILED`. It can take a few minutes for a simulation to start. If your simulation starts successfully, you will see output similar to the following:

```
[2022-10-04T22:15:28] [INFO] Describe Simulation Results:
[2022-10-04T22:15:28] [INFO] {
    "Status": "STARTED",
    "Name": "MyProjectSimulation_22-10-04_22_10_15",
    "RoleArn": "arn:aws:iam::111122223333:role/weaver-MyProject-app-role",
    "CreationTime": 1664921418.09,
    "SchemaS3Location": {
        "ObjectKey": "MyProject-schema.yaml",
        "BucketName": "weaver-myproject-111122223333-us-west-2"
    },
```

Note

SimSpace Weaver app SDK version 1.12.x projects use separate buckets for the app .zip files and the schema:

- weaver-*lowercase-project-name*-*account-number*-app-zips-*region*
- weaver-*lowercase-project-name*-*account-number*-schemas-*region*

Step 7: Get simulation details

The SimSpace Weaver app SDK provides a helper script that wraps around the AWS CLI. The script simplifies calls to the AWS CLI by providing the SimSpace Weaver service endpoint. You use this helper script to call the SimSpace Weaver APIs. The **DescribeSimulation** API provides details about your simulation, including its state. A simulation can be in one of the following states:

Simulation life cycle states

1. **`STARTING`** – Initial state after you call **StartSimulation**
2. **`STARTED`** – all spatial apps are launched and healthy
3. **`STOPPING`** – Initial state after you call **StopSimulation**
4. **`STOPPED`** – All compute resources are stopped
5. **`DELETING`** – Initial state after you call **DeleteSimulation**
6. **`DELETED`** – All resources assigned to the simulation are deleted
7. **`FAILED`** – The simulation had a critical error/failure and stopped
8. **`SNAPSHOT_IN_PROGRESS`** – A snapshot is in progress

Docker

To get your simulation details

1. If you aren't there already, go to the tools folder for your project and platform. Your *project-folder* is *path**project-name* using the values that you provided when you created the project.

   At a **Windows command prompt**, enter:

   ```
   cd project-folder\tools\windows
   ```

2. Use the CLI helper script to call the **ListSimulations** API.

   ```
   .\weaver-project-name-cli.bat list-simulations
   ```

   > **Important**
   >
   > If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *cli-profile-name* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

   The script should display details about your each of your simulations, similar to the following:

   ```
   {
       "Status": "STARTED",
       "CreationTime": 1664921418.09,
       "Name": "MyProjectSimulation_22-10-04_22_10_15",
   ```

```
    "Arn": "arn:aws:simspaceweaver:us-west-2:111122223333:simulation/
MyProjectSimulation_22-10-04_22_10_15",
    "TargetStatus": "STARTED"
}
```

3. Call **DescribeSimulation** to get your simulation details. Substitute *simulation-name* with the **Name** of your simulation from the output of the previous step.

```
.\weaver-project-name-cli.bat describe-simulation --simulation simulation-name
```

The script should display more details about the simulation that you specified, similar to the following:

```
{

    "Name": "MyProjectSimulation_22-10-04_22_10_15",
    "ExecutionId": "1a2b3c4d-0ab1-1234-567a-12ab34cd5e6f",
    "Arn": "arn:aws:simspaceweaver:us-west-2:111122223333:simulation/
MyProjectSimulation_22-10-04_22_10_15",
    "RoleArn": "arn:aws:iam::111122223333:role/weaver-MyProject-app-role",
    "CreationTime": 1664921418.09,
    "Status": "STARTED",
    "TargetStatus": "STARTED",
    "SchemaS3Location": {
        "ObjectKey": "MyProject-schema.yaml",
        "BucketName": "weaver-myproject-111122223333-us-west-2"
    },
    "SchemaError": "[]",
    "LoggingConfiguration": {
        "Destinations": [
            {
                "CloudWatchLogsLogGroup": {
                    "LogGroupArn": "arn:aws:logs:us-west-2:111122223333:log-
group:MySimulationLogs"
                }
            }
        ]
    },
    "LiveSimulationState": {
        "Domains": [
```

```
            {
                "Type": "",
                "Name": "MySpatialSimulation",
                "Lifecycle": "Unknown"
            },
            {
                "Type": "",
                "Name": "MyViewDomain",
                "Lifecycle": "ByRequest"
            }
        ],
        "Clocks": [
            {
                "Status": "STARTED",
                "TargetStatus": "STARTED"
            }
        ]
    },
    "MaximumDuration": "1H",
    "StartError": "[]"
}
```


WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

To get your simulation details

1. If you aren't there already, go to the tools folder for your project and platform.Your *project-folder* is *path*/*project-name* using the values that you provided when you created the project.

   At a **Linux shell prompt**, enter:

```
cd project-folder/tools/linux
```

2. Use the CLI helper script to call the **ListSimulations** API.

```
./weaver-project-name-cli.sh list-simulations
```

> **Important**
>
> If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

The script should display details about your each of your simulations, similar to the following:

```
{
    "Status": "STARTED",
    "CreationTime": 1664921418.09,
    "Name": "MyProjectSimulation_22-10-04_22_10_15",
    "Arn": "arn:aws:simspaceweaver:us-west-2:111122223333:simulation/
MyProjectSimulation_22-10-04_22_10_15",
    "TargetStatus": "STARTED"
}
```

3. Call **DescribeSimulation** to get your simulation details. Substitute *`simulation-name`* with the **Name** of your simulation from the output of the previous step.

```
./weaver-project-name-cli.sh describe-simulation --simulation simulation-name
```

The script should display more details about the simulation that you specified, similar to the following:


```
{

    "Name": "MyProjectSimulation_22-10-04_22_10_15",
    "ExecutionId": "1a2b3c4d-0ab1-1234-567a-12ab34cd5e6f",
    "Arn": "arn:aws:simspaceweaver:us-west-2:111122223333:simulation/
MyProjectSimulation_22-10-04_22_10_15",
    "RoleArn": "arn:aws:iam::111122223333:role/weaver-MyProject-app-role",
    "CreationTime": 1664921418.09,
    "Status": "STARTED",
    "TargetStatus": "STARTED",
    "SchemaS3Location": {
        "ObjectKey": "MyProject-schema.yaml",
        "BucketName": "weaver-myproject-111122223333-us-west-2"
    },
    "SchemaError": "[]",
    "LoggingConfiguration": {
        "Destinations": [
            {
                "CloudWatchLogsLogGroup": {
                    "LogGroupArn": "arn:aws:logs:us-west-2:111122223333:log-
group:MySimulationLogs"
                }
            }
        ]
    },
    "LiveSimulationState": {
        "Domains": [
            {
                "Type": "",
                "Name": "MySpatialSimulation",
                "Lifecycle": "Unknown"
            },
            {
                "Type": "",
                "Name": "MyViewDomain",
```

```
                "Lifecycle": "ByRequest"
            }
        ],
        "Clocks": [
            {
                "Status": "STARTED",
                "TargetStatus": "STARTED"
            }
        ]
    },
    "MaximumDuration": "1H",
    "StartError": "[]"
}
```

Step 8: Start custom apps

SimSpace Weaver doesn't manage the lifecycle of custom apps. You must start your custom apps. It's best practice to start your custom apps before you start your simulation clock, but you can start custom apps after you start the clock.

You can use the CLI helper script to call the **StartApp** API to start your custom apps.

> **Important**
>
> If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

Docker

```
.\weaver-project-name-cli.bat start-app --simulation simulation-name --name app-name
 --domain domain-name
```

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

```
./weaver-project-name-cli.sh start-app --simulation simulation-name --name app-name
 --domain domain-name
```

The **StartApp** API call will create and start a new instance of the custom app using the name that you provide. If you provide the name of an app that already exists then you will receive an error. If you want to restart a particular app (instance), you must first stop that app and delete it.

> **Note**
>
> The status of your simulation must be STARTED before you can start custom apps. To check the status of your simulation, see Step 7: Get simulation details.

The sample application provides the `ViewApp` custom app to view your simulation. This app provides you with a static IP address and port number to connect the simulation clients (you will do this in a later step in this tutorial). You can think of a **domain** as a class of apps that have the same executable code and launch options. The **app name** identifies the instance of the app. For more information on SimSpace Weaver concepts, see Key concepts for SimSpace Weaver.

You can use the **DescribeApp** API to check the status of a custom app after you start it.

Docker

```
.\weaver-project-name-cli.bat describe-app --simulation simulation-name --app app-name --domain domain-name
```

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

```
./weaver-project-name-cli.sh describe-app --simulation simulation-name --app app-name --domain domain-name
```

Docker

To start the view app in this tutorial

1. If you aren't there already, go to the tools folder for your project and platform. Your *project-folder* is *path**project-name* using the values that you provided when you created the project.

   At a **Windows command prompt**, enter:

   ```
   cd project-folder\tools\windows
   ```

2. Use the CLI helper script to call **StartApp** for `ViewApp`.

   ```
   .\weaver-project-name-cli.bat start-app --simulation simulation-name --name
    ViewApp --domain MyViewDomain
   ```

3. Call **DescribeApp** to check the status of your custom app.

   ```
   .\weaver-project-name-cli.bat describe-app --simulation simulation-name --app
    ViewApp --domain MyViewDomain
   ```

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

To start the view app in this tutorial

1. If you aren't there already, go to the tools folder for your project and platform.Your *project-folder* is *path*/*project-name* using the values that you provided when you created the project.

   At a **Linux shell prompt**, enter:

   ```
   cd project-folder/tools/linux
   ```

2. Use the CLI helper script to call **StartApp** for `ViewApp`.

   ```
   ./weaver-project-name-cli.sh start-app --simulation simulation-name --name
    ViewApp --domain MyViewDomain
   ```

3. Call **DescribeApp** to check the status of your custom app.

   ```
   ./weaver-project-name-cli.sh describe-app --simulation simulation-name --app
    ViewApp --domain MyViewDomain
   ```

After the status of your custom app (instance) is `STARTED`, the output of **DescribeApp** will include the IP address and port number for that custom app (instance). In the following example output, the IP address is the value of `Address` and the port number is the value of `Actual` in the `EndpointInfo` block.

```
{
    "Status": "STARTED",
    "Domain": "MyViewDomain",
```

```
    "TargetStatus": "STARTED",
    "Simulation": "MyProjectSimulation_22-10-04_22_10_15",
    "LaunchOverrides": {
        "LaunchCommands": []
    },
    "EndpointInfo": {
        "IngressPortMappings": [
            {
                "Declared": 7000,
                "Actual": 4321
            }
        ],
        "Address": "198.51.100.135"
    },
    "Name": "ViewApp"
}
```

Note

The value of `Declared` is the port number that your app code should bind to. The value of `Actual` is the port number that SimSpace Weaver exposes to clients to connect to your app. SimSpace Weaver maps the `Declared` port to the `Actual` port.

Note

You can use the procedure from the quick start tutorial to get the IP address and port number of any started custom app, independent of this workflow.

Step 9: Start the clock

When you first create your simulation, it has a clock but the clock isn't running. When your clock isn't running, your simulation won't update its state. After you start the clock, it will begin sending ticks to your apps. Each tick, your spatial apps step through the entities they own and commit the results to SimSpace Weaver

Note

It can take 30-60 seconds to start the clock.

Important

If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

Docker

To start the clock

1. If you aren't there already, go to the tools folder for your project and platform. Your *`project-folder`* is *`path`**`project-name`* using the values that you provided when you created the project.

   At a **Windows command prompt**, enter:

   ```
   cd project-folder\tools\windows
   ```

2. Use the CLI helper script to call the **StartClock** API.

   ```
   .\weaver-project-name-cli.bat start-clock --simulation simulation-name
   ```

> **Note**
>
> The **StartClock** API uses your *simulation-name*, which you can find using the **ListSimulations** API:
>
> ```
> .\weaver-project-name-cli.bat list-simulations
> ```

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

To start the clock

1. If you aren't there already, go to the tools folder for your project and platform.Your *project-folder* is *path*/*project-name* using the values that you provided when you created the project.

   At a **Linux shell prompt**, enter:

   ```
   cd project-folder/tools/linux
   ```

2. Use the CLI helper script to call the **StartClock** API.

   ```
   ./weaver-project-name-cli.sh start-clock --simulation simulation-name
   ```

   > **Note**
   >
   > The **StartClock** API uses your *simulation-name*, which you can find using the **ListSimulations** API:

```
./weaver-project-name-cli.sh list-simulations
```

Step 10: Check the logs

SimSpace Weaver writes simulation management messages and the console output from your apps to Amazon CloudWatch Logs. For more information on working with logs, see Working with log groups and log streams in the *Amazon CloudWatch Logs User Guide*.

Each simulation that you create has its own log group in CloudWatch Logs. The name of the log group is specified in the simulation schema. In the following schema snippet, the value of `log_destination_service` is `logs`. This means that the value of `log_destination_resource_name` is the name of a log group. In this case, the log group is `MySimulationLogs`.

```
simulation_properties:
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
  default_entity_index_key_type: "Vector3<f32>"
```

You can also use the **DescribeSimulation** API to find the name of the log group for simulation after you start it.

Important

If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS*

Command Line Interface User Guide and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

Docker

```
project-folder\tools\windows\weaver-project-name-cli.bat describe-simulation --
simulation simulation-name
```

WSL

Important

We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

```
project-folder/tools/linux/weaver-project-name-cli.sh describe-simulation --
simulation simulation-name
```

The following example shows the part of the output from **DescribeSimulation** that describes the logging configuration. The name of the log group is shown at the end of the `LogGroupArn`.

```
    "LoggingConfiguration": {
        "Destinations": [
            {
                "CloudWatchLogsLogGroup": {
                    "LogGroupArn": "arn:aws:logs:us-west-2:111122223333:log-
group:MySimulationLogs"
                }
            }
        ]
    },
```

Each simulation log group contains several log streams:

- **Management log stream** – simulation management messages produced by the SimSpace Weaver service.

```
/sim/management
```

- **Errors log stream** – error messages produced by the SimSpace Weaver service. This log stream only exists if there are errors. SimSpace Weaver stores errors written by your apps in their own app log streams (see the following log streams).

```
/sim/errors
```

- **Spatial app log streams** (1 for each spatial app on each worker) – console output produced by spatial apps. Each spatial app writes to its own log stream. The *spatial-app-id* is all characters after the trailing slash at the end of the *worker-id*.

```
/domain/spatial-domain-name/app/worker-worker-id/spatial-app-id
```

- **Custom app log streams** (1 for each custom app instance) – console output produced by custom apps. Each custom app instance writes to its own log stream.

```
/domain/custom-domain-name/app/custom-app-name/random-id
```

- **Service app log streams** (1 for each service app instance) – console output produced by service apps. Each service app writes to its own log stream. The *service-app-id* is all characters after the trailing slash at the end of the *service-app-name*.

```
/domain/service-domain-name/app/service-app-name/service-app-id
```

> **Note**
>
> The sample application doesn't have service apps.

Step 11: View your simulation

The SimSpace Weaver app SDK provides different options to view the sample application. You can use the sample console client if you don't have any local support for Unreal Engine development. The instructions for the Unreal Engine client assume that you are using Windows.

The console client displays a list of entity events as they occur. The client gets the entity event information from the `ViewApp`. If your console client displays the list of events, then it confirms network connectivity with the `ViewApp` and activity in your simulation.

The `PathfindingSample` simulation creates stationary and moving entities on a 2-dimensional plane. The moving entities move around the stationary entities. The Unreal Engine client provides a visualization of the entity events.

Windows console client

Requirements

- Microsoft Windows 10 or higher
- Microsoft Visual Studio 2019 or later, with the *Desktop development with C++* workload installed
- CMake3
- Git

To connect to the sample application with the sample console client

1. In a **command prompt window**, go to the folder for the console client (in the app SDK folder).

   ```
   cd sdk-folder\packaging-tools\clients\PathfindingSampleClients\ConsoleClient
   ```

2. Use CMake3 to create a Visual Studio solution in this folder.

   ```
   cmake .
   ```

   Note

   Make sure to include the space and period at the end.

   Important

   Keep the command prompt window open for further steps.

3. In **Visual Studio**, open the `PathfindingSampleConsoleClient.sln` that you created in the previous step.
4. Select the **RelWithDebInfo** build configuration.
5. Choose **Build** > **Build Solution**.
6. In your previous **command prompt window**, go to the build output folder in the folder for the console client.

```
cd RelWithDebInfo
```

7. Run the client with the IP address and port number of your `ViewApp`.

```
.\ConsoleClient.exe --url tcp://ip-address:port-number
```

Your command prompt window should display numbers for entity update, delete, and create events, similar to the following example output.

> **Note**
>
> The IP addresses and port numbers in the following example output are placeholders. Provide the IP address and port number of your `ViewApp` to the console client. Provide the `Actual` port number if you want to connect to a `ViewApp` that runs in the AWS Cloud. Provide the IP address and port number 127.0.0.1:7000 when you connect to a `ViewApp` that runs on your local system. For more information, see Local development.

```
##PathfindingSample#ViewApp Message Reader##

Added argument url:tcp://198.51.100.135:4321
Some subscription arguments are missing, restoring defaults.

*****
Sample usage without a MoveStrategy:
ConsoleClient --url tcp://198.51.100.135:4321 --subs-center-x 600 --subs-center-
y 500 --subs-radius 50
Sample usage with CircleMoveStrategy:
```

```
ConsoleClient --url tcp://198.51.100.135:4321 --subs-center-x 600 --subs-center-
y 500 --subs-radius 50 --subs-move-strategy circle --circle-center-x 500 --
circle-center-y 500 --circle-speed 0.001

*****
Starting NNG client. NNG version: 1.2.4
Creating socket ...done.
Connecting to View App ... done.
Initiating connection to tcp:// 198.51.100.135:4321 ... done.

Receiving messages ...
[2022-10-04 19:13:00.710] CreateEntity Count: 72
[2022-10-04 19:13:00.756] UpdateEntity Count: 42
[2022-10-04 19:13:00.794] DeleteEntity Count: 72
[2022-10-04 19:13:03.690] CreateEntity Count: 11
[2022-10-04 19:13:03.725] UpdateEntity Count: 2
[2022-10-04 19:13:03.757] UpdateEntity Count: 2
[2022-10-04 19:13:03.790] UpdateEntity Count: 2
```

> **Note**
>
> For troubleshooting guidance, see PathfindingSample console client fails to connect.

8. Press **CTRL+C** to quit the console client.

Linux console client

> **Important**
>
> We provide these instructions for your convenience. They might not work in some Linux environments. These procedures are unsupported.

This procedure assumes that you are working entirely within a Linux environment. You can also view your simulation using clients built in Windows.

Requirements

- CMake3

- C compiler (already included in Amazon Linux 2)
- Git

To connect to the sample application with the sample console client

1. At a **Linux shell prompt**, go to the folder for the console client (in the app SDK folder).

```
cd sdk-folder/packaging-tools/clients/PathfindingSampleClients/ConsoleClient
```

2. Create a build folder.

```
mkdir build
```

3. Go to the build folder.

```
cd build
```

4. Use CMake3 to build the client.

```
cmake3 ../ && cmake3 --build .
```

> **Note**
>
> Make sure to include the space and period at the end.

5. Run the client with the IP address and port number of your `ViewApp`.

```
./ConsoleClient --url tcp://ip-address:port-number
```

Your command prompt window should display numbers for entity update, delete, and create events, similar to the following example output.

> **Note**
>
> The IP addresses and port numbers in the following example output are placeholders. Provide the IP address and port number of your `ViewApp` to the console client. Provide the `Actual` port number if you want to connect to a `ViewApp` that runs in the AWS Cloud. Provide the IP address and port number

127.0.0.1:7000 when you connect to a `ViewApp` that runs on your local system. For more information, see Local development.

```
##PathfindingSample#ViewApp Message Reader##

Added argument url:tcp://198.51.100.135:4321
Some subscription arguments are missing, restoring defaults.

*****
Sample usage without a MoveStrategy:
ConsoleClient --url tcp://198.51.100.135:4321 --subs-center-x 600 --subs-center-
y 500 --subs-radius 50
Sample usage with CircleMoveStrategy:
ConsoleClient --url tcp://198.51.100.135:4321 --subs-center-x 600 --subs-center-
y 500 --subs-radius 50 --subs-move-strategy circle --circle-center-x 500 --
circle-center-y 500 --circle-speed 0.001

*****
Starting NNG client. NNG version: 1.2.4
Creating socket ...done.
Connecting to View App ... done.
Initiating connection to tcp:// 198.51.100.135:4321 ... done.

Receiving messages ...
[2022-10-04 19:13:00.710] CreateEntity Count: 72
[2022-10-04 19:13:00.756] UpdateEntity Count: 42
[2022-10-04 19:13:00.794] DeleteEntity Count: 72
[2022-10-04 19:13:03.690] CreateEntity Count: 11
[2022-10-04 19:13:03.725] UpdateEntity Count: 2
[2022-10-04 19:13:03.757] UpdateEntity Count: 2
[2022-10-04 19:13:03.790] UpdateEntity Count: 2
```

Note

For troubleshooting guidance, see PathfindingSample console client fails to connect.

6. Press **CTRL+C** to quit the console client.

Unreal Engine on Windows

Requirements

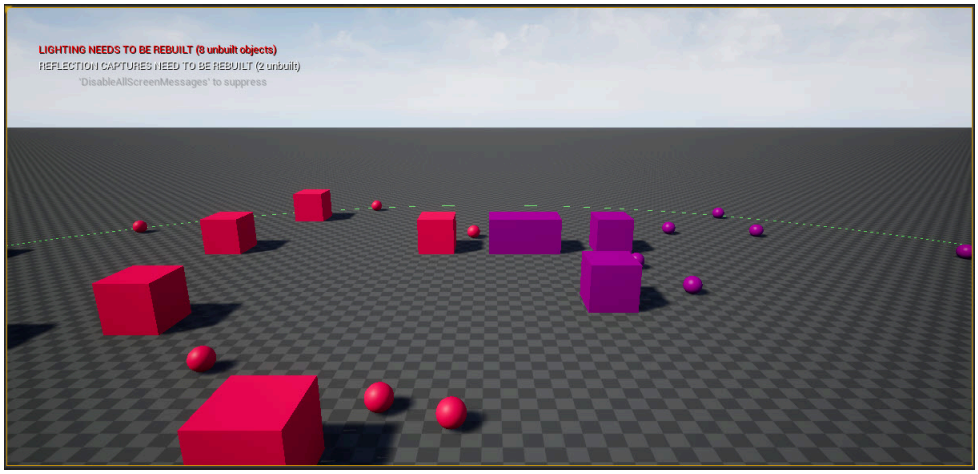
- Unreal Engine 5 development environment
- Microsoft .NET Framework 4.8 Developer Pack
- Windows console client (see the **Windows console client** tab on this page)

> **Important**
>
> Other versions of Unreal Engine and .NET are not supported and might cause problems.

To connect to the sample application with the sample Unreal client

1. The Unreal Engine client uses the NNG library from the console client. You must build the console client for Windows if you haven't already built it. For more information, see the **Windows console client** tab on this page.
2. In a **file manager window**, go to *sdk-folder*\packaging-tools\clients\PathfindingSampleClients\UnrealClient.
3. Open UnrealClient.uproject.
4. If the editor asks you if you want to rebuild the UnrealClient modules, choose **yes**.
5. In a **text editor**, open *sdk-folder*\packaging-tools\clients\PathfindingSampleClients\UnrealClient\view_app_url.txt.
6. Update the URL with the IP address and port number for your view app: tcp://*ip-address*:*port-number* (it should look like tcp://198.51.100.135:1234).
7. In **Unreal editor**, choose **play**.

Your Unreal editor should display a visualization of the simulation, similar to the following screenshot.



> **Note**
>
> Depending on the power of your local development system, it could take a few minutes for the Unreal editor to display the simulation. During this time, the system might appear to freeze.

Use the **W**, **A**, **S**, **D** keys to move in the Unreal client. Hold the mouse button and drag the mouse to turn.

You can press the **[** (left square bracket) key to decrease the size of the subscription area. You can press the **]** (right square bracket) key to increase the size of the subscription area. The size of the subscription area determines the number of entities that appear in the client.

You can press the **C** key to create an entity in the simulation. The client sends a `CreateEntity` command to the view app. The view app will then create the entity and transfer it to the spatial domain.

You can examine the code for `ViewAppDriver::HandleEntityCreationRequests` in *`project-folder`*`\src\PathfindingSample\ViewApp\Driver\ViewAppDriver.cpp` to see how the app implements this process.

> **Note**
>
> If you don't know the IP address and port number of your view app, you can use the procedure from the quick start tutorial to get that information.

Step 12: Stop and clean up your simulation

It's important to clean up your simulations when you don't need them anymore. SimSpace Weaver simulation resources count towards your service quotas (limits), even if your simulation is stopped. You will continue to get billing charges for simulations that are running. You might also get billing charges for data storage in supporting services, such as Amazon CloudWatch Logs and Amazon Simple Storage Service For more information about SimSpace Weaver service quotas, see SimSpace Weaver endpoints and quotas.

Follow the procedures in this section when you are ready to clean up your simulation.

> **Important**
>
> You can't restart a stopped simulation.

> **Important**
>
> You can't recover a deleted simulation.

Clean up simulation resources in SimSpace Weaver

You must stop your simulation before you can delete it. Deleting a simulation removes resources in SimSpace Weaver only. You must perform separate steps to delete resources that your simulation created or uses in other services (see the following section).

Docker

To clean up your simulation

1. If you aren't there already, go to the tools folder for your project and platform. Your *project-folder* is *path**project-name* using the values that you provided when you created the project.

   At a **Windows command prompt**, enter:

   ```
   cd project-folder\tools\windows
   ```

2. Find the names of your simulations.

```
.\weaver-project-name-cli.bat list-simulations
```

> **Important**
>
> If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile cli-profile-name` parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

3. Stop a simulation.

```
.\weaver-project-name-cli.bat stop-simulation --simulation simulation-name
```

4. Delete a stopped simulation.

```
.\weaver-project-name-cli.bat delete-simulation --simulation simulation-name
```

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

To clean up your simulation

1. If you aren't there already, go to the tools folder for your project and platform.Your *project-folder* is *path*/*project-name* using the values that you provided when you created the project.

   At a **Linux shell prompt**, enter:

   ```
   cd project-folder/tools/linux
   ```

2. Find the names of your simulations.

   ```
   ./weaver-project-name-cli.sh list-simulations
   ```

   > **Important**
   >
   > If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

3. Stop a simulation.

   ```
   ./weaver-project-name-cli.sh stop-simulation --simulation simulation-name
   ```

4. Delete a stopped simulation.

   ```
   ./weaver-project-name-cli.sh delete-simulation --simulation simulation-name
   ```

AWS Management Console

To clean up your simulation

1. Open the SimSpace Weaver console at SimSpace Weaver console.
2. From the navigation pane, choose **Simulations**.
3. From the **Simulations** list, select the option next to the name of the simulation that you want to delete.
4. If the **Status** of the simulation that you selected is **STARTED**:
   a. Choose the **Actions** drop-down menu.
   b. Choose **Stop**.
   c. To confirm, enter your simulation name.
   d. Choose **Stop**.
   e. Wait until the **Status** of your simulation is **STOPPED**.
5. Choose the **Actions** drop-down menu.
6. Choose **Delete**.
7. To confirm, choose **Delete**.

Clean up simulation resources in supporting services

To support your simulation, SimSpace Weaver creates resources in other services. SimSpace Weaver doesn't delete these resources when you delete your simulation. You can delete these additional resources if you don't need them.

> ⚠ **Important**
>
> You might get billing charges for any of these resources that you don't delete.

To delete support resources for your project

1. If you are done with your **project**, delete its CloudFormation stack. For more information about working with AWS CloudFormation, see Deleting a stack on the AWS CloudFormation console in the *AWS CloudFormation User Guide*.
   - weaver-*`project-name`*-stack

> **Important**
>
> Simulations that you started from the same project share resources such as the app role. When you delete the CloudFormation stack, you delete the app role. Don't delete your CloudFormation stack if you have other simulations that share the same resources.

> **Note**
>
> Your CloudFormation stack will probably report `DELETE_FAILED` because it can't delete Amazon S3 buckets that aren't empty. You will delete your Amazon S3 buckets in the following step.

2. If you are done with your **project**, delete its Amazon S3 bucket(s). For more information on working with Amazon S3 buckets, see Deleting a bucket in the *Amazon Simple Storage Service User Guide*.

   - `weaver-`*`lowercase-project-name`*`-`*`account-number`*`-`*`region`*

   For example, the project named `MyProject` in the account `111122223333` in the `us-west-2` Region has the following bucket:

   - `weaver-myproject-111122223333-us-west-2`

   > **Note**
   >
   > You must delete the contents of an Amazon S3 bucket before you can delete the bucket.

   > **Note**
   >
   > SimSpace Weaver app SDK version 1.12.x projects use separate buckets for the app .zip files and the schema:

- weaver-*lowercase-project-name*-*account-number*-app-zips-*region*
- weaver-*lowercase-project-name*-*account-number*-schemas-*region*

3. If you turned on logging for your simulation, delete the CloudWatch Logs log group. For more information about working with CloudWatch Logs, see Working with log groups and log streams in the *Amazon CloudWatch Logs User Guide*.

   The name of the log group for your simulation is specified in its schema (configuration file): *project-folder*\tools*project-name*.yaml

   The name of the log group is the value of `log_destination_resource_name`.
   The following schema snippet shows that the log group for the sample application is `MySimulationLogs`.

```
simulation_properties:
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
  default_entity_index_key_type: "Vector3<f32>"
```

   Warning

   If you start multiple simulations that specify the same log group then the log data for all of those simulations will go to the same log group. If you delete the log group then you delete the log data for all simulations that use that log group. If you delete a log group for a running simulation then the simulation will fail.

   Important

   If your simulation's schema specifies `log_destination_service: "logs"` and a `log_destination_resource_name` but you can't find the log group in CloudWatch Logs, make sure that you check the same AWS Region that your simulation ran in.

Working with SimSpace Weaver

This chapter provides information and guidance to help you build your own applications in SimSpace Weaver.

Topics

Configuring your simulation

A **simulation schema** (or **schema**) is a YAML-formatted text file that specifies the configuration for a simulation. You can use the same schema to start multiple simulations. The schema file is located in the project folder for your simulation. You can use any text editor to edit the file. SimSpace Weaver only reads your schema when it starts the simulation. Any edits that you make to a schema file only affect new simulations that you start after the edits.

Docker

To configure your simulation, edit your simulation schema file:

```
project-folder\tools\project-name-schema.yaml
```

You upload the simulation schema when you create a new simulation. The quick start helper script for your project will upload the schema as part of its process to build your simulation:

```
project-folder\tools\windows\quick-start-project-name-cli.bat
```

You can also use the upload schema helper script for your project if you aren't using the quick start script to build your simulation:

```
project-folder\tools\windows\upload-schema-project-name.bat
```

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

You configure your simulation by editing its simulation schema file:

```
project-folder/tools/project-name-schema.yaml
```

You upload the simulation schema when you create a new simulation. The quick start helper script for your project will upload the schema as part of its process to build your simulation:

```
project-folder/tools/linux/quick-start-project-name-cli.sh
```

You can also use the upload schema helper script for your project if you aren't using the quick start script to build your simulation:

```
project-folder/tools/linux/upload-schema-project-name.sh
```

Important

If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

Simulation configuration parameters

The simulation schema contains bootstrapping information, including:

- **Simulation properties** – SDK version and compute configuration (type and number of workers)
- **Clocks** – tick rate and tolerances
- **Spatial partitioning strategies** – spatial topology (such as a grid), bounds, and placement groups (spatial partition grouping on workers)
- **Domains and their apps** – app bucket, path, and launch command(s)

SimSpace Weaver uses your schema configuration to configure and arrange spatial partitions, launch apps, and advance the simulation at your specified tick rate.

Note

The create-project script in the SimSpace Weaver app SDK will automatically generate a simulation schema for you, based on the sample application.

The following topics describe the parameters in the simulation schema. For a full description of the simulation schema, see SimSpace Weaver simulation schema reference.

Topics

SDK version

The `sdk_version` field specifies the version of SimSpace Weaver that the schema is formatted for. Valid values: `1.15`, `1.14`, `1.13`, `1.12`

> **Important**
>
> The value of `sdk_version` only includes the major version number and first minor version number. For example, the value `1.12` specifies all versions `1.12.x`, such as `1.12.0`, `1.12.1`, and `1.12.2`.

Simulation properties

The `simulation_properties` section of your schema specifies the logging configuration and a data type for the index field (usually the spatial location) of entities.

```
simulation_properties:
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
  default_entity_index_key_type: "Vector3<f32>"
```

The value of `log_destination_service` determines the interpretation of the value of `log_destination_resource_name`. Currently, the only supported value is `logs`. This means that the value of `log_destination_resource_name` is the name of a log group in Amazon CloudWatch Logs

> **Note**
>
> Logging is optional. If you don't configure log destination properties then your simulation won't produce logs.

The `default_entity_index_key_type` property is required. The only valid value is `Vector3<f32>`.

Workers

The `workers` section specifies the type and number of workers that you want for your simulation. SimSpace Weaver uses its own worker types that map to Amazon EC2 instance types.

```
workers:
  MyComputeWorkers:
    type: "sim.c5.24xlarge"
    desired: 1
```

Enabling multi-worker simulations

You can create a simulation that uses more than 1 worker. By default, simulations use 1 worker. You must modify your simulation schema before you start the simulation.

> **Note**
>
> You can't change a simulation that has already started. If you want to enable multi-worker for a running simulation, you must stop and delete the simulation first.

To use more than one worker, set the `desired` number of compute instances to a value greater than 1. There is a maximum number of apps for each worker. For more information, see SimSpace Weaver endpoints and quotas. SimSpace Weaver will only use more than 1 worker when the number of apps on a worker exceeds this limit. SimSpace Weaver can place an app on any of the available workers. App placement on a specific worker isn't guaranteed.

The following schema snippet demonstrates a configuration for a simulation that requests 2 workers. SimSpace Weaver will attempt to allocate the second worker if the number of apps exceeds the maximum number of apps for 1 worker.

```
workers:
  MyComputeWorkers:
    type: "sim.c5.24xlarge"
    desired: 2
```

Clock

The `clock` section specifies properties of the simulation clock. Currently, you can only configure the **tick rate** (the number of ticks per second that the clock sends to apps). The tick rate is a maximum rate. The effective tick rate could be lower because all operations (such as entity updates) for a tick must finish before the next tick can start. The tick rate is also called the **clock rate**.

The valid values for `tick_rate` depend on the `sdk_version` specified in your schema.

Valid values for the tick rate

- Versions earlier than `"1.14"`:
  - `10`
  - `15`
  - `30`
- Version `"1.14"` or later:
  - `"10"`
  - `"15"`
  - `"30"`
  - `"unlimited"`

    For more information, see Unlimited tick rate.

Important

- For schemas with a `sdk_version` earlier than `"1.14"` the value of `tick_rate` is an **integer**, such as `30`.
- For schemas with a `sdk_version` of `"1.14"` or later, the value of `tick_rate` is a **string**, such as `"30"`. The value **must include the double quotes**.

  If you convert a version `"1.12"` or `"1.13"` schema to version `"1.14"` or later, you must enclose the value of `tick_rate` in double quotes.

Unlimited tick rate

You can set the `tick_rate` to `"unlimited"` to enable your simulation to run as fast as your code can execute. With an unlimited tick rate, SimSpace Weaver sends the next tick immediately after all apps finish the commits for the current tick.

Important

Unlimited tick rate isn't supported in SimSpace Weaver versions before 1.14.0. The minimum value of `sdk_version` in the schema is `"1.14"`.

Unlimited tick rate in SimSpace Weaver Local

SimSpace Weaver Local implements `"unlimited"` as if the schema specified a tick rate of 10 kHz (10000). The effect is the same as an unlimited tick rate in the AWS Cloud. You still specify `tick_rate: "unlimited"` in your schema. For more information about SimSpace Weaver Local, see Local development.

Frequently asked questions about the clock

Q1. Can I change a STARTED simulation to use a different tick rate?

You can't change the tick rate of a simulation that already exists in the AWS Cloud at any stage of its life cycle. You also can't change the tick rate of a simulation running in SimSpace Weaver Local. You can set the `tick_rate` in the schema and start a new simulation from that schema.

Q2. Can I run my simulation with an unlimited tick rate in a version earlier than 1.14?

No, unlimited tick rate isn't supported in versions before 1.14.0.

Troubleshooting clock errors

If your simulation fails to start, you can check the value of `"StartError"` in the output of the **DescribeSimulation** API. An invalid `tick_rate` value in your schema will produce the following errors.

> **Note**
>
> The error output shown here is displayed on multiple lines to improve readability. The actual error output is a single line.

- The `sdk_version` is earlier than `"1.14"` and the value of `tick_rate` is an invalid integer. Valid values: `10`, `15`, `30`

```
"[{\"errorType\":\"SchemaFormatInvalid\",\"errorMessage\":
    \"$.clock.tick_rate: does not have a value in the enumeration [10, 15, 30]\"}]"
```

- The `sdk_version` is earlier than `"1.14"` and the value of `tick_rate` is a string. Valid values: `10`, `15`, `30`

```
"[{\"errorType\":\"SchemaFormatInvalid\",\"errorMessage\":
    \"$.clock.tick_rate: does not have a value in the enumeration [10, 15, 30]\"},
    {\"errorType\":\"SchemaFormatInvalid\",
    \"errorMessage\":\"$.clock.tick_rate: string found, integer expected\"}]"
```

- The `sdk_version` is `"1.14"` or later and the value of `tick_rate` is an invalid string. Valid values: `"10"`, `"15"`, `"30"`, `"unlimited"`

```
"[{\"errorType\":\"SchemaFormatInvalid\",\"errorMessage\":
    \"$.clock.tick_rate: does not have a value in the enumeration [10, 15, 30,
 unlimited]\"}]"
```

- The `sdk_version` is `"1.14"` or later and the value of `tick_rate` is an integer. Valid values: `"10"`, `"15"`, `"30"`, `"unlimited"`

```
"[{\"errorType\":\"SchemaFormatInvalid\",\"errorMessage\":
```

```
    \"$.clock.tick_rate: does not have a value in the enumeration [10, 15, 30,
 unlimited]\"},
    {\"errorType\":\"SchemaFormatInvalid\",
    \"errorMessage\":\"$.clock.tick_rate: integer found, string expected\"}]"
```

Partitioning strategies

The `partitioning_strategies` section specifies configuration properties for the partitions of spatial apps. You provide your own name for a partitioning strategy (a set of properties in this section) and use it in your spatial app configuration.

```
partitioning_strategies:
  MyGridPartitioning:
    topology: "Grid"
    aabb_bounds:
      x: [0, 1000]
      y: [0, 1000]
    grid_placement_groups:
      x: 1
      y: 1
```

The `topology` property specifies the type of coordinate system that your simulation uses. The value `Grid` specifies a 2-dimensional (2D) grid.

For a `Grid` topology, the simulation space is modeled as an axis-aligned bounding box (AABB). You specify the coordinate bounds for each axis of your AABB in the `aabb_bounds` property. All entities that exist spatially in your simulation must have a position inside the AABB.

Grid placement groups

A **placement group** is a collection of spatial app partitions that you want SimSpace Weaver to place on the same worker. You specify the number and arrangement of placement groups (in a grid) in the `grid_placement_groups` property. SimSpace Weaver will attempt to evenly distribute the partitions across the placement groups. The ownership areas of spatial apps with partitions in the same placement group will be spatially adjacent.

We recommend that x * y is equal to your desired number of workers. If it isn't equal, SimSpace Weaver will attempt to balance your placement groups across the available workers.

If you don't specify a placement group configuration, SimSpace Weaver will calculate one for you.

Domains

You provide a name for a set of configuration properties for a domain. The launch setting for apps in a domain determines the type of domain:

- **`launch_apps_via_start_app_call`** – custom domain
- **`launch_apps_by_partitioning_strategy`** – spatial domain
- **`launch_apps_per_worker`** (not included in the sample application) – service domain

> **Important**
>
> SimSpace Weaver supports up to 5 domains for each simulation. This includes all spatial, custom, and service domains.

```
domains:
  MyViewDomain:
    launch_apps_via_start_app_call: {}
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MyViewApp.zip"
      launch_command: ["MyViewApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 7000
  MySpatialDomain:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 2
        y: 2
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MySpatialApp.zip"
      launch_command: ["MySpatialApp"]
      required_resource_units:
        compute: 1
```

Note

SimSpace Weaver app SDK version 1.12.x projects use separate buckets for the app .zip files and the schema:

- weaver-*lowercase-project-name*-*account-number*-app-zips-*region*
- weaver-*lowercase-project-name*-*account-number*-schemas-*region*

Topics

App configuration

You specify the configuration of an app (`app_config`) as part of the configuration for its domain. All types of domains use these same app configuration properties.

```
app_config:
  package: "s3://weaver-myproject-111122223333-us-west-2/MyViewApp.zip"
  launch_command: ["MyViewApp"]
  required_resource_units:
    compute: 1
```

Note

SimSpace Weaver app SDK version 1.12.x projects use separate buckets for the app .zip files and the schema:

- weaver-*lowercase-project-name*-*account-number*-app-zips-*region*
- weaver-*lowercase-project-name*-*account-number*-schemas-*region*

The `package` property specifies the S3 URI of a zip file in an S3 bucket. The zip file contains the app executable (also called a *binary*) and any other resources that it needs (such as libraries). Each instance of the app executable runs in a Docker container on a worker.

The `launch_command` property specifies the name of the executable and any command-line options to run the app. The value of `launch_command` is an array. Each token of the entire launch command string is an element in the array.

Example

- For the launch command: `MyTestApp --option1 value1`
- Specify: `launch_command: ["MyTestApp", "-option1", "value1"]`

The `required_resource_units` property specifies the number of compute resource units that SimSpace Weaver should allocate to this app. A compute resource unit is a fixed amount of processing capacity (vCPU) and memory (RAM) on a worker. You can increase this value to increase the amount of computing power available to the app when it runs on a worker. There is a limited number of compute resource units on each worker. For more information, see SimSpace Weaver endpoints and quotas.

Configuring spatial domains

For spatial domains, you must specify a `partitioning_strategy`. The value of this property is the name that you gave to a partitioning strategy that you defined in another part of the schema.

```
MySpatialDomain:
  launch_apps_by_partitioning_strategy:
    partitioning_strategy: "MyGridPartitioning"
    grid_partition:
      x: 2
      y: 2
  app_config:
    package: "s3://weaver-myproject-111122223333-us-west-2/MySpatialApp.zip"
    launch_command: ["MySpatialApp"]
    required_resource_units:
      compute: 1
```

> **Note**
>
> SimSpace Weaver app SDK version 1.12.x projects use separate buckets for the app .zip files and the schema:
>
> - weaver-*lowercase-project-name*-*account-number*-app-zips-*region*
> - weaver-*lowercase-project-name*-*account-number*-schemas-*region*

A partitioning strategy with a `Grid` topology (the only topology supported in this release) directs SimSpace Weaver to arrange spatial app partitions of this domain in a grid. The `grid_partition` property specifies the number rows and columns of the partition grid.

SimSpace Weaver will start 1 instance of the spatial app for each cell in the partition grid. For example, if a spatial domain has `grid_partition` values `x: 2` and `y: 2`, there are 2 * 2 = 4 partitions in the spatial domain. SimSpace Weaver will start 4 instances of the app configured in the spatial domain and assign 1 partition to each app instance.

Topics

Resource requirements for spatial domains

You can assign up to 17 compute resource units for each worker. You specify the number of compute resource units that each spatial app uses in the `app_config` section of your spatial domain.

Example schema snippet showing the compute resource units for a spatial app

```
MySpatialDomain:
  launch_apps_by_partitioning_strategy:
    partitioning_strategy: "MyGridPartitioning"
    grid_partition:
      x: 2
```

```
        y: 2
    app_config:
      package: "s3://weaver-myproject-111122223333-artifacts-us-west-2/
MySpatialApp.zip"
      launch_command: ["MySpatialApp"]
      required_resource_units:
        compute: 1
```

To calculate the number of compute resource units that a domain requires, multiply the number of cells in your grid (in your `grid_partition`, x * y) by the number of compute resource units assigned to the spatial apps.

For the previous example, the domain `MySpatialDomain` specifies:

- x: 2
- y: 2
- `compute`: 1

The grid for `MySpatialDomain` has 2 * 2 = 4 cells. The spatial domain requires 4 * 1 = 4 compute resource units.

The total number of compute resource units for all domains specified in your schema must be less than or equal to the `desired` number of workers multiplied by the maximum number of compute resource units for each worker (17).

Multiple spatial domains

You can configure your simulation to use more than 1 spatial domain. For example, you can use 1 spatial domain to control the main actors in a simulation (such as people and cars) and a different spatial domain to control the environment.

You can also use multiple spatial domains to assign different resources to different parts of your simulation. For example, if your simulation has a type of entity that has 10x more entity instances than another type, you can create different domains to handle each entity type and allocate more resources for the domain with more entities.

> **Important**
>
> SimSpace Weaver versions before 1.14.0 don't support multiple spatial domains.

Important

AWS SimSpace Weaver Local doesn't currently support multiple spatial domains. For more information about SimSpace Weaver Local, see Local development.

Important

SimSpace Weaver supports up to 5 domains for each simulation. This includes all spatial, custom, and service domains.

Configure multiple spatial domains

To configure more than 1 spatial domain, add the other spatial domain definitions as separate named sections in your schema. Each domain must specify the `launch_apps_by_partitioning_strategy` key. See the following example schema.

```
sdk_version: "1.14"
workers:
  MyComputeWorkers:
    type: "sim.c5.24xlarge"
    desired: 1
clock:
  tick_rate: "30"
partitioning_strategies:
  MyGridPartitioning:
    topology: Grid
    aabb_bounds:
      x: [0, 1000]
      y: [0, 1000]
domains:
  MySpatialDomain:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 2
        y: 2
    app_config:
      package: "s3://weaver-myproject-111122223333-artifacts-us-west-2/
MySpatialApp.zip"
```

```
      launch_command: ["MySpatialApp"]
      required_resource_units:
        compute: 1
  MySecondSpatialDomain:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 2
        y: 2
    app_config:
      package: "s3://weaver-myproject-111122223333-artifacts-us-west-2/
MySpatialApp2.zip"
      launch_command: ["MySpatialApp2"]
      required_resource_units:
        compute: 1
```

Placing spatial domains together

In some scenarios, you might want to place partitions for a spatial domain on workers next to partitions from another domain. This can improve performance characteristics if those partitions create cross-domain subscriptions to each other.

Add the top level key `placement_constraints` to your schema to specify which domains SimSpace Weaver should place together. The required on_`workers` key must refer to a named `workers` configuration in the schema.

Example schema snippet showing spatial domains placed together

```
workers:
  MyComputeWorkers:
    type: "sim.c5.24xlarge"
    desired: 2
placement_constraints:
  - placed_together: ["MySpatialDomain", "MySecondSpatialDomain"]
    on_workers: ["MyComputeWorkers"]
```

⚠ Important

- If you use placement groups:
  - Make sure that x * y is a multiple of the number of workers.

- Make sure that the placement group values are common divisors for the grid dimensions of the domains you place together.

- If you **don't use** placement groups:
  - Make sure that 1 axis of your spatial domain grids has a common divisor that is equal to the number of workers.

For more information about placement groups, see Partitioning strategies.

Frequently asked questions about spatial domains

Q1. How can I add another spatial domain to an existing simulation?

- **For a running simulation** – You can't change the configuration for a running simulation. Change the domain configuration in the schema, upload the schema and app zips, and start a new simulation.
- **For a new simulation** – Add the domain configuration to the schema, upload the schema and app zips, and start the new simulation.

Troubleshooting spatial domains

The might get the following error when you try to start your simulation but your domain configuration is invalid.

```
"StartError": "[{\"errorType\":\"SchemaFormatInvalid\",\"errorMessage\":
    \"We were unable to determine an arrangement of your domains that would fit
    within the provided set of workers. This can generally be resolved by
    increasing the number of workers if able, decreasing your domains\u0027
    [\u0027\u0027grid_partition\u0027\u0027] values, or adjusting the
    dimensions of your [\u0027\u0027grid_placement_groups\u0027\u0027].\"}]"
```

Potential causes

- The schema allocates more compute resource units for apps than are available on workers.
- SimSpace Weaver can't determine an arrangement to place domains together on workers. This happens when you specify multiple spatial domains but there isn't a common divisor or multiple between domain grids, such as between a 2x4 grid and a 3x5 grid).

Network endpoints

Custom and service apps can have network endpoints that external clients can connect to. You specify a list of port numbers as the value for `ingress_ports` in the `endpoint_config`. These port numbers are both TCP and UDP. The custom or service app should bind to the port numbers that you specify in `ingress_ports`. SimSpace Weaver dynamically allocates port numbers at runtime and maps these ports to the dynamic ports. You can call the **describe-app** API after your apps have started to find the dynamic (actual) port numbers. For more information, see Step 4: Get your IP address and port number from the quick start tutorial.

```
domains:
  MyViewDomain:
    launch_apps_via_start_app_call: {}
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MyViewApp.zip"
      launch_command: ["MyViewApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 7000
```

Note

SimSpace Weaver app SDK version 1.12.x projects use separate buckets for the app .zip files and the schema:

- weaver-*lowercase-project-name*-*account-number*-app-zips-*region*
- weaver-*lowercase-project-name*-*account-number*-schemas-*region*

Note

`endpoint_config` is an optional property for custom apps and service apps. If you don't specify an `endpoint_config` then the app won't have a network endpoint.

Configuring service domains

The presence of `launch_apps_per_worker:` in a domain configuration indicates that it is a service domain that has service apps. SimSpace Weaver starts and stops service apps for you. When SimSpace Weaver starts and stops an app, the app is considered to have a *managed lifecycle* . SimSpace Weaver currently supports starting 1 or 2 service apps on each and every worker.

Example of a domain configured to launch 1 service app on each worker

```
domains:
  MyServiceDomain:
    launch_apps_per_worker:
      count: 1
    app_config:
      package: "s3://weaver-myproject-111122223333-app-zips-us-west-2/
PlayerConnectionServiceApp.zip"
      launch_command: ["PlayerConnectionServiceApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 9000
          - 9001
```

Example of a domain configured to launch 2 service apps on each worker

```
domains:
  MyServiceDomain:
    launch_apps_per_worker:
      count: 2
    app_config:
      package: "s3://weaver-myproject-111122223333-app-zips-us-west-2/
PlayerConnectionServiceApp.zip"
      launch_command: ["PlayerConnectionServiceApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 9000
          - 9001
```

Maximum duration of a simulation

Each simulation in AWS SimSpace Weaver has a *maximum duration* setting that specifies the maximum time that the simulation can run. You provide the maximum duration as a parameter when you start a simulation. The `StartSimulation` application programming interface (API) has an optional parameter `MaximumDuration`. The value of the parameter is a number of minutes (m or M), hours (h or H), or days (d or D). For example, `1h` or `1H` means 1 hour. SimSpace Weaver stops your simulation when it reaches this limit.

Maximum value

The highest valid value for `MaximumDuration` is `14D`, or its equivalent in hours (`336H`) or minutes (`20160M`).

Default value

The `MaximumDuration` parameter is optional. If you don't provide a value, SimSpace Weaver uses a value of `14D`.

Minimum value

The lowest valid value for `MaximumDuration` is a value that is numerically equivalent to `0`. For example, the values `0M`, `0H`, and `0D`, are all numerically equivalent to `0`.

If you provide the minimum value for maximum duration, your simulation immediatly transitions to the `STOPPING` state as soon as it reaches the `STARTED` state.

Starting a simulation using SimSpace Weaver app SDK scripts

You can provide a value for the `maximum-duration` parameter when you use one of the following scripts to start a simulation:

- `quick-start-`*`project-name`*`-cli.bat --maximum-duration` *`value`*
- `start-simulation-`*`project-name`*`.bat --maximum-duration` *`value`*
- `run-`*`project-name`*`.bat --maximum-duration` *`value`*

Each script passes the value of `maximum-duration` to the `StartSimulation` API.

Important

If you don't provide a value for `maximum-duration`, SimSpace Weaver uses the default value (`14D`).

Starting a simulation using the console

You can provide a value for the **Maximum duration** when you start a simulation in the SimSpace Weaver console. Enter the value in the **Maximum duration** field of the **Simulation settings** form when you choose **Start simulation**.

Important

If you don't provide a value for **Maximum duration**, SimSpace Weaver uses the default value (`14D`).

The status of a simulation that reaches its maximum duration

When SimSpace Weaver automatically stops a simulation that reaches its maximum duration, the **status** of the simulation is `STOPPING` (if in progress) or `STOPPED`. In the SimSpace Weaver console, the **target status** of the simulation is still `STARTED`, because that was the last state requested by a user.

Developing apps

SimSpace Weaver development requires an Amazon Linux 2 (AL2) environment to build apps because your simulations run on Amazon Linux in the AWS Cloud. If you're using Windows, you can use scripts in the SimSpace Weaver app SDK to create and launch a Docker container that runs AL2 with the dependencies that you need to build SimSpace Weaver apps. You can also launch an AL2 environment using Windows Subsystem for Linux (WSL), or use a native AL2 system. For more information, see Set up your local environment for SimSpace Weaver.

> **Note**
>
> Regardless of how you configure your local development environment, your apps run in Docker containers when you upload them to run in the AWS Cloud. **Your apps don't have direct access to the host operating system**.

General flow of a SimSpace Weaver app

1. Create an application.
2. Loop:
   a. Begin the update by creating a `Transaction`.
      - Exit the loop if the simulation is shutting down.
   b. Process subscription and ownership entity events.
   c. Update the simulation.
   d. Commit the `Transaction` to end the update.
3. Destroy the application.

Spatial apps

Each spatial app has an ownership area that is a spatial region of the simulation world. Entities located in a spatial app's ownership area are stored in the app's assigned partition. The single spatial app has full ownership (read and write permissions) over all entities within its assigned partition. No other apps can write to those entities. The spatial app advances the state of its entities. Each spatial app owns only 1 partition. SimSpace Weaver uses the spatial location of an entity to index and assign it to a spatial app partition.

The SimSpace Weaver app SDK provides a sample application. You can find the source code for the spatial app of the sample application in the following folder:

Docker

```
project-folder\src\PathfindingSample\SpatialApp
```

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

```
project-folder/src/PathfindingSample/SpatialApp
```

Custom apps

You create and use custom apps to interact with the simulation.

Custom apps can

- Create entities
- Subscribe to other partitions
- Commit changes

General flow of a custom app

1. Create an application.
2. Subscribe to a specific region in the simulation:
    a. Create a `Transaction` to begin the first update.
    b. Create a subscription for the specific region.
    c. Commit the `Transaction` to end the first update.
3. Loop:
    a. Create a `Transaction` to begin the update.
        - Exit the loop if the simulation is shutting down.
    b. Process state changes.
    c. Commit the `Transaction` to end the update.

4. Destroy the application.

After a custom app creates an entity, it must transfer the entity to a spatial domain in order for the entity to exist spatially within the simulation. SimSpace Weaver uses the entity's spatial location to place the entity in the appropriate spatial app partition. The custom app that created the entity can't update or delete the entity after transferring it to a spatial domain.

The SimSpace Weaver app SDK provides a sample application. You can use the custom apps included in the sample application as models for your own custom apps. You can find the source code for the view app (a custom app) of the sample application in the following folder:

Docker

```
project-folder\src\PathfindingSample\ViewApp
```

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

```
project-folder/src/PathfindingSample/ViewApp
```

Developing client applications

Some of the reasons you might want to connect a client to a simulation include:

- Inject real-time traffic information into a city-scale simulation.
- Create *human-in-the-loop* simulations, where a human operator controls some aspect of the simulation.
- Make it possible for users to interact with the simulation, such as for a training simulation.

The custom apps in these examples act as the interface between the simulation state and the outside world. Clients connect to the custom apps to interact with the simulation.

SimSpace Weaver doesn't handle the client applications and their communication with your custom apps. You're responsible for the design, creation, operation, and security of your client applications and their communication with your custom apps. SimSpace Weaver only exposes an IP address and port number for each of your custom apps so that clients can connect to them.

The SimSpace Weaver app SDK provides clients for its sample application. You can use these clients as models for your own client applications. You can find the source code for the sample application clients in the following folder:

Docker

```
sdk-folder\packaging-tools\clients\PathfindingSampleClients
```

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

```
sdk-folder/packaging-tools/clients/PathfindingSampleClients
```

For more information about building and using the sample application clients, see Step 5: View your simulation of the quick start tutorial in this guide.

Local development

You can deploy your SimSpace Weaver applications locally for rapid testing and debugging. SimSpace Weaver Local is supported for building on Microsoft Windows only.

Important

For information about developing with Unity and Unreal Engine, see Support for other engines.

Important

If you are working in version 1.15.3 with C++, Python, Unity, or Unreal Engine, see Local development differences in version 1.15.3.

Requirements

- Microsoft Windows 10 or higher
- Microsoft Visual Studio 2019 or later, with the *Desktop development with C++* workload installed

Topics

Build your simulation for SimSpace Weaver Local

To learn to use SimSpace Weaver Local, you can use the same Pathfinding Sample application that you ran in the cloud during the Getting started with SimSpace Weaver tutorials, but this time on your local hardware.

To build the sample application for SimSpace Weaver Local

1. At a **command prompt**, go to *`project-folder`*`\tools\local`.
2. Run `generate_visual_studio_project.bat`.

3. Open *project-folder*\buildlocal\PathfindingSampleLocal.sln in Visual Studio.
4. Set your build configuration to **RelWithDebInfo**.
5. Choose **Build** > **Build Solution**.

   Visual Studio will put your build artifacts in:

   - *project-folder*\buildlocal\out\RelWithDebInfo.

   Inside that folder, you should see the following executable files:

   - PathfindingSampleLocalSpatial.exe
   - PathfindingSampleLocalView.exe

Run your simulation with SimSpace Weaver Local

You can use SimSpace Weaver Local to run any combination of up to 24 spatial or custom apps on your local computer. The simulation clock starts after all spatial apps defined in the schema start.

To run your apps with SimSpace Weaver Local

1. In a **file chooser window**, go to *project-folder*\buildlocal\out\RelWithDebInfo.
2. SimSpace Weaver Local apps require a schema file named schema.yaml in the working directory of the apps. Any failure to read the required information from the schema terminates the apps.

   The schema for SimSpace Weaver Local doesn't have to be identical to *project-folder*\tools*project-name*-schema.yaml, but you can use it as a starting point.

   Choose **one** of the following:

   - Copy that schema to *project-folder*\buildlocal\out\RelWithDebInfo\schema.yaml.
   - Set the environment variable WEAVERLOCAL_SCHEMA_PATH to the name of a schema file with a different path or file name.

     Example

     ```
     set WEAVERLOCAL_SCHEMA_PATH=c:\projects\MyProject\tools\MyProject-schema.yaml
     ```

> **Note**
>
> If you set your environment variable from the command line, that environment variable (with that value) is only accessible from that command prompt session (console window).

3. The schema for the sample application defines a 2x2 grid, which creates 4 partitions. You will run a script that will launch 4 instances of the spatial app, to match the number of spatial apps specified in the schema. The script will also launch 1 view app. After all spatial apps launch and are assigned a partition, the simulation will automatically start ticking.

   To launch your apps

   a. At a **command prompt**, go to the local tools folder for your project.

   ```
   cd project-folder\tools\local
   ```

   b. Run the script to launch the apps.

   ```
   launch_simulation_locally.bat
   ```

   > **Note**
   >
   > If you set `WEAVERLOCAL_SCHEMA_PATH` to the name of a schema file, you must start your spatial apps at the command line in the same session (window) that you set the environment variable.

   > **Important**
   >
   > If you get a Windows security pop-up, choose **Allow Access** so that you can connect to the view app to visualize the simulation.

> **Note**
>
> You can also start your spatial and view apps manually. To do so, you must manually launch 4 instances of the spatial app and one view app.
>
> - Spatial app: `start PathfindingSampleLocalSpatial.exe`
> - View app: `start PathfindingSampleLocalView.exe`

View your local simulation

To view your local simulation, you can use any of the clients that are included with the SimSpaceWeaverAppSdkDistributable. For more information on building and using the sample clients, see Step 5: View your simulation of the quick start tutorial.

You must update the IP address and port number in the client to connect to the view app for your local simulation. Always use the following values with SimSpace Weaver Local:

```
tcp://127.0.0.1:7000
```

Depending on the client you select, you can update the IP address and port number as follows:

- **Unreal** – Change the URL on line 1 of `view_app_url.txt`
- **Console** – Launch the client with the IP address and port number URL as a parameter

Stop your local simulation

Your local simulation will continue to run if your local spatial apps are active. If you close one of the spatial app windows then you will stop the entire simulation. Close all of the other windows to clean up the rest of the simulation.

You can manually close each app window, or you can use the following script to automatically close all of them:

- *project-folder*`\tools\local\terminate_local_simulation.bat`

> **Note**
>
> Even though closing one spatial app window stops the simulation, make sure to close the other app windows. You won't be able to launch another local simulation successfully if there are windows still open from a previous simulation.

Debugging local simulations

You can debug your SimSpace Weaver Local apps with Microsoft Visual Studio. For more information about how to debug with Visual Studio, see the Microsoft Visual Studio documentation.

To debug your local simulation

1. Make sure that your `schema.yaml` is in your working directory.
2. In **Visual Studio**, open the context menu for each app that you want to debug (such as `PathfindingSampleLocalSpatial` or `PathfindingSampleLocalView`) and set the working directory in the debugging section.
3. Open the context menu for the app you want to debug and select **Set as Startup project**.
4. Choose **F5** to start debugging the app.

The requirements to debug a simulation are the same as the requirements to run a simulation normally. You must start the number of spatial apps specified in the schema. For example, if your schema specifies a 2x2 grid and you start a spatial app in debug mode, the simulation will not run until you start 3 more spatial apps (in debug mode or not in debug mode).

To debug a custom app, you must first start your spatial apps and then start the custom app in the debugger.

Note that your simulation runs in lock step. As soon as an app reaches a breakpoint, all other apps will pause. After you continue from that breakpoint, the other apps will continue.

Local development differences in version 1.15.3

This section describes changes in development with SimSpace Weaver Local starting in version 1.15.3. These changes impact workflows for SimSpace Weaver Local projects in C++, Python, Unity, and Unreal Engine.

Topics

Changes to files

- `SimSpaceWeaverAppSdkLocal` library files are now named `weaver_app_sdk_cxx_v1_full_local`.
  - You can find these files in *`sdk-folder`*`\SimSpaceWeaverAppSdk-1.15.3\lib\weaverlocal\windows`.
- Your projects should no longer link to or include `SimSpaceWeaverAppSDK-1.15.1\include\aws\weaverruntime\local_ffi`.
- The Python scripts for SimSpace Weaver Local no longer require `cmake`.
- There are new or renamed Python scripts for SimSpace Weaver Local:
  - `build-local` — Builds the Python app for launch. It puts all of your Python code and the SimSpace Weaver Local library files in the `buildlocal` directory.
  - `local-config` — Defines the environment variables for the local scripts.
  - `start-python-locally` — Sets up the app's `PythonPath` and starts Python.
  - The following scripts are equivalent to the cloud scripts with similar names:
    - `quick-start-local`
    - `start-simulation-local`
    - `stop-simulation-local`

Update an existing C++ project to SimSpace Weaver Local 1.15.3

Requirements

- Version 1.15.3 SimSpace Weaver app SDK
- An existing C++ project for SimSpace Weaver Local version 1.15.2 or earlier

To update your project

1. Make sure that your project includes the required header:

```
#include <aws/weaverruntime/ffi/weaver_app_sdk_cxx_ffi_v1/src/lib.rs.h>
```

> **Note**
>
> This is usually included in `aws\weaverruntime\detail.h`

2. Make sure that your project **does not** include the following deprecated header:

```
#include <aws/weaverruntime/local_ffi/Bridge.h>
```

> **Note**
>
> This is usually included in `aws\weaverruntime\detail.h`

3. In your CMake file or build script, replace `SimSpaceWeaverAppSdkLocal` static library names with the new `weaver_app_sdk_cxx_v1_full_local` name.
4. If you haven't run *`sdk-folder`*`\docker-create-image.bat` from version 1.15.3, run it now. You only need to do this 1 time.
5. Follow the usual procedures to build and run your project.

Troubleshooting

You get linker errors (unresolved external symbol)

Your compiler outputs linker errors such as the following (line breaks added for readability).

```
Error    LNK2019    unresolved external symbol
  "class outcome_v2_92ee5284::basic_result<class
  Aws::WeaverRuntime::Application,
  enum Aws::WeaverRuntime::ffi::weaver_app_sdk_cxx_ffi_v1::ErrorCode
```

To fix the problem

- Make sure your project does **not** include <aws/weaverruntime/local_ffi/Bridge.h>.

You get a type incompatibility error

You might get an error such as the following (line breaks added for readability).

```
the object has type qualifiers that are not compatible with the member function
  object type is: const rust::cxxbridge1::String
```

To fix the problem

1. Update your SimSpace Weaver header files with the 1.15.3 versions.
2. Make sure any use of `rust::cxxbridge1::String` is similar to the following:

   ```
   rust::cxxbridge1::String domain_name = domain.name.value;
   if (domain.type_ == Api::DomainType::Spatial && Name.Compare(domain_name.c_str())
    == 0)
   ```

Run a new Python project in SimSpace Weaver Local 1.15.3

Requirements

- Version 1.15.3 SimSpace Weaver app SDK
- A new SimSpace Weaver Local version 1.15.3 `PythonBubblesSample` project

To run your project

1. In a command prompt window, go to *`project-folder`*`\tools\local\windows`.
2. Run `quick-start-local.bat`

   4 spatial apps, 1 view app, and 1 client should start locally. The client should display bubbles.

Troubleshooting

You get an error that says GLIBCXX isn't found

The following error indicates a failure to import the SimSpace Weaver Python app SDK. A likely cause is that your C++ libraries are out of date.

```
ImportError: /lib64/libstdc++.so.6: version `GLIBCXX_3.4.29' not found
```

To fix the problem

1. Update the source of your `libstdc++` libraries (such as `gcc` or `msvc`) to a version that includes the specified version of `GLIBCXX` (in the example provided, the version is 3.4.29).
2. Make sure that the system environment variable `LD_LIBRARY_PATH` is set to the correct path to `lib64`.

The quick start script fails

The quick start script fails with a message such as the following.

```
python: can't open file 'C:\usr\project\buildlocal\bIn\bubbles_tkinter_client.py':
  [Errno 2] No such file or directory
```

To fix the problem

1. Go to *`project-folder`*`\tools\local\windows` and edit `local-config.bat`.
2. Make sure that the following environment variables are set to the correct paths for your local system:
   - `TOOLS_DIR` is set to your *`project-folder`*`\tools`
   - `TOOLS_DIR_WINDOWS` is set to your *`project-folder`*`\tools\local\windows`
   - `PROJECT_ROOT` is set to your *`project-folder`*
   - `BUILD_DIR` is set to your *`project-folder`*`\buildlocal`
   - `APP_SDK_DIR` is set to your *`sdk-folder`*

Update an existing Unity project to SimSpace Weaver Local 1.15.3

Requirements

- Version 1.15.3 SimSpace Weaver app SDK
- An existing Unity project for SimSpace Weaver Local version 1.15.2 or earlier

To update your project

1. In Unity, remove the existing **AWS SimSpace Weaver** package.
   a. Open your existing Unity project.
   b. In the editor window, choose **Window** > **Package Manager**.
   c. Under **Packages - Unity Technologies**, select **AWS SimSpace Weaver** and choose **Remove**.
2. In your version 1.15.3 *sdk-folder*, run `download-unity-package.bat`
3. Follow the instructions in `Unity_SDK_for_AWS_SimSpace_Weaver.pdf` to add the newly downloaded `SimSpaceWeaverUnityPackage.zip` as the **AWS SimSpace Weaver** package in the Unity editor.

Update an existing Unreal Engine project to SimSpace Weaver Local 1.15.3

Requirements

- Version 1.15.3 SimSpace Weaver app SDK
- An existing Unreal Engine project for SimSpace Weaver Local version 1.15.2 or earlier

To update your project

1. Close all Unreal project and code editor windows.
2. In a command prompt window, go to your version 1.15.3 *sdk-folder*.
3. Run `update-unreal-project.bat --path` *project-folder* `--name` *project-name*.

> **Note**
>
> This replaces your existing plugin with the new plugin. Any modifications are erased.

4. Follow the instructions in `AWS_SimSpace_Weaver_Unreal_Guide.pdf` to build your project for SimSpace Weaver Local.

Troubleshooting

Failure to delete exsiting plugin files

You might get errors similar to the following.

```
cannot remove '/usr/src/project/{PROJECT_NAME}/src/PathfindingSampleUnrealSpatial/
PathfindingUnrealProject/Plugins/SimSpaceWeaverAppSdkPlugin/Binaries/Win64/
UnrealEditor-WeaverAppSdk.dll': Operation not permitted
cannot remove '/usr/src/project/{PROJECT_NAME}/src/PathfindingSampleUnrealSpatial/
PathfindingUnrealProject/Plugins/SimSpaceWeaverAppSdkPlugin/Binaries/Win64/
UnrealEditor-WeaverAppSdkLocal.dll': Operation not permitted
cannot remove '/usr/src/project/{PROJECT_NAME}/src/PathfindingSampleUnrealSpatial/
PathfindingUnrealProject/Plugins/SimSpaceWeaverAppSdkPlugin/Binaries/Win64/
UnrealEditor-WeaverCppMetrics.dll': Operation not permitted
```

To fix the problem

1. Make sure that the Unreal project editor window and code editors are closed.
2. Run `update-unreal-project.bat`.

Frequently asked questions about SimSpace Weaver Local 1.15.3

- **Q1: How do I change my SimSpace Weaver Local environment variables for my Python project?**
  - Edit *`project-folder`*`\tools\windows\local-config` for your Python project.

AWS SimSpace Weaver app SDK

The SimSpace Weaver app SDK provides APIs that you can use to control the entities in your simulation and respond to SimSpace Weaver events. It includes the following namespace:

- **API** – core definitions of the API and its use

Link with the following library:

- `libweaver_app_sdk_cxx_v1_full.so`

> **Important**
>
> The library is available for dynamic linking when you run your apps in the AWS Cloud. You don't need to upload it with your apps.

> **Note**
>
> The SimSpace Weaver app SDK APIs control data within your simulation. These APIs are separate from the SimSpace Weaver service APIs, which control your SimSpace Weaver service resources (such as simulations, apps, and clocks) in AWS. For more information, see SimSpace Weaver API references.

Topics

API methods return a Result

The majority of SimSpace Weaver API functions have a return type `Aws::WeaverRuntime::Result<T>`. If the function has executed successfully, the `Result` contains T. Otherwise, the `Result` contains an `Aws::WeaverRuntime::ErrorCode` that represents an error code from the Rust App SDK.

Example

```
Result<Transaction> BeginUpdate(Application& app)
```

This method:

- Returns `Transaction` if `BeginUpdate()` executes successfully.
- Returns `Aws::WeaverRuntime::ErrorCode` if `BeginUpdate()` fails.

Interacting with the app SDK at the top level

Lifecycle

- The SimSpace Weaver app SDK manages app life cycle. You don't need to read or write the lifecycle state of an app.

Partitions

- Use `Result <PartitionSet> AssignedPartitions(Transaction& txn);` to get owned partitions.
- Use `Result <PartitionSet> AllPartitions(Transaction& txn);` to get all partitions in the simulation.

Simulation management

This section describes solutions for common simulation management tasks.

Topics

Start a simulation

Use `CreateApplication()` to create an app.

Example

```
Result<Application> applicationResult = Api::CreateApplication();

if (!applicationResult)
{
    ErrorCode errorCode = WEAVERRUNTIME_EXPECT_ERROR(applicationResult);

    std::cout << "Failed to create application. Error code " <<
        static_cast<std::underlying_type_t<ErrorCode>>(errorCode) <<
        " Last error message "<< Api::LastErrorMessage() << ".";

    return 1;
}

/**
* Run simulation
*/
RunSimulation(std::move(applicationResult.assume_value()));
```

Update a simulation

Use the following `BeginUpdate` functions to update the app:

- `Result<Transaction> BeginUpdate(Application& app)`
- `Result<bool> BeginUpdateWillBlock(Application& app)` – tells you if `BeginUpdate()` will block or not block.

Use `Result<void> Commit(Transaction& txn)` to commit the changes.

Example

```
Result<void> AppDriver::RunSimulation(Api::Application app) noexcept
{
    while (true)
    {
        {
            bool willBlock;

            do
            {
```

```
                WEAVERRUNTIME_TRY(willBlock, Api::BeginUpdateWillBlock(m_app));
            } while (willBlock);
        }

        WEAVERRUNTIME_TRY(Transaction transaction, Api::BeginUpdate(app));

        /**
         * Simulate app.
         */
        WEAVERRUNTIME_TRY(Simulate(transaction));
        WEAVERRUNTIME_TRY(Api::Commit(std::move(transaction)));
    }

    return Success();
}
```

Terminate a simulation

Use `Result<void> DestroyApplication(Application&& app)` to terminate the app and the simulation.

Other apps find out that the simulation is shutting down when they receive `ErrorCode::ShuttingDown` from their calls to `BeginUpdateWillBlock()` or `BeginUpdate()`. When an app receives `ErrorCode::ShuttingDown`, it can call `Result<void> DestroyApplication(Application&& app)` to terminate itself.

Example

```
Result<void> AppDriver::EncounteredAppError(Application&& application) noexcept
{
    const ErrorCode errorCode = WEAVERRUNTIME_EXPECT_ERROR(runAppResult);

    switch (errorCode)
    {
    case ErrorCode::ShuttingDown:
        {
            // insert custom shutdown process here.

            WEAVERRUNTIME_TRY(Api::DestroyApplication(std::move(application)));
            return Success();
        }
    default:
        {
```

```
            OnAppError(errorCode);
            return errorCode;
        }
    }
}
```

Important

Only call `Result<void> DestroyApplication(Application&& app)` after `Api::Commit()`. Destroying an application during an update can cause undefined behavior.

Important

You must call `DestroyApplication()` before the program exits to make sure that the application reports as terminating successfully.
Failure to call `DestroyApplication()` when the program exits will cause the status to be considered as `FATAL`.

Subscriptions

You create a subscription with a subscription area and a domain ID. The domain ID represents the domain that owns that subscription area. A `BoundingBox2F32` describes the subscription area. Use the following function to create a subscription:

```
Result<SubscriptionHandle> CreateSubscriptionBoundingBox2F32(Transaction& txn, DomainId
 id, const BoundingBox2F32& boundingBox)
```

Example

```
Result<void> CreateSubscriptionInSpatialDomain(Transaction& transaction)
{
    WEAVERRUNTIME_TRY(Api::PartitionSet partitionSet, Api::AllPartitions(transaction));


    Api::DomainId spatialDomainId;
```

```
    for (const Api::Partition& partition : partitionSet.partitions)
    {
        if (partition.domain_type == Api::DomainType::Spatial)
        {
            /**
            * Get the spatial domain ID.
            */
            spatialDomainId = partition.domain_id;
            break;
        }
    }

    constexpr Api::BoundingBox2F32 subscriptionBounds {
        /* min */ { /* x */ 0, /* y */ 0 },
        /* max */ { /* x */ 1000, /* y */ 1000 } }

    WEAVERRUNTIME_TRY(
        Api::SubscriptionHandle subscriptionHandle,
        Api::CreateSubscriptionBoundingBox2F32(
        transaction,
        spatialDomainId,
        subscriptionBounds));

    return Success();
}
```

You can use the `Api::SubscriptionHandle` returned by `CreateSubscriptionBoundingBox2F32()` to modify the subscription. You pass it as an argument to the following functions:

```
Result<void> ModifySubscriptionBoundingBox2F32(Transaction& txn, SubscriptionHandle
 handle, const BoundingBox2F32& boundingBox)
```

```
Result<void> DeleteSubscription(Transaction& txn, SubscriptionHandle handle)
```

Entities

You call the `Store` and `Load` APIs using the `Api:Entity` of the `Result<Api::Entity>` returned from `CreateEntity()`, or from an ownership change event when an entity enters the app's subscription area (for more information, see Entity events). We recommend that you track your `Api::Entity` objects so that you can use them with these APIs.

Topics

Create entities

Use `CreateEntity()` to create an entity. You define the meaning of the `Api::TypeId` that you pass to this function.

```
Namespace
{
    constexpr Api::TypeId k_entityTypeId { /* value */ 512 };
}

Result<void> CreateEntity(Transaction& transaction)
{
    WEAVERRUNTIME_TRY(
        Api::Entity entity,
        Api::CreateEntity(
            transaction, Api::BuiltinTypeIdToTypeId(k_entityTypeId )));
}
```

> **Note**
>
> The values 0-511 for `Api::BuiltinTypeId` are reserved. Your entity TypeID (`k_entityTypeId` in this example) must have a value of 512 or higher.

Transfer an entity to a spatial domain

After a custom app or service app creates an entity, the app must transfer the entity to a spatial domain in order for the entity to exist spatially in the simulation. Entities in a spatial domain can be read by other apps and updated by a spatial app. Use the `ModifyEntityDomain()` API to transfer an entity to a spatial domain.

```
AWS_WEAVERRUNTIME_API Result<void> ModifyEntityDomain(Transaction& txn, const Entity&
 entity, DomainId domainId) noexcept;
```

If the DomainId doesn't match the assigned Partition of the calling app, then the DomainId must be for a DomainType::Spatial Domain. The ownership transfer to the new Domain occurs during the Commit(Transaction&&).

Parameters

txn

The current Transaction.

entity

The target Entity for the change of Domain.

domainId

The DomainId of the destination Domain for the Entity.

This API returns Success if the entity domain was successfully changed.

Write and read entity field data

All entity data fields are blob types. You can write up to 1,024 bytes of data to an entity. We recommend that you keep blobs as small as possible because larger sizes will reduce performance. When you write to a blob, you pass SimSpace Weaver a pointer to the data and its length. When you read from a blob, SimSpace Weaver provides you with a pointer and a length to read. All reads must be complete before the app calls Commit(). Pointers returned from a read call are invalidated when the app calls Commit().

Important

- Reading from a cached blob pointer after a Commit() is unsupported and can cause the simulation to fail.
- Writing to a blob pointer returned from a read call is unsupported and can cause the simulation to fail.

Topics

Store the field data of an entity

The following examples demonstrate how you can store (write to the state fabric) the field data of an entity that the app owns. These examples use the following function:

```
AWS_WEAVERRUNTIME_API Result<void> StoreEntityField(
    Transaction& txn,
    const Entity& entity,
    TypeId keyTypeId,
    FieldIndex index,
    std::int8_t* src,
    std::size_t length) noexcept;
```

The `Api::TypeId keyTypeId` parameter represents the data type of the passed in data.

The `Api::TypeId keyTypeId` parameter should receive the corresponding `Api::TypeId` from `Api::BuiltinTypeId`. If there is no appropriate conversion, you can use `Api::BuiltinTypeId::Dynamic`.

For complex data types, use `Api::BuiltInTypeId::Dynamic`.

> **Note**
>
> The value of `FieldIndex index` must be greater than 0. The value 0 is reserved for the index key (see `StoreEntityIndexKey()`).

Example using primitive data types

```
namespace
{
    constexpr Api::FieldIndex k_isTrueFieldId { /* value */ 1 };
}
```

```
Result<void> SetEntityFields(
    Api::Entity& entity,
    Transaction& transaction)
{
    bool value = true;

    auto* src = reinterpret_cast<std::int8_t*>(value);
    size_t length = sizeof(*value);

    WEAVERRUNTIME_TRY(Api::StoreEntityField(
        transaction,
        entity,
        Api::BuiltinTypeIdToTypeId(
            Aws::WeaverRuntime::Api::BuiltinTypeId::Bool),
        k_isTrueFieldId,
        src,
        length));
}
```

Example using a struct to hold the data

```
namespace
{
    constexpr Api::FieldIndex k_dataFieldId { /* value */ 1 };
}

struct Data
{
    bool boolData;
    float floatData;
};

Result<void> SetEntityFields(
    Api::Entity& entity,
    Transaction& transaction)
{
    Data data = { /* boolData */ false, /* floatData */ -25.93 };

    auto* src = reinterpret_cast<std::int8_t*>(data);
    size_t length = sizeof(*data);

    WEAVERRUNTIME_TRY(Api::StoreEntityField(
        transaction,
```

```
        entity,
        Api::BuiltinTypeIdToTypeId(
            Aws::WeaverRuntime::Api::BuiltinTypeId::Dynamic),
        k_dataFieldId,
        src,
        length));
}
```

Load the field data of an entity

The following examples demonstrate how you can load (read from the state fabric) the field data of an entity. These examples use the following function:

```
Result<std::size_t> LoadEntityField(
    Transaction& txn,
    const Entity& entity,
    TypeId keyTypeId,
    FieldIndex index,
    std::int8_t** dest) noexcept;
```

The `Api::TypeId keyTypeId` parameter should receive the corresponding `Api::TypeId` from `Api::BuiltinTypeId`. If there is no appropriate conversion, you can use `Api::BuiltinTypeId::Dynamic`.

> **Note**
>
> The value of `FieldIndex index` must be greater than 0. The value 0 is reserved for the index key (see `StoreEntityIndexKey()`).

Example using primitive data types

```
namespace
{
    constexpr Api::FieldIndex k_isTrueFieldId { /* value */ 1 };
}

Result<void> LoadEntityFields(
    Api::Entity& entity,
    Transaction& transaction)
```

```
{
    std::int8_t* dest = nullptr;

    WEAVERRUNTIME_TRY(Api::LoadEntityField(
        transaction,
        entity,
        Api::BuiltinTypeIdToTypeId(
            Aws::WeaverRuntime::Api::BuiltinTypeId::Bool),
        k_isTrueFieldId,
        &dest));

    bool isTrueValue = *reinterpret_cast<bool*>(dest);
}
```

Example using a struct to hold the data

```
namespace
{
    constexpr Api::FieldIndex k_dataFieldId { /* value */ 1 };
}

struct Data
{
    bool boolData;
    float floatData;
};

Result<void> LoadEntityFields(
    Api::Entity& entity,
    Transaction& transaction)
{
    std::int8_t* dest = nullptr;

    WEAVERRUNTIME_TRY(Api::LoadEntityField(
        transaction,
        entity,
        Api::BuiltinTypeIdToTypeId(
            Aws::WeaverRuntime::Api::BuiltinTypeId::Dynamic),
        k_dataFieldId,
        &dest));

    Data dataValue = *reinterpret_cast<Data*>(dest);
}
```

Loading the field data of removed entities

You can't load (read from the state fabric) entity field data for entities that have been removed from the app's ownership and subscription areas. The following example results in an error because it calls `Api::LoadIndexKey()` on an entity as a result of an `Api::ChangeListAction::Remove`. The second example shows a correct way to store and load entity data directly in the app.

Example of incorrect code

```
Result<void> ProcessSubscriptionChanges(Transaction& transaction)
{
    /* ... */

    WEAVERRUNTIME_TRY(Api::SubscriptionChangeList subscriptionChangeList,
        Api::AllSubscriptionEvents(transaction));

    for (const Api::SubscriptionEvent& event :
        subscriptionChangeList.changes)
    {
        switch (event.action)
        {
        case Api::ChangeListAction::Remove:
            {
                std::int8_t* dest = nullptr;

                /**
                 * Error!
                 * This calls LoadEntityIndexKey on an entity that
                 * has been removed from the subscription area.
                 */
                WEAVERRUNTIME_TRY(Api::LoadEntityIndexKey(
                    transaction,
                    event.entity,
                    Api::BuiltinTypeIdToTypeId(
                        Api::BuiltinTypeId::Vector3F32),
                    &dest));

                AZ::Vector3 position =
                    *reinterpret_cast<AZ::Vector3*>(dest);
                break;
            }
        }
```

```
    }

    /* ... */
}
```

Example of a correct way to store and load entity data in the app

```
Result<void> ReadAndSaveSubscribedEntityPositions(Transaction& transaction)
{
    static std::unordered_map<Api::EntityId, AZ::Vector3>
        positionsBySubscribedEntity;

    WEAVERRUNTIME_TRY(Api::SubscriptionChangeList subscriptionChangeList,
        Api::AllSubscriptionEvents(transaction));

    for (const Api::SubscriptionEvent& event :
        subscriptionChangeList.changes)
    {
        switch (event.action)
        {
        case Api::ChangeListAction::Add:
            {
                std::int8_t* dest = nullptr;

                /**
                 * Add the position when the entity is added.
                 */
                WEAVERRUNTIME_TRY(Api::LoadEntityIndexKey(
                    transaction,
                    event.entity,
                    Api::BuiltinTypeIdToTypeId(
                        Api::BuiltinTypeId::Vector3F32),
                    &dest));

                AZ::Vector3 position =
                    *reinterpret_cast<AZ::Vector3*>(dest);
                positionsBySubscribedEntity.emplace(
                    event.entity.descriptor->id, position);

                break;
            }
        case Api::ChangeListAction::Update:
```

```
            {
                std::int8_t* dest = nullptr;

                /**
                 * Update the position when the entity is updated.
                 */
                WEAVERRUNTIME_TRY(Api::LoadEntityIndexKey(
                    transaction,
                    event.entity,
                    Api::BuiltinTypeIdToTypeId(
                        Api::BuiltinTypeId::Vector3F32),
                    &dest));

                AZ::Vector3 position =
                    *reinterpret_cast<AZ::Vector3*>(dest);
                positionsBySubscribedEntity[event.entity.descriptor->id] =
                    position;

                break;
            }
        case Api::ChangeListAction::Remove:
            {
                /**
                 * Load the position when the entity is removed.
                 */
                AZ::Vector3 position = positionsBySubscribedEntity[
                    event.entity.descriptor->id];

                /**
                 * Do something with position...
                 */
                break;
            }
        }
    }

    /* ... */
}
```

Store the position of an entity

You can store (write to the state fabric) the position of an entity using an integer data structure. These examples use the following function:

```
Result<void> StoreEntityIndexKey(
    Transaction& txn,
    const Entity& entity,
    TypeId keyTypeId,
    std::int8_t* src,
    std::size_t length)
```

Note

You must provide `Api::BuiltinTypeId::Vector3F32` to `Api::StoreEntityIndexKey()`, as shown in the following examples.

Example using an array to represent the position

```
Result<void> SetEntityPositionByFloatArray(
    Api::Entity& entity,
    Transaction& transaction)
{
    std::array<float, 3> position = { /* x */ 25, /* y */ 21, /* z */ 0 };

    auto* src = reinterpret_cast<std::int8_t*>(position.data());
    std::size_t length = sizeof(position);

    WEAVERRUNTIME_TRY(Api::StoreEntityIndexKey(
        transaction,
        entity,
        Api::BuiltinTypeIdToTypeId(Api::BuiltinTypeId::Vector3F32),
        src,
        length));
}
```

Example using a struct to represent the position

```
struct Position
{
   float x;
   float y;
   float z;
};
```

```
Result<void> SetEntityPositionByStruct(
    Api::Entity& entity,
    Transaction& transaction)
{
    Position position = { /* x */ 25, /* y */ 21, /* z */ 0 };

    auto* src = reinterpret_cast<std::int8_t*>(&position);
    std::size_t length = sizeof(position);

    WEAVERRUNTIME_TRY(Api::StoreEntityIndexKey(
        transaction,
        entity,
        Api::BuiltinTypeIdToTypeId(Api::BuiltinTypeId::Vector3F32),
        src,
        length));
}
```

Load the position of an entity

You can load (read from the state fabric) the position of an entity using an integer data structure. These examples use the following function:

> **Note**
>
> You must provide `Api::BuiltinTypeId::Vector3F32` to `Api::LoadEntityIndexKey()`, as shown in the following examples.

Example using an array to represent the position

```
Result<void> GetEntityPosition(Api::Entity& entity,
    Transaction& transaction)
{
    std::int8_t* dest = nullptr;

    WEAVERRUNTIME_TRY(Aws::WeaverRuntime::Api::LoadEntityIndexKey(
        transaction,
        entity,
        Api::BuiltinTypeIdToTypeId(
            Aws::WeaverRuntime::Api::BuiltinTypeId::Vector3F32),
        &dest));
```

```
    std::array<float, 3> position =
        *reinterpret_cast<std::array<float, 3>*>(dest);
}
```

Example using a struct to represent the position

```
struct Position
{struct
   float x;
   float y;
   float z;
};

Result<void> GetEntityPosition(Api::Entity& entity, Transaction& transaction)
{
    std::int8_t* dest = nullptr;

    WEAVERRUNTIME_TRY(Aws::WeaverRuntime::Api::LoadEntityIndexKey(
        transaction,
        entity,
        Api::BuiltinTypeIdToTypeId(
            Aws::WeaverRuntime::Api::BuiltinTypeId::Vector3F32),
        &dest));

    Position position = *reinterpret_cast<Position*>(dest);
}
```

Entity events

You can use the following functions in the SimSpace Weaver app SDK to get all ownership and subscription events:

- `Result<OwnershipChangeList> OwnershipChanges(Transaction& txn)`
- `Result<SubscriptionChangeList> AllSubscriptionEvents(Transaction& txn)`

You can use the SimSpace Weaver demo framework if you need callback-driven entity event processing. For more information, see the following header file:

- *sdk-folder*`/packaging-tools/samples/ext/DemoFramework/include/DemoFramework/EntityEventProcessor.h`

You can also create your own entity event processing.

Topics

Iterate through events for owned entities

Use `OwnershipChanges()` to get a list of events for owned entities (entities in the app's ownership area). The function has the following signature:

```
Result<OwnershipChangeList> OwnershipChanges(Transaction& txn)
```

Then iterate through the entities with a loop, as demonstrated in the following example.

Example

```
WEAVERRUNTIME_TRY(Result<Api::OwnershipChangeList> ownershipChangesResult,
 Api::OwnershipChanges(transaction));

for (const Api::OwnershipChange& event : ownershipChangeList.changes)
{
    Api::Entity entity = event.entity;
    Api::ChangeListAction action = event.action;

    switch (action)
    {
    case Api::ChangeListAction::None:
        // insert code to handle the event
        break;
    case Api::ChangeListAction::Remove:
        // insert code to handle the event
        break;
    case Api::ChangeListAction::Add:
        // insert code to handle the event
        break;
    case Api::ChangeListAction::Update:
        // insert code to handle the event
        break;
```

```
    case Api::ChangeListAction::Reject:
        // insert code to handle the event
        break;
    }
}
```

Event types

- None – The entity is in the area and its position and field data weren't modified.
- `Remove` – The entity was removed from the area.
- `Add` – The entity was added to the area.
- `Update` – The entity is in the area and was modified.
- `Reject` – The app failed to remove the entity from the area.

> **Note**
>
> In the case of a `Reject` event, the app will attempt the transfer again on the next tick.

Iterate through events for subscribed entities

Use `AllSubscriptionEvents()` to get a list of events for subscribed entities (entities in the app's subscription area). The function has the following signature:

```
Result<SubscriptionChangeList> AllSubscriptionEvents(Transaction& txn)
```

Then iterate through the entities with a loop, as demonstrated in the following example.

Example

```
WEAVERRUNTIME_TRY(Api::SubscriptionChangeList subscriptionChangeList,
 Api::AllSubscriptionEvents(transaction));

for (const Api::SubscriptionEvent& event : subscriptionChangeList.changes)
{
    Api::Entity entity = event.entity;
    Api::ChangeListAction action = event.action;
```

```
    switch (action)
    {
    case Api::ChangeListAction::None:
        // insert code to handle the event
        break;
    case Api::ChangeListAction::Remove:
        // insert code to handle the event
        break;
    case Api::ChangeListAction::Add:
        // insert code to handle the event
        break;
    case Api::ChangeListAction::Update:
        // insert code to handle the event
        break;
    case Api::ChangeListAction::Reject:
        // insert code to handle the event
        break;
    }
}
```

Event types

- None – The entity is in the area and its position and field data weren't modified.
- `Remove` – The entity was removed from the area.
- `Add` – The entity was added to the area.
- `Update` – The entity is in the area and was modified.
- `Reject` – The app failed to remove the entity from the area.

> **Note**
>
> In the case of a `Reject` event, the app will attempt the transfer again on the next tick.

Iterate through ownership change events for entities

To get events where an entity moves between an ownership area and subscription area, compare the changes between the current and previous entity ownership and subscription events.

You can handle these events by reading:

- `Api::SubscriptionChangeList`
- `Api::OwnershipEvents`

You can then compare the changes to previously stored data.

The following example shows how you can handle entity ownership change events. This example assumes that for entities transitioning between being subscribed entities and owned entities (in either direction), the ownership remove/add event occurs first followed by the subscription remove/add event in the next tick.

Example

```
Result<void> ProcessOwnershipEvents(Transaction& transaction)
{
    using EntityIdsByAction =
        std::unordered_map<Api::ChangeListAction,
        std::vector<Api::EntityId>>;
    using EntityIdSetByAction =
        std::unordered_map<Api::ChangeListAction,
        std::unordered_set<Api::EntityId>>;

    static EntityIdsByAction m_entityIdsByPreviousOwnershipAction;

    EntityIdSetByAction entityIdSetByAction;

    /**
     * Enumerate Api::SubscriptionChangeList items
     * and store Add and Remove events.
     */
    WEAVERRUNTIME_TRY(Api::SubscriptionChangeList subscriptionEvents,
        Api::AllSubscriptionEvents(transaction));

    for (const Api::SubscriptionEvent& event : subscriptionEvents.changes)
    {
        const Api::ChangeListAction action = event.action;

        switch (action)
        {
        case Api::ChangeListAction::Add:
        case Api::ChangeListAction::Remove:

            {
```

```
                entityIdSetByAction[action].insert(
                    event.entity.descriptor->id);
                break;
            }
        case Api::ChangeListAction::None:
        case Api::ChangeListAction::Update:
        case Api::ChangeListAction::Reject:
            {
                break;
            }
        }
    }

    EntityIdsByAction entityIdsByAction;

    /**
     * Enumerate Api::OwnershipChangeList items
     * and store Add and Remove events.
     */

    WEAVERRUNTIME_TRY(Api::OwnershipChangeList ownershipChangeList,
        Api::OwnershipChanges(transaction));

    for (const Api::OwnershipChange& event : ownershipChangeList.changes)
    {
        const Api::ChangeListAction action = event.action;

        switch (action)
        {
        case Api::ChangeListAction::Add:
        case Api::ChangeListAction::Remove:
            {
                entityIdsByAction[action].push_back(
                    event.entity.descriptor->id);
                break;
            }
        case Api::ChangeListAction::None:
        case Api::ChangeListAction::Update:
        case Api::ChangeListAction::Reject:
            {
                break;
            }
        }
```

```
    }

    std::vector<Api::EntityId> fromSubscribedToOwnedEntities;
    std::vector<Api::EntityId> fromOwnedToSubscribedEntities;

    /**
     * Enumerate the *previous* Api::OwnershipChangeList Remove items
     * and check if they are now in
     * the *current* Api::SubscriptionChangeList Add items.
     *
     * If true, then that means
     * OnEntityOwnershipChanged(bool isOwned = false)
     */
    for (const Api::EntityId& id : m_entityIdsByPreviousOwnershipAction[
        Api::ChangeListAction::Remove])
    {
        if (entityIdSetBySubscriptionAction[
            Api::ChangeListAction::Add].find(id) !=
                entityIdSetBySubscriptionAction[
                Api::ChangeListAction::Add].end())
        {
            fromOwnedToSubscribedEntities.push_back(id);
        }
    }


    /**
     * Enumerate the *previous* Api::OwnershipChangeList Add items
     * and check if they are now in
     * the *current* Api::SubscriptionChangeList Remove items.
     *
     * If true, then that means
     * OnEntityOwnershipChanged(bool isOwned = true)
     */
    for (const Api::EntityId& id : m_entityIdsByPreviousOwnershipAction[
        Api::ChangeListAction::Add])
    {
        if (entityIdSetBySubscriptionAction[
            Api::ChangeListAction::Remove].find(id) !=

                entityIdSetBySubscriptionAction[
                Api::ChangeListAction::Remove].end())
        {
            fromSubscribedToOwnedEntities.push_back(id);
```

```
        }
    }

    m_entityIdsByPreviousOwnershipAction = entityIdsByOwnershipAction;

    return Success();
}
```

Result and error handling

The `Aws::WeaverRuntime::Result<T>` class uses a third-party `Outcome` library. You can use the following pattern to check the `Result` and catch errors returned by API calls.

```
void DoBeginUpdate(Application& app)
{
    Result<Transaction> transactionResult = Api::BeginUpdate(app);

    if (transactionResult)
    {
        Transaction transaction =
            std::move(transactionResult).assume_value();

        /**
         * Do things with transaction ...
         */
    }
    else
    {
        ErrorCode errorCode = WEAVERRUNTIME_EXPECT_ERROR(transactionResult);
        /**
         * Macro compiles to:
         * ErrorCode errorCode = transactionResult.assume_error();
         */
    }
}
```

Result control statement macro

Inside a function with a return type `Aws::WeaverRuntime::Result<T>`, you can use the `WEAVERRUNTIME_TRY` macro instead of the previous code pattern. The macro will execute the function passed to it. If the passed function fails, the macro will make the enclosing function return an error. If the passed function succeeds, execution progresses to the next line. The following

example shows a rewrite of the previous `DoBeginUpdate()` function. This version uses the `WEAVERRUNTIME_TRY` macro instead of the if-else control structure. Note that the return type of the function is `Aws::WeaverRuntime::Result<void>`.

```
Aws::WeaverRuntime::Result<void> DoBeginUpdate(Application& app)
{
    /**
     * Execute Api::BeginUpdate()
     * and return from DoBeginUpdate() if BeginUpdate() fails.
     * The error is available as part of the Result.
     */
    WEAVERRUNTIME_TRY(Transaction transaction, Api::BeginUpdate(m_app));

    /**
     * Api::BeginUpdate executed successfully.
     *
     * Do things here.
     */

    return Aws::Success();
}
```

If `BeginUpdate()` fails, the macro makes `DoBeginUpdate()` return early with a failure. You can use the `WEAVERRUNTIME_EXPECT_ERROR` macro to get the `Aws::WeaverRuntime::ErrorCode` from `BeginUpdate()`. The following example shows how the `Update()` function calls `DoBeginUpdate()` and gets the error code on failure.

```
void Update(Application& app)
{
    Result<void> doBeginUpdateResult = DoBeginUpdate(app);

    if (doBeginUpdateResult)
    {
        /**
         * Successful.
         */
    }
    else
    {
        /**
         * Get the error from Api::BeginUpdate().
         */
```

```
        ErrorCode errorCode = WEAVERRUNTIME_EXPECT_ERROR(doBeginUpdateResult);

    }
}
```

You can make the error code from `BeginUpdate()` available to a function that calls `Update()` by changing the return type of `Update()` to `Aws::WeaverRuntime::Result<void>`. You can repeat this process to keep sending the error code further down the call stack.

Generics and domain types

The SimSpace Weaver app SDK provides the single-precision data types `Api::Vector2F32` and `Api::BoundingBox2F32`, and the double-precision `Api::Vector2F64` and `Api::BoundingBox2F64`. These data types are passive data structures with no convenience methods. Note that the API only uses `Api::Vector2F32` and `Api::BoundingBox2F32`. You can use these data types to create and modify subscriptions.

The SimSpace Weaver demo framework provides a minimal version of the AzCore math library, which contains `Vector3` and `Aabb` . For more information, see the header files in:

- *`sdk-folder`*`/packaging-tools/samples/ext/DemoFramework/include/AzCore/Math`

Miscellaneous app SDK operations

Topics

AllSubscriptionEvents and OwnershipChanges contain events from the last call

The return values of calls to `Api::AllSubscriptionEvents()` and `Api::OwnershipChanges()` contain events from the last call, **not the last tick**. In the following example, `secondSubscriptionEvents` and `secondOwnershipChangeList` are empty because their functions are called immediately after the first calls.

If you wait 10 ticks and then call `Api::AllSubscriptionEvents()` and `Api::OwnershipChanges()`, their results will both contain events and changes from the last 10 ticks (not the last tick).

Example

```
Result<void> ProcessOwnershipChanges(Transaction& transaction)
{
    WEAVERRUNTIME_TRY(
        Api::SubscriptionChangeList firstSubscriptionEvents,
        Api::AllSubscriptionEvents(transaction));
    WEAVERRUNTIME_TRY(
        Api::OwnershipChangeList firstOwnershipChangeList,
        Api::OwnershipChanges(transaction));

    WEAVERRUNTIME_TRY(
        Api::SubscriptionChangeList secondSubscriptionEvents,
        Api::AllSubscriptionEvents(transaction));
    WEAVERRUNTIME_TRY(
        Api::OwnershipChangeList secondOwnershipChangeList,
        Api::OwnershipChanges(transaction));

    /**
     * secondSubscriptionEvents and secondOwnershipChangeList are
     * both empty because there are no changes since the last call.
     */
}
```

> **Note**
>
> The function `AllSubscriptionEvents()` is implemented but the function `SubscriptionEvents()` is **not implemented**.

Release read locks after processing SubscriptionChangeList

When you begin an update, there are shared memory segments for the committed data in other partitions for the previous tick. These shared memory segments might be locked by readers. An app can't fully commit until all the readers have released the locks. As an optimization, an app should call `Api::ReleaseReadLeases()` to release the locks after processing `Api::SubscriptionChangelist` items. This reduces contention at commit time.

`Api::Commit()` releases the read leases by default, but it's a best practice to manually release them after processing subscription updates.

Example

```
Result<void> ProcessSubscriptionChanges(Transaction& transaction)
{
    WEAVERRUNTIME_TRY(ProcessSubscriptionChanges(transaction));

    /**
     * Done processing Api::SubscriptionChangeList items.
     * Release read locks.
     */

    WEAVERRUNTIME_EXPECT(Api::ReleaseReadLeases(transaction));

    ...
}
```

Create a standalone app instance for testing

You can use `Api::CreateStandaloneApplication()` to create a standalone app to test app logic before running the code in an actual simulation.

Example

```
int main(int argc, char* argv[])
{
    Api::StandaloneRuntimeConfig config = {
        /* run_for_seconds (the lifetime of the app) */ 3,
        /* tick_hertz (the app clock rate) */ 10 };

    Result<Application> applicationResult =
        Api::CreateStandaloneApplication(config);

    ...
}
```

AWS SimSpace Weaver demo framework

The AWS SimSpace Weaver demo framework (demo framework) is a library of utilities that you can use to develop SimSpace Weaver apps.

The demo framework provides

- Code samples and programming patterns for you to use and examine
- Abstractions and utility functions that streamline development for simple apps
- A simpler way to test experimental features of the SimSpace Weaver app SDK

We designed the SimSpace Weaver app SDK with low-level access to SimSpace Weaver APIs in order to deliver higher performance. In contrast, we designed the demo framework to provide higher-level abstractions and access to APIs that make SimSpace Weaver easier to use. The cost of ease of use is a lower level of performance compared to directly using the SimSpace Weaver app SDK. Simulations that can tolerate lower performance (such as those without real-time performance requirements) might be good candidates to use the demo framework. We recommend that you use the native functionality in the SimSpace Weaver app SDK for complex applications because the demo framework isn't a complete toolkit.

The demo framework includes

- Working code samples that support and demonstrate:
  - App flow management
  - Callback-driven entity event processing
- A set of third-party utility libraries:
  - spdlog (a logging library)
  - A minimal version of AZCore (a math library) that contains only:
    - Vector3
    - Aabb
  - cxxopts (a command line option parser library)
- Utility functions specific to SimSpace Weaver

The demo framework consists of a library, source files, and CMakeLists. The files are included in the SimSpace Weaver app SDK distributable package.

Working with service quotas

This section describes how to work with the service quotas for SimSpace Weaver. **Quotas** are also called **limits**. For a list of service quotas, see SimSpace Weaver endpoints and quotas. The APIs in

this section are from the set of **app APIs**. App APIs are a different than the service APIs. The app APIs are part of the SimSpace Weaver app SDK. You can find the documentation for the app APIs in the app SDK folder on your local system:

```
sdk-folder\SimSpaceWeaverAppSdk-sdk-version\documentation\index.html
```

Topics

Get the limits for an app

You can use the **RuntimeLimits** app API to query the limits for an app.

```
Result<Limit> RuntimeLimit(Application& app, LimitType type)
```

Parameters

Application& app

A reference to the app.

LimitType type

An enum with the following limit types:

```
enum LimitType {
    Unset = 0,
    EntitiesPerPartition = 1,
    RemoteEntityTransfers = 2,
    LocalEntityTransfers = 3
};
```

The following example queries the entity count limit.

```
WEAVERRUNTIME_TRY(auto entity_limit,
    Api::RuntimeLimit(m_app, Api::LimitType::EntitiesPerPartition))
Log::Info("Entity count limit", entity_limit.value);
```

Get the amount of resources used by an app

You can call the **RuntimeMetrics** app API to get the amount of resources used by an app:

```
Result<std::reference_wrapper<const AppRuntimeMetrics>> RuntimeMetrics(Application&
 app) noexcept
```

Parameters

Application& app

A reference to the app.

The API returns a reference to a struct that contains the metrics. A counter metric holds a running total value and only increases. A gauge metric holds a value that can increase or decrease. The application runtime updates a counter whenever an event increases the value. The runtime only updates the gauges when you call the API. SimSpace Weaver guarantees that the reference is valid for the lifetime of the app. Repeat calls to the API won't change the reference.

```
struct AppRuntimeMetrics {
    uint64_t total_committed_ticks_gauge,

    uint32_t active_entity_gauge,
    uint32_t ticks_since_reset_counter,

    uint32_t load_field_counter,
    uint32_t store_field_counter,

    uint32_t created_entity_counter,
    uint32_t deleted_entity_counter,

    uint32_t entered_entity_counter,
    uint32_t exited_entity_counter,

    uint32_t rejected_incoming_transfer_counter,
    uint32_t rejected_outgoing_transfer_counter
```

```
}
```

Reset metrics

The **ResetRuntimeMetrics** app API resets the values in the `AppRuntimeMetrics` struct.

```
Result<void> ResetRuntimeMetrics(Application& app) noexcept
```

The following example demonstrates how you can call **ResetRuntimeMetrics** in your app.

```
if (ticks_since_last_report > 100)
{
    auto metrics = WEAVERRUNTIME_EXPECT(Api::RuntimeMetrics(m_app));
    Log::Info(metrics);

    ticks_since_last_report = 0;

    WEAVERRUNTIME_EXPECT(Api::ResetRuntimeMetrics(m_app));
}
```

Exceeding a limit

An app API call that exceeds a limit will return an `ErrorCode::CapacityExceeded`, except for entity transfers. SimSpace Weaver handles entity transfers asynchronously as part of **Commit** and **BeginUpdate** app API operations, so there isn't a specific operation that returns an error if a transfer fails because of the entity transfer limit. To detect transfer failures, you can compare the current values of `rejected_incoming_transfer_counter` and `rejected_outgoing_transfer_counter` (in the `AppRuntimeMetrics` struct) with their previous values. Rejected entities won't be in the partition, but the app can still simulate them.

Running out of memory

SimSpace Weaver uses a garbage collector process to clean up and release freed memory. It's possible to write data faster than the garbage collector can release memory. If this happens, write operations might exceed the app's reserved memory limit. SimSpace Weaver will return an internal error with a message that contains `OutOfMemory` (and additional details). For more information, see Spread writes across time.

Best practices

The following best practices are general guidelines for designing your apps to avoid exceeding limits. They might not apply to your specific app design.

Monitor frequently and slow down

You should monitor your metrics frequently and slow down operations that are close to reaching a limit.

Avoid exceeding subscription limits and transfer limits

If possible, design your simulation to reduce the number of remote subscriptions and entity transfers. You can use placement groups to place multiple partitions on the same worker and reduce the need for remote entity transfers between workers.

Spread writes across time

The number and size of updates in a tick can have a significant impact on the time and memory required to commit a transaction. Large memory requirements can cause the application runtime to run out of memory. You can spread writes across time to lower the average total size of updates per tick. This can help improve performance and avoid exceeding limits. We recommend that you don't write more than an average of 12 MB on each tick or 1.5 KB for each entity.

Debugging simulations

You can use the following methods to get information about your simulations.

Topics

Use SimSpace Weaver Local and look at console output

We recommend that you develop your simulations locally first and then run them in the AWS Cloud. You can view console output directly when you run with SimSpace Weaver Local. For more information, see Local development.

Look at your logs in Amazon CloudWatch Logs

When you run your simulation in the AWS Cloud the console output of your apps is sent to log streams in Amazon CloudWatch Logs. Your simulation also writes other log data. You must enable logging in your simulation schema if you want your simulation to write log data. For more information, see SimSpace Weaver logs in Amazon CloudWatch Logs.

> **Warning**
>
> Your simulation can produce large amounts of log data. The log data can grow very quickly. You should watch your logs closely and stop your simulations when you don't need them running anymore. Logging can create large costs.

Use describe API calls

You can use the following service APIs to get information about your simulations in the AWS Cloud.

> **Important**
>
> If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

- **ListSimulations** – get a list of all of your simulations in the AWS Cloud.

Example

```
tools\windows\weaver-MyProject-cli.bat list-simulations
```

- **DescribeSimulation** – get details about a simulation.

  Example

  ```
  tools\windows\weaver-MyProject-cli.bat describe-simulation --simulation MySimulation
  ```

- **DescribeApp** – get details about an app.

  Example

  ```
  tools\windows\weaver-MyProject-cli.bat describe-app --simulation MySimulation --
  domain MyCustomDomain --app MyCustomApp
  ```

For more information about the SimSpace Weaver APIs, see SimSpace Weaver API references.

Connect a client

You can connect a client to a running custom or service app that you defined with an `endpoint_config` in your simulation schema. The SimSpace Weaver app SDK includes sample clients that you can use to view the sample application. You can look at the source code for these sample clients and the sample application to see how you can create your own clients. For more information about how to build and run the sample clients, see Step 5: View your simulation.

You can find the source code for the sample clients in the following folder:

- *sdk-folder*`\packaging-tools\clients\PathfindingSampleClients\`

Debugging local simulations

You can debug your SimSpace Weaver Local apps with Microsoft Visual Studio. For more information about how to debug with Visual Studio, see the Microsoft Visual Studio documentation.

To debug your local simulation

1. Make sure that your `schema.yaml` is in your working directory.
2. In **Visual Studio**, open the context menu for each app that you want to debug (such as `PathfindingSampleLocalSpatial` or `PathfindingSampleLocalView`) and set the working directory in the debugging section.
3. Open the context menu for the app you want to debug and select **Set as Startup project**.
4. Choose **F5** to start debugging the app.

The requirements to debug a simulation are the same as the requirements to run a simulation normally. You must start the number of spatial apps specified in the schema. For example, if your schema specifies a 2x2 grid and you start a spatial app in debug mode, the simulation will not run until you start 3 more spatial apps (in debug mode or not in debug mode).

To debug a custom app, you must first start your spatial apps and then start the custom app in the debugger.

Note that your simulation runs in lock step. As soon as an app reaches a breakpoint, all other apps will pause. After you continue from that breakpoint, the other apps will continue.

Custom containers

AWS SimSpace Weaver apps run in containerized Amazon Linux 2 (AL2) environments. In the AWS Cloud, SimSpace Weaver runs your simulations in Docker containers built from an `amazonlinux:2` image served from Amazon Elastic Container Registry (Amazon ECR). You can create a custom Docker image, store it in in Amazon ECR, and use that image for your simulation instead of the default Docker image that we provide.

You can use a custom container to manage your software dependencies and include additional software components that aren't in the standard Docker image. For example, you can add the publicly-available software libraries that your app uses to the container and only put your custom code in the app zip file.

> **Important**
>
> We only support AL2 Docker images hosted in Amazon ECR repositories, either in Amazon ECR Public Gallery or your private Amazon ECR registry. We don't support Docker images

hosted outside Amazon ECR. For more information about Amazon ECR, see *Amazon Elastic Container Registry Documentation*.

Topics

Create a custom container

These instructions assume that you know how to use Docker and Amazon Elastic Container Registry (Amazon ECR). For more information about Amazon ECR, see the *Amazon ECR User Guide*.

Prerequisites

- The IAM identity (use or role) that you use to perform these actions has the correct permissions to use Amazon ECR
- Docker is installed on your local system

To create a custom container

1. Create your `Dockerfile`.

   A `Dockerfile` to run AWS SimSpace Weaver apps starts with the Amazon Linux 2 image in Amazon ECR.

   ```
   # parent image required to run AWS SimSpace Weaver apps
   FROM public.ecr.aws/amazonlinux/amazonlinux:2
   ```

2. Build your `Dockerfile`.
3. Upload your container image to Amazon ECR.
   - Use the AWS Management Console.
   - Use the AWS Command Line Interface.

> **Note**
>
> If you get an `AccessDeniedException` error when you try to upload your container image to Amazon ECR, your IAM identity (user or role) might not have the necessary permissions to use Amazon ECR. You can attach the `AmazonEC2ContainerRegistryPowerUser` AWS managed policy to your IAM identity and try again. For more information about how to attach a policy, see Adding and removing IAM identity permissions in the *AWS Identity and Access Management User Guide*.

Modify a project to use a custom container

These instructions assume that you already know how to use AWS SimSpace Weaver and want to make your app storage and development workflows in the AWS Cloud more efficient.

Prerequisites

- You are modifying an existing SimSpace Weaver project created by the `create-project.bat` script.
- You have a custom container in Amazon Elastic Container Registry (Amazon ECR). For more information about creating a custom container, see Create a custom container.

To modify your project to use a custom container

1. Add permissions to your project's simulation app role to use Amazon ECR.

   a. If you don't already have an IAM policy with following permissions, create the policy. We suggest the policy name `simspaceweaver-ecr`. For more information about how to create an IAM policy, see Creating IAM policies in the *AWS Identity and Access Management User Guide*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Statement",
            "Effect": "Allow",
```

```
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:GetAuthorizationToken"
            ],
            "Resource": "*"
        }
    ]
}
```

b. Find the name of your project's simulation app role:

i. In a text editor, open the CloudFormation template for your project:

```
project-folder\cloudformation\weaver-project-name-stack.yaml
```

ii. Find the `RoleName` property under `WeaverAppRole`. The value is the name of your project's simulation app role.

Example

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  WeaverAppRole:
    Type: 'AWS::IAM::Role'
    Properties:
      RoleName: 'weaver-MySimulation-app-role'
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
        - Effect: Allow
          Principal:
            Service:
              - 'simspaceweaver.amazonaws.com'
```

c. Attach the `simspaceweaver-ecr` policy to the project's simulation app role. For more information about how to attach a policy, see Adding and removing IAM identity permissions in the *AWS Identity and Access Management User Guide*.

2. Specify your container images in the project's simulation schema.

- You can add the optional `default_image` property under `simulation_properties` to specify a default custom container image for all domains.

- Add the `image` property in the `app_config` for a domain that you want to use a custom container image. Specify the Amazon ECR repository URI as the value. You can specify a different image for each domain.
  - If `image` isn't specified for a domain and `default_image` is specified, apps in that domain use the default image.
  - If `image` isn't specified for a domain and `default_image` isn't specified, apps in that domain run in a standard SimSpace Weaver container.

Example Schema snippet that includes custom container settings

```
sdk_version: "1.15.3"
simulation_properties:
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
  default_entity_index_key_type: "Vector3<f32>"
  default_image: "111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-
repository:latest" # image to use if no image specified for a domain
domains:
  MyCustomDomain:
    launch_apps_via_start_app_call: {}
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MyViewApp.zip"
      launch_command: ["MyViewApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 7000
      image: "111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-
repository:latest" # custom container image to use for this domain
  MySpatialDomain:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 2
        y: 2
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MySpatialApp.zip"
      launch_command: ["MySpatialApp"]
      required_resource_units:
        compute: 1
```

```
      image: "111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-
repository:latest" # custom container image to use for this domain
```

3. Build and upload your project as usual.

Frequently asked questions about custom containers

Q1. What do I do if I want to change the contents of my container?

- **For a running simulation** – You can't change the container for a running simulation. You must build a new container and start a new simulation that uses that container.
- **For a new simulation** – Build a new container, upload it to Amazon Elastic Container Registry (Amazon ECR), and start a new simulation that uses that container.

Q2. How can I change the container image for my simulation?

- **For a running simulation** – You can't change the container for a running simulation. You must start a new simulation that uses the new container.
- **For a new simulation** – Specify the new container image in your project's simulation schema. For more information, see Modify a project to use a custom container.

Troubleshooting custom containers

Topics

AccessDeniedException when uploading your image to Amazon Elastic Container Registry (Amazon ECR)

If you get an `AccessDeniedException` error when you try to upload your container image to Amazon ECR, your IAM identity (user or role) might not have the necessary permissions to use Amazon ECR. You can attach the `AmazonEC2ContainerRegistryPowerUser` AWS managed policy to your IAM identity and try again. For more information about how to attach a policy, see

Adding and removing IAM identity permissions in the *AWS Identity and Access Management User Guide*.

A simulation that uses a custom container fails to start

Troubleshooting tips

- If logging is enabled for your simulation, check your error logs. For more information, see the detailed tutorial.
- Test your simulation without a custom container.
- Test your simulation locally. For more information, see Local development.

Working with Python

You can use Python for your SimSpace Weaver apps and client. The Python software development kit (Python SDK) is included as part of the standard SimSpace Weaver app SDK distributable package. Development with Python works in a similar way as development in the other supported languages.

> **Important**
>
> SimSpace Weaver only supports Python version 3.9.

> **Important**
>
> SimSpace Weaver support for Python requires SimSpace Weaver version 1.15.0 or later.

Topics

- Creating a Python project
- Starting a Python simulation
- The sample Python client
- Writing your own build scripts
- Frequently asked questions about using Python

- Troubleshooting problems related to Python

Creating a Python project

You use the `create-project.bat` script to create a Python project, the same way as you would to create a non-Python project. You can use the `PythonBubblesSample` template as the starting point for your Python project. See Create a Python project, below.

Python custom container

To run your Python-based SimSpace Weaver simulation in the AWS Cloud, you can create a custom container that includes the necessary dependencies. For more information, see Custom containers.

A Python custom container must include the following:

- gcc
- openssl-devel
- bzip2-devel
- libffi-devel
- wget
- tar
- gzip
- make
- Python (version 3.9)

If you use the `PythonBubblesSample` template to create your project, you can run the `create-custom-container.bat` script (located in the `tools` folder of your project) to create a Docker image with the necessary dependencies. The script uploads the image to Amazon Elastic Container Registry (Amazon ECR).

The `create-custom-container.bat` script uses the following `Dockerfile`:

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2
RUN yum -y install gcc openssl-devel bzip2-devel libffi-devel
RUN yum -y install wget
```

```
RUN yum -y install tar
RUN yum -y install gzip
RUN yum -y install make
WORKDIR /opt
RUN wget https://www.python.org/ftp/python/3.9.0/Python-3.9.0.tgz
RUN tar xzf Python-3.9.0.tgz
WORKDIR /opt/Python-3.9.0
RUN ./configure --enable-optimizations
RUN make altinstall
COPY requirements.txt ./
RUN python3.9 -m pip install --upgrade pip
RUN pip3.9 install -r requirements.txt
```

You can add your own dependencies to the `Dockerfile`:

```
RUN yum -y install dependency-name
```

The `requirements.txt` file contains a list of Python packages required for the `PythonBubblesSample` sample simulation:

```
Flask==2.1.1
```

You can add your own Python package dependencies to the `requirements.txt`:

```
package-name==version-number
```

The `Dockerfile` and `requirements.txt` are in the `tools` folder of your project.

Important

You must run `create-custom-container.bat` after any changes to the `Dockerfile` or `requirements.txt`.

Important

You technically don't have to use a custom container with your Python simulation, but we strongly recommend that you use a custom container. The standard Amazon Linux 2 (AL2) container that we provide doesn't have Python. Therefore, if you don't use a custom

container that has Python (such as the container image created by the `create-custom-container.bat` script), you must include Python and the required dependencies in each app zip file that you upload to SimSpace Weaver.

Create a Python project

The following procedure is for Microsoft Windows. If you are using Windows Subsystem for Linux (WSL), use the `.sh` versions of the `.bat` scripts instead. You must complete the setup for Amazon Elastic Container Registry (Amazon ECR) to use this procedure. For more information, see Setting up with Amazon ECR in the *Amazon ECR User Guide*.

To create a Python project

1. In a **command prompt window**, go to your SimSpace Weaver SDK folder.

   ```
   cd sdk-folder
   ```

2. Run `create-project.bat` with the `PythonBubblesSample` template.

   ```
   .\create-project.bat --name project-name --path project-folder-parent-path --
   template PythonBubblesSample
   ```

3. Go to the `tools` folder in your project folder. Your *`project-folder`* is *`project-folder-parent-path`**`project-name`*.

   ```
   cd project-folder\tools
   ```

4. Create the custom container.

   ```
   .\create-custom-container.bat
   ```

Starting a Python simulation

You can start your Python-based simulation the same way as a regular SimSpace Weaver simulation, both in SimSpace Weaver Local and in SimSpace Weaver in the AWS Cloud. For more information, see the following:

SimSpace Weaver Local

- Local development

AWS Cloud

- Step 3: Run the quick start script in the quick start tutorial
- Detailed tutorial: Learn the details while building the sample application

The `PythonBubblesSample` includes its own Python sample client. For more information, see The sample Python client.

The sample Python client

If you use the `PythonBubblesSample` template to create a project then your project contains a Python sample client. You can use the sample client to view the `PythonBubblesSample` simulation. You can also use the sample client as the starting point to create your own Python client.

The following procedure assumes that you created a `PythonBubblesSample` project and started its simulation.

To start the Python client

1. In a **command prompt window**, go to the `src\PythonBubblesSample\bin` folder of your project.

   ```
   cd project-folder\src\PythonBubblesSample\bin
   ```

2. Run the Python client.

   ```
   python bubbles_tkinter_client.py --host ip-address --port port-number --
   simsize max-entitites
   ```

Parameters

host

The IP address of your simulation. For a simulation started in the AWS Cloud, you can find your simulation's IP address in the SimSpace Weaver console or use the procedure in Step 4: Get your IP address and port number of the quick start tutorial. For a local simulation, use `127.0.0.1` as the IP address.

port

The port number of your simulation. For a simulation started in the AWS Cloud, this is the `Actual` port number. You can find your simulation's port number in the SimSpace Weaver console or use the procedure in Step 4: Get your IP address and port number of the quick start tutorial. For a local simulation, use `7000` as the port number.

simsize

The maximum number of entities to display in the client.

Writing your own build scripts

You can write your own build scripts for your Python simulation. To have a successful build, the following steps must occur:

1. Copy the contents of `src/PythonBubblesSample/` into the `Build/out` directory.
2. Copy the contents of `${WEAVER_SDK_DIRECTORY}/lib/weaver/weaver_python_app_sdk_v1` into the `Build/out/lib/weaver_app_sdk_v1` directory.
3. Copy `${WEAVER_SDK_DIRECTORY}/lib/weaver/libweaver_app_sdk_python_v1_39.so` into the `Build/out/lib/weaver_app_sdk_v1` directory.
4. Rename `Build/out/lib/weaver_app_sdk_v1/libweaver_app_sdk_python_v1_39.so` to `libweaver_app_sdk_python_v1.so`.
5. Zip the contents of the `Build/out/` directory.
6. Repeat the zip process for each app zip specified in the simulation's schema. For the `PythonBubblesSample`, the schema expects a *`project-name`*`Spatial.zip` and a *`project-name`*`View.zip`.

After these steps, the zip files are ready to upload to the project's Amazon S3 bucket.

Frequently asked questions about using Python

Q1. What versions of Python are supported?

SimSpace Weaver only supports Python version 3.9.

Troubleshooting problems related to Python

Topics

Failure during custom container creation

If you get an error `no basic auth credentials` after you run `create-custom-container.bat` then there could be a problem with your temporary credentials for Amazon ECR. Run the following command with your AWS Region ID and AWS account number:

```
aws ecr get-login-password --region region | docker login --username AWS --password-stdin account_id.dkr.ecr.region.amazonaws.com
```

Example

```
aws ecr get-login-password --region us-west-2 | docker login --username AWS --password-stdin 111122223333.dkr.ecr.region.amazonaws.com
```

> **Important**
>
> Make sure that the AWS Region you specify is the same one that you use for your simulation. Use one of the AWS Regions that SimSpace Weaver supports. For more information, see SimSpace Weaver endpoints and quotas.

After you run the `aws ecr` command, run `create-custom-container.bat` again.

Other troubleshooting resources to check

- Troubleshooting custom containers
- Amazon ECR troubleshooting in the *Amazon ECR User Guide*
- Setting up with Amazon ECR in the *Amazon ECR User Guide*

Your Python simulation fails to start

You might see an `Unable to start app` error in your simulation's management log. This can happen if your custom container creation failed. For more information, see Failure during custom container creation. For more information about logs, see SimSpace Weaver logs in Amazon CloudWatch Logs.

If you're sure that there's nothing wrong with your container, check your app's Python source code. You can use SimSpace Weaver Local to test your app. For more information, see Local development.

A Python simulation or view client throws a ModuleNotFound error

Python throws a `ModuleNotFound` error when it can't locate a required Python package.

If your simulation is in the AWS Cloud, make sure that your custom container has all of the required dependencies listed in your `requirements.txt`. Remember to run `create-custom-container.bat` again if you edit `requirements.txt`.

If you get the error for the `PythonBubblesSample` client, use `pip` to install the indicated package:

```
pip install package-name==version-number
```

Support for other engines

You can use your own custom C++ engine with SimSpace Weaver. We are currently developing support for the following engines. There is separate documentation for each of these engines.

> **Important**
>
> Integrations with the engines listed here are experimental. They are available for preview.

Engines

- Unity (minimum version 2021.3.7f1)
- Unreal Engine (minimum version 5.0)

Unity

You must have the Unity development environment already installed before you build SimSpace Weaver simulations with Unity. Download the separate Unity SDK for AWS SimSpace Weaver (Unity SDK) and follow the instructions in that package.

> **Important**
>
> You must use the latest version of the SimSpace Weaver app SDK. The latest version is 1.15.3. For more information, see AWS SimSpace Weaver versions.

To download and use the Unity SDK

1. At a **Windows command prompt**, go to your *sdk-folder*.
2. Run the download script. Replace *region* with the AWS Region where you will start your simulation (for example, `us-west-2`).

   ```
   .\download-unity-package.bat --region region
   ```

   The script downloads and unzips `SimSpaceWeaverUnityPackage.zip` in your current folder.
3. Read `SimSpaceWeaverUnityPackage\Release\Documentation\Unity_SDK_for_AWS_SimSpace_Weaver.pdf`.

> **Important**
>
> If you get an error in Unity about a missing namespace for `JsonProperty` or `JsonAttribute`, follow these steps to add the NewtonsoftJson package:
>
> 1. In the Unity editor, choose **Window** > **Package Manager** from the menu bar.
> 2. In the Package Manager window, choose the **+** (plus) button at the top of the window.

3. Select **Add package from git URL**.
4. Enter the following:

```
com.unity.nuget.newtonsoft-json
```

5. Choose **Add**.

Important

The Unity SDK doesn't support named profiles for the AWS Command Line Interface (AWS CLI). If you use AWS IAM Identity Center or AWS CLI profiles, you must copy or rename your named profile to the `default` profile before you use the Unity SDK. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

Unreal Engine

You must build an Unreal Engine dedicated server from source code. The SimSpaceWeaverAppSdkDistributable includes a version of the PathfindingSample for Unreal Engine. For more information, see the separate directions:

```
sdk-folder\AWS_SimSpace_Weaver_Unreal_Guide.pdf
```

Use of licensed software with AWS SimSpace Weaver

AWS SimSpace Weaver allows you to build simulations with your choice of simulation engine and content. In connection with your use of SimSpace Weaver, you are responsible for obtaining, maintaining, and adhering to the license terms of any software or content you use in your simulations. Verify your licensing agreement allows you to deploy your software and content in a virtual hosted environment.

Managing your resources with AWS CloudFormation

You can use AWS CloudFormation to manage your AWS SimSpace Weaver resources. CloudFormation is a separate AWS service that helps you specify, provision, and manage your AWS infrastructure as code. With CloudFormation you create a JSON or YAML file, called a *template*. Your template specifies the details of your infrastructure. CloudFormation uses your template to provision your infrastructure as a single unit, called a *stack*. When you delete your stack, you can have CloudFormation delete everything in the stack at the same time. You can manage your template using standard source code management processes (for example, tracking it in a version control system like Git). For more information about CloudFormation, see the *AWS CloudFormation User Guide*.

Your simulation resource

In AWS, a *resource* is an entity that you can work with. Examples include an Amazon EC2 instance, an Amazon S3 bucket, or an IAM role. Your SimSpace Weaver simulation is a resource. In configurations, you usually specify an AWS resource in the form `AWS::service::resource`. For SimSpace Weaver, you specify your simulation resource as `AWS::SimSpaceWeaver::Simulation`. For more information about your simulation resource in CloudFormation, see the SimSpace Weaver section in the *AWS CloudFormation User Guide*.

How can I use CloudFormation with SimSpace Weaver?

You can create an CloudFormation template that specifies the AWS resources that you want to provision. Your template can specify an entire architecture, part of an architecture, or a small solution. For example, you could specify an architecture for your SimSpace Weaver solution that includes Amazon S3 buckets, IAM permissions, a supporting database in Amazon Relational Database Service or Amazon DynamoDB, and your `Simulation` resource. You can then use CloudFormation to provision all of those resources as a unit, and at the same time.

Example template that creates IAM resources and starts a simulation

The following example template creates an IAM role and permissions that SimSpace Weaver needs to perform actions in your account. The SimSpace Weaver app SDK scripts create the role and permissions in a specific AWS Region when you create a project, but you can use an CloudFormation template to deploy the simulation to another AWS Region without running the scripts again. For example, you can do this to set up a backup simulation for disaster recovery purposes.

In this example, the original simulation name is `MySimulation`. A bucket for the schema already exists in the AWS Region where CloudFormation will build the stack. The bucket contains a version of the schema that is properly configured to run the simulation in that AWS Region. Recall that the schema specifies the location of your app zip files, which is an Amazon S3 bucket in the same AWS Region as the simulation. The app zips bucket and files must already exist in the AWS Region when CloudFormation builds the stack, otherwise your simulation won't start. Note that the bucket name in this example includes the AWS Region, but that doesn't determine where the bucket is actually located. You must make sure that the bucket is actually in that AWS Region (you can check the bucket properties in the Amazon S3 console, with the Amazon S3 APIs, or with the Amazon S3 commands in the AWS CLI).

This example uses some built-in functions and parameters in CloudFormation to perform variable substitution. For more information, see Intrinsic function reference and Pseudo parameters reference in the *AWS CloudFormation User Guide*.

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  WeaverAppRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: SimSpaceWeaverAppRole
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
        - Effect: Allow
          Principal:
            Service:
              - simspaceweaver.amazonaws.com
          Action:
            - sts:AssumeRole
      Path: /
      Policies:
        - PolicyName: SimSpaceWeaverAppRolePolicy
          PolicyDocument:
            Version: 2012-10-17
            Statement:
            - Effect: Allow
              Action:
                - logs:PutLogEvents
                - logs:DescribeLogGroups
                - logs:DescribeLogStreams
```

```
                - logs:CreateLogGroup
                - logs:CreateLogStream
              Resource: *
            - Effect: Allow
              Action:
                - cloudwatch:PutMetricData
              Resource: *
            - Effect: Allow
              Action:
                - s3:ListBucket
                - s3:PutObject
                - s3:GetObject
              Resource: *
  MyBackupSimulation:
    Type: AWS::SimSpaceWeaver::Simulation
    Properties:
      Name: !Sub 'mySimulation-${AWS::Region}'
      RoleArn: !GetAtt WeaverAppRole.Arn
      SchemaS3Location:
        BucketName: !Sub 'weaver-mySimulation-${AWS::AccountId}-schemas-${AWS::Region}'
        ObjectKey: !Sub 'schema/mySimulation-${AWS::Region}-schema.yaml'
```

Using snapshots with AWS CloudFormation

A snapshot is a backup of a simulation. The following example starts a new simulation from a snapshot instead of from a schema. The snapshot in this example was created from a SimSpace Weaver app SDK project simulation. CloudFormation creates the new simulation resource and initializes it with data from the snapshot. The new simulation can have a different `MaximumDuration` than the original simulation.

We recommend that you make and use a copy of your original simulation's app role. The original simulation's app role could be deleted if you delete that simulation's CloudFormation stack.

```
Description: "Example - Start a simulation from a snapshot"
Resources:
  MyTestSimulation:
    Type: "AWS::SimSpaceWeaver::Simulation"
    Properties:
      MaximumDuration: "2D"
      Name: "MyTestSimulation_from_snapshot"
```

```
    RoleArn: "arn:aws:iam::111122223333:role/weaver-MyTestSimulation-app-role-copy"

    SnapshotS3Location:
      BucketName: "weaver-mytestsimulation-111122223333-artifacts-us-west-2"
      ObjectKey: "snapshot/MyTestSimulation_22-12-15_12_00_00-230428-1207-13.zip"
```

Snapshots

You can create a *snapshot* to back up your simulation entity data at any point in time. SimSpace Weaver creates a .zip file in an Amazon S3 bucket. You can create a new simulation with the snapshot. SimSpace Weaver initializes the State Fabric of your new simulation with the entity data stored in the snapshot, starts the spatial and service apps that were running when the snapshot was created, and sets the clock to the appropriate tick. SimSpace Weaver gets the configuration of your simulation from the snapshot instead of from a schema file. Your app .zip files must be in the same location in Amazon S3 as they were in the original simulation. You must start any custom apps separately.

Topics

Use cases for snapshots

Return to a previous state and explore branching scenarios

You can create a snapshot of your simulation to save it at a specific state. You can then create multiple new simulations from that snapshot and explore different scenarios that could branch from that state.

Disaster recovery and security best practices

We recommend that you regularly backup your simulation, especially for simulations that run for more than 1 hour or use multiple workers. Backups can help you recover from disasters and security incidents. Snapshots provide a way for you to backup your simulation. Snapshots require your app .zip files to exist in the same location in Amazon S3 as they were before. If you need to be able to move your app .zip files to another location, you must use a custom backup solution.

For more information about other best practices, see Best practices when working with SimSpace Weaver and Security best practices for SimSpace Weaver.

Extend the duration of your simulation

Your *simulation resource* is the representation of your simulation in SimSpace Weaver. All simulation resources have a `MaximumDuration` setting. A simulation resource automatically stops when it reaches its `MaximumDuration`. The maximum value of `MaximumDuration` is `14D` (14 days).

If you need your simulation to persist longer than the `MaximumDuration` of its simulation resource, you can create a snapshot before the simulation resource reaches its `MaximumDuration`. You can start a new simulation (create a new simulation resource) with your snapshot. SimSpace Weaver initializes your entity data from the snapshot, starts the same spatial and service apps that ran before, and restores the clock. You can start your custom apps and perform any additional custom initialization. You can set the `MaximumDuration` of the new simulation resource to a different value when you start it.

Use the SimSpace Weaver app SDK to work with snapshots

You can use scripts provided in the SimSpace Weaver app SDK (minimum version 1.13) to create and use snapshots.

The SimSpace Weaver app SDK organizes your simulations by *project*. You can start multiple simulations from a single project. Each of those simulations uses the same schema and app .zip files. The SimSpace Weaver app SDK scripts place the assets for a simulation in a specific Amazon S3 bucket based on the project name, AWS account number, and AWS Region. The scripts work with snapshot files located in a `snapshot` folder at the root of that bucket. The Amazon S3 URI to the `snapshot` folder has following form:

```
s3://weaver-project-name-lowercase-account-number-artifacts-region/snapshot
```

Example

- **Project name:** `MyProject`
- **AWS account number:** `111122223333`
- **AWS Region:** `us-west-2`
- **Snapshot folder Amazon S3 URI:** `s3://weaver-myproject-111122223333-artifacts-us-west-2/snapshot`

If you want to use a different Amazon S3 bucket, see the following alternative ways to work with snapshots.

Other ways to work with snapshots

- SimSpace Weaver console
- AWS CLI
- SimSpace Weaver APIs

Topics

Use the SimSpace Weaver app SDK to create a snapshot

To create a snapshot, your simulation must be in the `STARTED` state. The snapshot creation starts after the current tick completes. SimSpace Weaver stops sending ticks to the apps but the clock status still indicates `STARTED`. The simulation status changes to `SNAPSHOT_IN_PROGRESS`. After the snapshot is finished, the simulation state changes back to `STARTED` and the apps receive ticks again.

To create a snapshot

1. At a **Windows command prompt**, go to the tools folder for your project.

```
cd project-folder\tools\windows
```

2. If you don't know the name of your simulation, call the `list-simulations` API to see a list of your simulation resources. Make sure that the simulation status is STARTED.

```
.\weaver-project-name-cli.bat list-simulations
```

3. Run the `create-snapshot` script for your project.

```
.\create-snapshot-project-name.bat --simulation simulation-name
```

Example

```
.\create-snapshot-MyProject.bat --simulation MyProjectSimulation_23-04-29_12_00_00
```

SimSpace Weaver creates the snapshot file in the artifacts bucket for your project.

Example

- **Project name:** `MyProject`
- **AWS account number:** 111122223333
- **AWS Region:** `us-west-2`
- **Snapshot folder Amazon S3 URI:** `s3://weaver-myproject-111122223333-artifacts-us-west-2/snapshot`
- **Simulation name:** `MyProjectSimulation_23-04-29_12_00_00`
- **Snapshot time:** April 29, 2023, 15:30:27 UTC
- **Snapshot file name:** `MyProjectSimulation_23-04-29_12_00_00-230429-1530-27.zip`
- **Snapshot file Amazon S3 URI:** `s3://weaver-myproject-111122223333-artifacts-us-west-2/snapshot/MyProjectSimulation_23-04-29_12_00_00-230429-1530-27.zip`

Use the SimSpace Weaver app SDK to start a simulation from a snapshot

When you use an app SDK script to start a simulation from a snapshot, the scripts create a new simulation name the same way they do for starting a simulation without a snapshot.

Your snapshot file must exist in the snapshot location in Amazon S3 with the following Amazon S3 URI:

```
s3://weaver-project-name-lowercase-account-number-artifacts-region/snapshot
```

The app .zip files must be in the same location they were when the snapshot was created.

SimSpace Weaver creates a new simulation resource, initializes the State Fabric with entity data stored in the snapshot, starts new instances of the same spatial and service apps that were running when the snapshot was created, and sets the clock to the appropriate tick. You must start custom apps separately through the normal process.

The `start-from-snapshot` script is the snapshot version of the `start-simulation` script. Just like the `start-simulation` script, the `start-from-snapshot` script doesn't start the clock for you. You must start the clock separately.

To start a simulation from a snapshot

1. At a **Windows command prompt**, go to the tools folder for your project.

   ```
   cd project-folder\tools\windows
   ```

2. Run the `start-from-snapshot` script.

   ```
   .\start-from-snapshot-project-name.bat --snapshot-s3-file snapshot-file-name
   ```

   Example

   ```
   .\start-from-snapshot-MyProject.bat --snapshot-s3-file
    MyProjectSimulation_23-04-29_12_00_00-230429-1530-27.zip
   ```

Use the SimSpace Weaver app SDK to quick start a simulation from a snapshot

You can *quick start* a simulation from a snapshot. This is similar to a quick start without a snapshot.

Your snapshot file must exist in the snapshot location in Amazon S3 with the following Amazon S3 URI:

```
s3://weaver-project-name-lowercase-account-number-artifacts-region/snapshot
```

The app .zip files must be in the same location they were when the snapshot was created.

SimSpace Weaver creates a new simulation resource, initializes the State Fabric with entity data stored in the snapshot, starts new instances of the same spatial and service apps that were running when the snapshot was created, and sets the clock to the appropriate tick. You must start custom apps separately through the normal process.

The `quick-start-from-snapshot` script is the snapshot version of the `quick-start` script. Just like the `quick-start` script, the `quick-start-from-snapshot` script starts the clock for you. It also starts the view app for the pathfinding sample project.

To quick start a simulation from a snapshot

1. At a **Windows command prompt**, go to the tools folder for your project.

   ```
   cd project-folder\tools\windows
   ```

2. Run the `quick-start-from-snapshot` script.

   ```
   .\quick-start-from-snapshot-project-name-cli.bat --snapshot-s3-file snapshot-file-name
   ```

   Example

   ```
   .\quick-start-from-snapshot-MyProject-cli.bat --snapshot-s3-file
    MyProjectSimulation_23-04-29_12_00_00-230429-1530-27.zip
   ```

Use the SimSpace Weaver app SDK to list snapshots for a project

You can use the `list-snapshots` script to list the snapshots for a project. This script lists the files in the `snapshot` folder for the project. Projects are unique to the SimSpace Weaver app SDK, so you can only do this with the app SDK scripts and only for projects. The script assumes that all files in the `snapshot` folder on Amazon S3 are snapshot files. If you move or delete files from the folder then those files won't appear in the list.

To list the snapshots for a project

1. At a **Windows command prompt**, go to the tools folder for your project.

```
cd project-folder\tools\windows
```

2. Run the `list-snapshots` script.

```
.\list-snapshots-project-name.bat
```

Example

```
.\list-snapshots-MyProject.bat
```

Use the SimSpace Weaver console to work with snapshots

You can use the SimSpace Weaver console to create a snapshot of your simulation.

Other ways to work with snapshots

- SimSpace Weaver app SDK scripts
- AWS CLI
- SimSpace Weaver APIs

Topics

Use the console to create a snapshot

To create a snapshot

1. Sign to the AWS Management Console and connect to the SimSpace Weaver console.
2. Choose **Simulations** from the navigation pane.
3. Select the radio button next to simulation name. Your simulation's **Status** must be **Started**.
4. At the top of the page, choose **Create snapshot**.
5. Under **Snapshot settings**, for **Snapshot destination**, enter the Amazon S3 URI of a bucket or a bucket and folder where you want SimSpace Weaver to create your snapshot. You can choose **Browse S3** if you prefer to browse your available buckets and select a location.

> **Important**
>
> The Amazon S3 bucket must be in the same AWS Region as the simulation.

> **Note**
>
> SimSpace Weaver creates a `snapshot` folder inside your selected snapshot destination. SimSpace Weaver creates the snapshot .zip file in that `snapshot` folder.

6. Choose **Create snapshot**.

Use the console to start a simulation from a snapshot

To start a simulation from a snapshot, your snapshot .zip file must exist in an Amazon S3 bucket that your simulation can access. Your simulation uses permissions defined in the app role that you select when you start the simulation. All of the app .zip files from the original simulation must exist at their same locations as when the snapshot was created.

To start a simulation from a snapshot

1. Sign to the AWS Management Console and connect to the SimSpace Weaver console.
2. Choose **Simulations** from the navigation pane.
3. At the top of the page, choose **Start simulation**.
4. Under **Simulation settings**, enter a name and optional description for your simulation. Your simulation name must be unique in your AWS account.
5. For **Simulation start method**, choose **Use a snapshot in Amazon S3**.
6. For **Amazon S3 URI for snapshot**, enter the Amazon S3 URI of your snapshot file, or choose **Browse S3** to browse and select the file.

> **Important**
>
> The Amazon S3 bucket must be in the same AWS Region as the simulation.

7. For **IAM role**, select the app role that your simulation will use.

8. For **Maximum duration**, enter the maximum amount of time that your simulation resource should run for. The maximum value is 14D. For more information about the maximum duration, see .
9. Under **Tags - *optional***, choose **Add new tag** if you want to add a tag.
10. Choose **Start simulation**.

Use the AWS CLI to work with snapshots

You can use the AWS CLI to call the SimSpace Weaver APIs from a command prompt. You must have the AWS CLI installed and configured properly. For more information, see Installing or updating the latest version of the AWS CLI in the *AWS Command Line Interface User Guide for Version 2*.

Other ways to work with snapshots

- SimSpace Weaver app SDK scripts
- SimSpace Weaver console
- SimSpace Weaver APIs

Topics

- Use the AWS CLI to create a snapshot
- Use the AWS CLI to start a simulation from a snapshot

Use the AWS CLI to create a snapshot

To create a snapshot

- At a **command prompt**, call the `CreateSnapshot` API.

```
aws simspaceweaver create-snapshot --simulation simulation-name –destination s3-destination
```

Parameters

simulation

The name of a started simulation. You can use `aws simspaceweaver list-simulations` to see the names and statuses of your simulations.

destination

A string that specifies the destination Amazon S3 bucket and optional object key prefix for your snapshot file. Your object key prefix is usually a folder in your bucket. SimSpace Weaver creates your snapshot inside a `snapshot` folder at this destination.

> **Important**
>
> The Amazon S3 bucket must be in the same AWS Region as the simulation.

Example

```
aws simspaceweaver create-snapshot –simulation
 MyProjectSimulation_23-04-29_12_00_00 –destination BucketName=weaver-
myproject-111122223333-artifacts-us-west-2,ObjectKeyPrefix=myFolder
```

For more information about the `CreateSnapshot` API, see CreateSnapshot in the *AWS SimSpace Weaver API Reference*.

Use the AWS CLI to start a simulation from a snapshot

To start a simulation from a snapshot

- At a **command prompt**, call the `StartSimulation` API.

```
aws simspaceweaver start-simulation --name simulation-name --role-arn role-arn --
snapshot-s3-location s3-location
```

Parameters

name

The name of the new simulation. The simulation name must be unique in your AWS account. You can use `aws simspaceweaver list-simulations` to see the names of your existing simulations.

role-arn

The Amazon Resource Name (ARN) of the app role that your simulation will use.

snapshot-s3-location

A string that specifies the Amazon S3 bucket and object key of your snapshot file.

> **Important**
>
> The Amazon S3 bucket must be in the same AWS Region as the simulation.

Example

```
aws simspaceweaver start-simulation –name MySimulation –role-arn
 arn:aws:iam::111122223333:role/weaver-MyProject-app-role –snapshot-s3-location
 BucketName=weaver-myproject-111122223333-artifacts-us-west-2,ObjectKey=myFolder/
snapshot/MyProjectSimulation_23-04-29_12_00_00-230429-1530-27.zip
```

For more information about the `StartSimulation` API, see StartSimulation in the *AWS SimSpace Weaver API Reference*.

Use the SimSpace Weaver APIs to work with snapshots

You can call the SimSpace Weaver APIs directly to work with snapshots. For information about the APIs, see the AWS SimSpace Weaver API Reference.

Other ways to work with snapshots

- SimSpace Weaver app SDK scripts
- SimSpace Weaver console
- AWS CLI

Create a snapshot

You can call the `CreateSnapshot` API to create a snapshot for a simulation. The simulation state must be `STARTED`. SimSpace Weaver creates the snapshot file in a `snapshot` folder in the Amazon S3 bucket and object prefix that you specify. For more information, see CreateSnapshot in the *AWS SimSpace Weaver API Reference*.

Start a simulation from a snapshot

You can provide a snapshot when you call the `StartSimulation` API to start a new simulation. You provide a JSON string as an argument to the `SnapshotS3Location` parameter. The string specifies the Amazon S3 bucket name and object key of the snapshot file. If you provide a `SnapshotS3Location`, then you can't provide a `SchemaS3Location`. For more information, see StartSimulation in the *AWS SimSpace Weaver API Reference*.

Frequently asked questions about snapshots

Does my simulation continue to run during a snapshot?

Your simulation resources continue to run during a snapshot and you continue to receive billing charges for that time. The time counts towards your simulation's maximum duration. Your apps don't receive ticks while the snapshot is in progress. If your clock status was `STARTED` when the snapshot creation started, your clock will still indicate `STARTED` status. Your apps receive ticks again after the snapshot finishes. If your clock status was `STOPPED` then your clock status will remain `STOPPED`. Note that a simulation with a `STARTED` status is running even if its clock status is `STOPPED`.

What happens if a snapshot is in progress and my simulation reaches its maximum duration?

Your simulation will finish the snapshot and then stop as soon as the snapshot process ends (either successfully or unsuccessfully). We recommend that you test the snapshot process beforehand to find out how long it takes, the size of the snapshot file you can expect, and if it should complete successfully.

What happens if I stop a simulation that has a snapshot in progress?

A snapshot in progress stops immediately when you stop the simulation. It won't create a snapshot file.

How can I stop a snapshot in progress?

The only way to stop a snapshot in progress is to the stop the simulation. **You can't restart a simulation after you stop it.**

How long will it take to complete my snapshot?

The time required to create a snapshot depends on your simulation. We recommend that you test the snapshot process beforehand to find out how long it will take for your simulation.

How large will my snapshot file be?

The size of a snapshot file depends on your simulation. We recommend that you test the snapshot process beforehand to find out how large the file could be for your simulation.

Best practices when working with SimSpace Weaver

We recommend the following best practices when working with SimSpace Weaver.

Topics

Set up billing alarms

It's easy to provision resources in AWS and leave them running all the time, even when they aren't needed anymore. This can result in runaway costs that can be a surprise when you get your bill. You can configure an alarm in Amazon CloudWatch that will trigger and notify you when your costs exceed a threshold that you set. You can examine your costs using cost management tools. For more information, see:

- Create a billing alarm to monitor your estimated AWS charges
- What is AWS Cost Management

Use SimSpace Weaver Local

We recommend that you use SimSpace Weaver Local to develop and test your simulations before uploading them to the SimSpace Weaver service in the AWS Cloud. The benefits of developing with SimSpace Weaver Local include:

- No need to wait for large uploads
- No limit on the number of local simulations you can create
- You aren't charged for the compute time on your local computer
- Direct access to console output from your apps
- Modify, rebuild, and restart your local simulation without having to recreate it in the AWS Cloud

Stop simulations that you don't need

You get billing charges for a simulation while it is running. You must stop a simulation to stop getting charges for it. Running simulations also count towards your quota for the maximum number of simulations. A running simulation that has logging configured can also generate large amounts of logs, which you also get billing charges for. You should stop any simulation that you don't need in order to stop getting additional charges.

> **Important**
>
> Stopping the simulation clock doesn't stop the simulation, the clock just stops publishing ticks to your apps. You can't restart a simulation after you stop it.

Delete resources that you don't need

Each simulation that you create in SimSpace Weaver also creates resources in other AWS services. You can get billing charges for resources and data in these other services. Running and failed simulations count towards your quota for the maximum number of simulations. You should delete unneeded failed simulations so that you can start new simulations. When you delete a simulation, resources for your simulation that exist in other AWS services might not be deleted. For example, any simulation log data in Amazon CloudWatch Logs will remain there until you delete it. You will get billing charges for that log data. You should cleanup all the associated resources for your simulations if you don't need them anymore. For more information, see Step 6: Stop and clean up your simulation in the quick start tutorial.

Have backups

It's a good idea to have backups and backup plans for everything. You shouldn’t assume that just because your data is in AWS that you don't have to back it up. You must create your own system if you need to back up your simulation state. Consider using multiple AWS Regions and having a plan in place to be able to quickly switch your production workload to another AWS Region if you need to. For more information about AWS Regions that support SimSpace Weaver, see SimSpace Weaver endpoints and quotas.

Security in AWS SimSpace Weaver

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to AWS SimSpace Weaver, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using SimSpace Weaver. The following topics show you how to configure SimSpace Weaver to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your SimSpace Weaver resources.

Topics

Data protection in AWS SimSpace Weaver

The AWS shared responsibility model applies to data protection in AWS SimSpace Weaver. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with SimSpace Weaver or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Encryption at rest

Data is considered *at rest* when it is located in non-volatile (persistent) data storage, such as a disk. Data located in volatile data storage, such as memory and registers, is not considered to be *at rest*.

When you use SimSpace Weaver, the only data at rest are:

- Apps and schemas that you upload to Amazon Simple Storage Service (Amazon S3)
- Simulation log data stored in Amazon CloudWatch

Other data that SimSpace Weaver uses internally doesn't persist after you stop your simulation.

To learn how to encrypt your data at rest, see:

- Encrypt your data in Amazon S3
- Encrypt your log data

Encryption in transit

Your connections to the SimSpace Weaver API through the AWS Command Line Interface (AWS CLI), AWS SDK, and SimSpace Weaver app SDK, use TLS encryption with the Signature Version 4 signing process. AWS manages authentication using the IAM-defined access policies for the security credentials you use to connect.

Internally, SimSpace Weaver uses TLS to connect to other AWS services that it uses.

> **Important**
>
> Communications between your apps and their clients don't involve SimSpace Weaver. It's your responsibility to encrypt communications with simulation clients, if required. We recommend that you create a solution to encrypt all data in transit across client connections.

To learn more about AWS services that can support your encryption solutions, see the AWS Security Blog.

Inter-network traffic privacy

SimSpace Weaver compute resources reside within 1 Amazon VPC shared by all SimSpace Weaver customers. All internal SimSpace Weaver service traffic stays within the AWS network and doesn't travel across the internet. Communication between simulation clients and your apps travels across the internet.

Identity and Access Management for AWS SimSpace Weaver

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use SimSpace Weaver resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting AWS SimSpace Weaver identity and access)
- **Service administrator** - determine user access and submit permission requests (see How AWS SimSpace Weaver works with IAM)
- **IAM administrator** - write policies to manage access (see Identity-based policy examples for AWS SimSpace Weaver)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-

based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS SimSpace Weaver works with IAM

Before you use IAM to manage access to SimSpace Weaver, learn what IAM features are available to use with SimSpace Weaver.

IAM features you can use with AWS SimSpace Weaver

IAM feature	SimSpace Weaver support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Principal permissions	Yes
Service roles	Yes
Service-linked roles	No

To get a high-level view of how SimSpace Weaver and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for SimSpace Weaver

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for SimSpace Weaver

To view examples of SimSpace Weaver identity-based policies, see Identity-based policy examples for AWS SimSpace Weaver.

Resource-based policies within SimSpace Weaver

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for SimSpace Weaver

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

To see a list of SimSpace Weaver actions, see Actions defined by AWS SimSpace Weaver in the *Service Authorization Reference*.

Policy actions in SimSpace Weaver use the following prefix before the action:

```
simspaceweaver
```

To specify multiple actions in a single statement, separate them with commas.

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

To view examples of SimSpace Weaver identity-based policies, see Identity-based policy examples for AWS SimSpace Weaver.

Policy resources for SimSpace Weaver

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

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

To see a list of SimSpace Weaver resource types and their ARNs, see Resources defined by AWS SimSpace Weaver in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by AWS SimSpace Weaver.

To view examples of SimSpace Weaver identity-based policies, see Identity-based policy examples for AWS SimSpace Weaver.

Policy condition keys for SimSpace Weaver

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of SimSpace Weaver condition keys, see Condition keys for AWS SimSpace Weaver in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by AWS SimSpace Weaver.

To view examples of SimSpace Weaver identity-based policies, see Identity-based policy examples for AWS SimSpace Weaver.

Access control lists (ACLs) in SimSpace Weaver

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Attribute-based access control (ABAC) with SimSpace Weaver

Supports ABAC (tags in policies): Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and AWS resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with SimSpace Weaver

Supports temporary credentials: Yes

Temporary credentials provide short-term access to AWS resources and are automatically created when you use federation or switch roles. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM and AWS services that work with IAM in the *IAM User Guide*.

Cross-service principal permissions for SimSpace Weaver

Supports forward access sessions (FAS): Yes

Forward access sessions (FAS) use the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. For policy details when making FAS requests, see Forward access sessions.

Service roles for SimSpace Weaver

Supports service roles: Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break SimSpace Weaver functionality. Edit service roles only when SimSpace Weaver provides guidance to do so.

The SimSpace Weaver app SDK scripts use an CloudFormation template to create resources in other AWS services to support your simulation. One of these resources is the *app role* for your simulation. SimSpace Weaver assumes the app role to perform actions in your AWS account on your behalf, such as to write log data to CloudWatch Logs. For more information about the app role, see Permissions that SimSpace Weaver creates for you.

Service-linked roles for SimSpace Weaver

Supports service-linked roles: No

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a Yes in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for AWS SimSpace Weaver

By default, users and roles don't have permission to create or modify SimSpace Weaver resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by SimSpace Weaver, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for AWS SimSpace Weaver in the *Service Authorization Reference*.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete SimSpace Weaver resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies

that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.

- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the SimSpace Weaver console

To access the AWS SimSpace Weaver console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the SimSpace Weaver resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can still use the SimSpace Weaver console, also attach the SimSpace Weaver *ConsoleAccess* or *ReadOnly* AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Allow users to create and run simulations

This example IAM policy provides the basic permissions required to create and run simulations in SimSpace Weaver.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateAndRunSimulations",
            "Effect": "Allow",
            "Action": [
                "simspaceweaver:*",
                "iam:GetRole",
                "iam:ListRoles",
                "iam:CreateRole",
                "iam:DeleteRole",
                "iam:UpdateRole",
  "iam:CreatePolicy",
  "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "iam:GetRolePolicy",
                "iam:DeleteRolePolicy",
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListAllMyBuckets",
                "s3:PutBucketPolicy",
                "s3:CreateBucket",
                "s3:ListBucket",
                "s3:PutEncryptionConfiguration",
                "s3:DeleteBucket",
                "cloudformation:CreateStack",
                "cloudformation:UpdateStack",
                "cloudformation:DescribeStacks"
            ],
            "Resource": "*"
        },
        {
            "Sid": "PassAppRoleToSimSpaceWeaver",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
```

```
                "StringEquals": {
                    "iam:PassedToService": "simspaceweaver.amazonaws.com"
                }
            }
        }
    ]
}
```

Permissions that SimSpace Weaver creates for you

When you create a SimSpace Weaver project, the service will create an AWS Identity and Access Management (IAM) role with the name `weaver-`*`project-name`*`-app-role` and an IAM trust policy. The trust policy allows SimSpace Weaver to assume the role so that it can perform operations for you.

App role permissions policy

The simulation app role has the following permissions policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:CreateLogGroup",
                "logs:CreateLogStream"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:PutMetricData"
            ],
            "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": "*"
        }
    ]
}
```

App role trust policy

SimSpace Weaver adds a trust relationship to the simulation app role as a trust policy. SimSpace Weaver creates a trust policy for each simulation, similar to the following example.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "simspaceweaver.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn":
     "arn:aws:simspaceweaver:us-west-2:111122223333:simulation/MySimName*"
        }
      }
    }
  ]
}
```

> **Note**
>
> In this example, the account number is 111122223333 and the simulation name is MySimName. These values are different in your trust policies.

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can trick a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it shouldn't otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource policies to limit the permissions that AWS SimSpace Weaver gives another service to the resource. If the `aws:SourceArn` value doesn't contain the account ID, such as an Amazon S3 bucket Amazon Resource Name (ARN), you must use both global condition context keys to limit permissions. If you use both global condition context keys and the `aws:SourceArn` value contains the account ID, the `aws:SourceAccount` value and the account in the `aws:SourceArn` value must use the same account ID when used in the same policy statement. Use `aws:SourceArn` if you want only one resource to be associated with the cross-service access. Use `aws:SourceAccount` if you want to allow any resource in that account to be associated with the cross-service use.

The value of `aws:SourceArn` must use the extension's ARN.

The most effective way to protect against the confused deputy problem is to use the `aws:SourceArn` global condition context key with the full ARN of the resource. If you don't know the full ARN of the extension or if you are specifying multiple extensions, use the `aws:SourceArn` global context condition key with wildcards (*) for the unknown portions of the ARN. For example, `arn:aws:`*`simspaceweaver`*`:*:`*`111122223333`*`:*`.

The following example shows how you can use the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in SimSpace Weaver to prevent the confused deputy problem. This policy will only permit SimSpace Weaver to assume the role when the request comes from the specified source account, and provided with the specified ARN. In this case, SimSpace Weaver can only assume the role for requests from simulations in the requestor's own account (*111122223333*), and only in the specified Region (*us-west-2*).

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "simspaceweaver.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:simspaceweaver:us-
west-2:111122223333:simulation/*"
        }
      }
    }
  ]
}
```

A more secure way to write this policy is to include the simulation name in the `aws:SourceArn`, as shown in the following example, which restricts the policy to a simulation named `MyProjectSimulation_22-10-04_22_10_15`:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "simspaceweaver.amazonaws.com"
```

```
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        },
        "StringLike": {
          "aws:SourceArn": "arn:aws:simspaceweaver:us-
west-2:111122223333:simulation/MySimulation"
        }
      }
    }
  ]
}
```

When your `aws:SourceArn` explicitly includes an account number, you can leave out the `Condition` element test for the `aws:SourceAccount` (see the IAM User Guide for more information), such as in the following simplified policy:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "simspaceweaver.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringLike": {
          "aws:SourceArn": "arn:aws:simspaceweaver:us-
west-2:111122223333:simulation/MySimulation"
        }
      }
    }
  ]
```

```
}
```

Troubleshooting AWS SimSpace Weaver identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with SimSpace Weaver and IAM.

Topics

I am not authorized to perform an action in SimSpace Weaver

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but does not have the fictional `simspaceweaver:`*`GetWidget`* permissions.

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

In this case, Mateo asks his administrator to update his policies to allow him to access the *`my-example-widget`* resource using the `simspaceweaver:`*`GetWidget`* action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to SimSpace Weaver.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in SimSpace Weaver. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

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

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

> **Important**
>
> Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your AWS account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of

two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing access keys in the *IAM User Guide*.

I'm an administrator and want to allow others to access SimSpace Weaver

To allow others to access SimSpace Weaver, you must grant permission to the people or applications that need access. If you are using AWS IAM Identity Center to manage people and applications, you assign permission sets to users or groups to define their level of access. Permission sets automatically create and assign IAM policies to IAM roles that are associated with the person or application. For more information, see Permission sets in the *AWS IAM Identity Center User Guide*.

If you are not using IAM Identity Center, you must create IAM entities (users or roles) for the people or applications that need access. You must then attach a policy to the entity that grants them the correct permissions in SimSpace Weaver. After the permissions are granted, provide the credentials to the user or application developer. They will use those credentials to access AWS. To learn more about creating IAM users, groups, policies, and permissions, see IAM Identities and Policies and permissions in IAM in the *IAM User Guide*.

I want to allow people outside of my AWS account to access my SimSpace Weaver resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether SimSpace Weaver supports these features, see How AWS SimSpace Weaver works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.

- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Security event logging and monitoring in AWS SimSpace Weaver

Monitoring is an important part of maintaining the reliability, availability, and performance of SimSpace Weaver and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs.

AWS and SimSpace Weaver provide several tools for monitoring your simulation resources and responding to potential incidents.

Logs in Amazon CloudWatch

SimSpace Weaver stores its logs in CloudWatch. You can use these logs to monitor events in your simulation (such as starting and stopping apps) as well as for debugging. For more information, see SimSpace Weaver logs in Amazon CloudWatch Logs.

Amazon CloudWatch Alarms

Using Amazon CloudWatch alarms, you watch a single metric over a time period that you specify. If the metric exceeds a given threshold, a notification is sent to an Amazon SNS topic or AWS Auto Scaling policy. CloudWatch alarms are triggered when their state changes and is maintained for a specified number of periods, not by being in a particular state. For more information, see Monitoring SimSpace Weaver with Amazon CloudWatch.

AWS CloudTrail Logs

CloudTrail provides a record of actions taken by a user, role, or an AWS service in SimSpace Weaver. Using the information collected by CloudTrail, you can determine the request that was made to SimSpace Weaver, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see Logging AWS SimSpace Weaver API calls using AWS CloudTrail.

Compliance Validation for AWS SimSpace Weaver

SimSpace Weaver is not in scope of any AWS compliance programs.

Third-party auditors assess the security and compliance of other AWS services as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others.

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. For more information about your compliance responsibility when using AWS services, see AWS Security Documentation.

Resilience in AWS SimSpace Weaver

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure Security in AWS SimSpace Weaver

As a managed service, AWS SimSpace Weaver is protected by AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access SimSpace Weaver through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.

- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

Network connectivity security model

Your simulations run on compute instances within an Amazon VPC located within an AWS Region that you select. An Amazon VPC is a virtual network in the AWS Cloud, which isolates infrastructure by workload or organizational entity. Communications between compute instances within the Amazon VPC stay within the AWS network and don't travel over the internet. Some internal service communication crosses the internet, and is encrypted. Simulations for all customers running in the same AWS Region share the same Amazon VPC. Simulations for different customers use separate compute instances within the same Amazon VPC.

Communications between your simulation clients and your simulations running in SimSpace Weaver travel over the internet. SimSpace Weaver does not handle these connections. It is your responsibility to secure your client connections.

Your connections to the SimSpace Weaver service cross the internet and are encrypted. This includes connections using the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS software development kits (SDK), and the SimSpace Weaver app SDK.

Configuration and vulnerability analysis in AWS SimSpace Weaver

Configuration and IT controls are a shared responsibility between AWS and you. For more information, see the AWS shared responsibility model. AWS handles basic security tasks for the underlying infrastructure, such as patching the operating system on compute instances, firewall configuration, and AWS infrastructure disaster recovery. These procedures have been reviewed and certified by the appropriate third parties. For more details, see Best Practices for Security, Identity, and Compliance.

You are responsible for the security of your simulation software:

- Maintain your app code, including updates and security patches.
- Authenticate and encrypt communication between your simulation clients and the apps they connect to.

- Update your simulations to use the latest SDK versions, including the AWS SDK and SimSpace Weaver app SDK.

> **Note**
>
> SimSpace Weaver doesn't support updates to apps in a running simulation. If you need to update your apps, you must stop and delete the simulation, then create a new simulation with the updated app code. We recommend that you save the simulation state in an external data store so that you can restore it if you need to recreate the simulation.

Security best practices for SimSpace Weaver

This section describes security best practices that are specific to SimSpace Weaver. To learn more about security best practices in AWS, see Best Practices for Security, Identity, and Compliance.

Topics

Encrypt communications between your apps and their clients

SimSpace Weaver doesn't manage communications between your apps and their clients. You should implement some form of authentication and encryption for client sessions.

Periodically backup your simulation state

SimSpace Weaver doesn't save your simulation state. Simulations that are stopped (as a result of an API call, console option, or system crash) do not save their state and have no inherent way to recover them. Stopped simulations cannot be restarted. The only way to perform the equivalent of a restart is to recreate your simulation using the same configuration and data. You can use backups of your simulation state to initialize the new simulation. AWS offers highly reliable and available cloud storage and database services that you can use to save your simulation state.

Maintain your apps and SDKs

Maintain your apps, local installations of the AWS software development kits (SDKs), and the SimSpace Weaver app SDK. You can download and install new versions of the AWS SDKs. Test new versions of the SimSpace Weaver app SDK with non-production app builds to ensure that your apps continue to run as expected. You cannot update your apps in a running simulation. To update your apps:

1. Update and test the app code locally (or in a test environment).
2. Stop changing your simulation state and save it (if necessary).
3. Stop your simulation (once stopped, it cannot be restarted).
4. Delete your simulation (stopped simulations that aren't deleted count towards your service limits).
5. Recreate your simulation with the same configuration and the updated app code.
6. Initialize your simulation using saved state data (if available).
7. Start your new simulation.

> **Note**
>
> A new simulation created with the same configuration is separate from the old simulation. It will have a new simulation ID and send logs to a new log stream in Amazon CloudWatch.

Logging and monitoring in SimSpace Weaver

Monitoring is an important part of maintaining the reliability, availability, and performance of SimSpace Weaver and your other AWS solutions. AWS provides the following monitoring tools to watch SimSpace Weaver, report when something is wrong, and take automatic actions when appropriate:

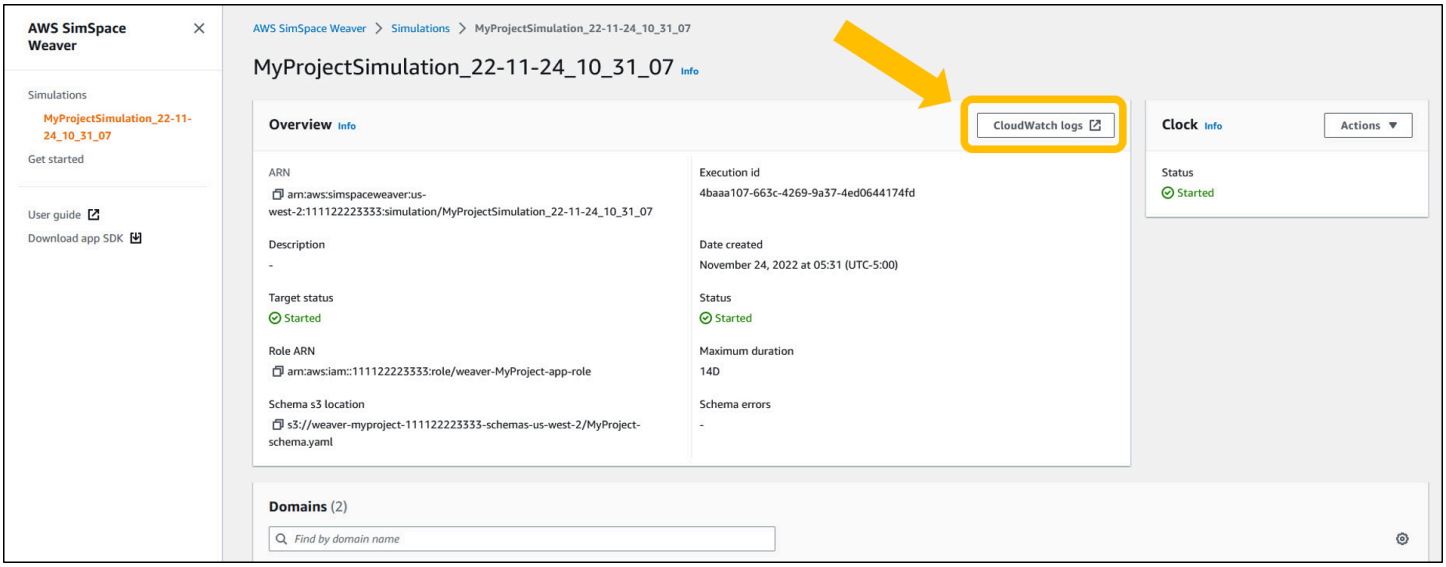
- *Amazon CloudWatch* monitors your AWS resources and the applications you run on AWS in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For more information, see the Amazon CloudWatch User Guide.
- *Amazon CloudWatch Logs* enables you to monitor, store, and access your log data from your SimSpace Weaver workers, CloudTrail, and other sources. CloudWatch Logs can monitor information in the log data and notify you when certain thresholds are met. You can also archive your log data in highly durable storage. For more information, see the Amazon CloudWatch Logs User Guide.
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.

Topics

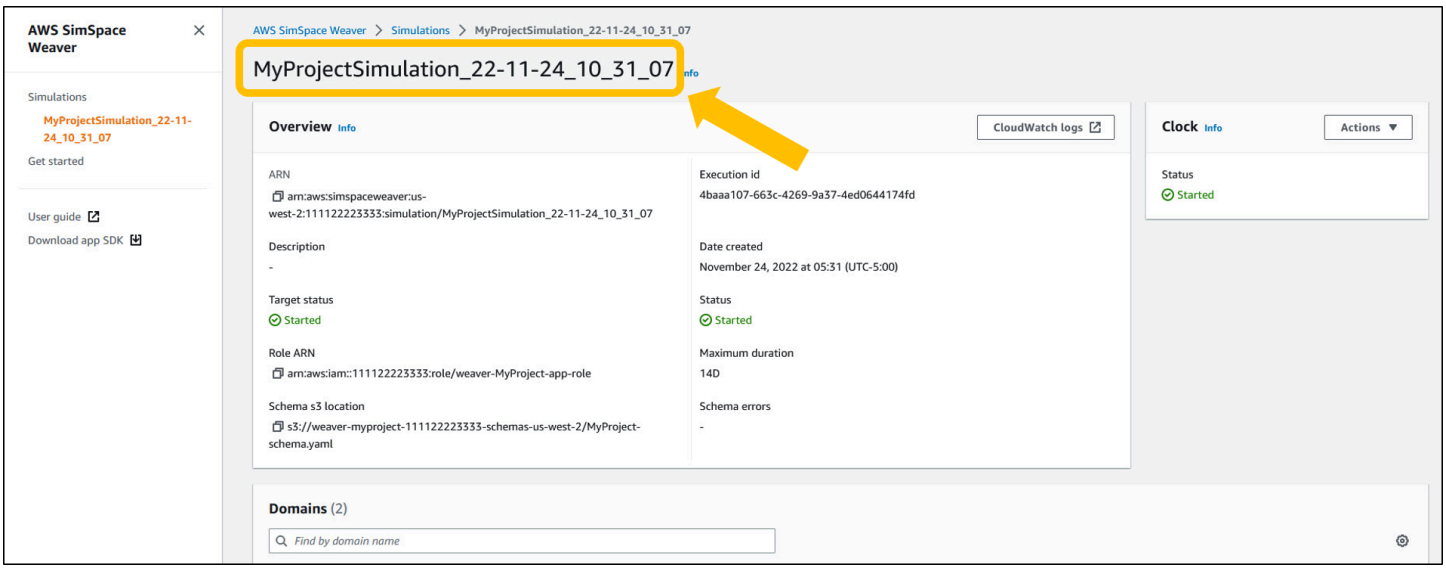
SimSpace Weaver logs in Amazon CloudWatch Logs

Accessing your SimSpace Weaver logs

All the logs generated from your SimSpace Weaver simulations are stored in Amazon CloudWatch Logs. To access your logs, you can use the **CloudWatch logs** button in the **Overview** pane of your simulation in the SimSpace Weaver console, which will take you directly to the logs for that specific simulation.



You can also access the logs through the CloudWatch console. You will need the name of your simulation in order to search for its logs.



SimSpace Weaver logs

SimSpace Weaver writes simulation management messages and the console output from your apps to Amazon CloudWatch Logs. For more information on working with logs, see Working with log groups and log streams in the *Amazon CloudWatch Logs User Guide*.

Each simulation that you create has its own log group in CloudWatch Logs. The name of the log group is specified in the simulation schema. In the following schema snippet, the value of `log_destination_service` is `logs`. This means that the value of

`log_destination_resource_name` is the name of a log group. In this case, the log group is `MySimulationLogs`.

```
simulation_properties:
  log_destination_service: "logs"
  log_destination_resource_name: "MySimulationLogs"
  default_entity_index_key_type: "Vector3<f32>"
```

You can also use the **DescribeSimulation** API to find the name of the log group for simulation after you start it.

> **Important**
>
> If you use AWS IAM Identity Center or named profiles for the AWS Command Line Interface (AWS CLI), you must use SimSpace Weaver app SDK version 1.12.1 or higher. The latest version is 1.15.3. For information about SimSpace Weaver versions, see SimSpace Weaver versions. The SimSpace Weaver app SDK scripts use the AWS CLI. If you use IAM Identity Center, you can either copy your IAM Identity Center profile for the AWS CLI to your `default` profile or provide the name of your IAM Identity Center profile to SimSpace Weaver app SDK scripts with the `--profile` *`cli-profile-name`* parameter. For more information, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide* and Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

Docker

```
project-folder\tools\windows\weaver-project-name-cli.bat describe-simulation --
simulation simulation-name
```

WSL

> **Important**
>
> We provide these instructions for your convenience. They are for use with Windows Subsystem for Linux (WSL), and are unsupported. For more information, see Set up your local environment for SimSpace Weaver.

```
project-folder/tools/linux/weaver-project-name-cli.sh describe-simulation --
simulation simulation-name
```

The following example shows the part of the output from **DescribeSimulation** that describes the logging configuration. The name of the log group is shown at the end of the `LogGroupArn`.

```
    "LoggingConfiguration": {
        "Destinations": [
            {
                "CloudWatchLogsLogGroup": {
                    "LogGroupArn": "arn:aws:logs:us-west-2:111122223333:log-
group:MySimulationLogs"
                }
            }
        ]
    },
```

Each simulation log group contains several log streams:

- **Management log stream** – simulation management messages produced by the SimSpace Weaver service.

  ```
  /sim/management
  ```

- **Errors log stream** – error messages produced by the SimSpace Weaver service. This log stream only exists if there are errors. SimSpace Weaver stores errors written by your apps in their own app log streams (see the following log streams).

```
/sim/errors
```

- **Spatial app log streams** (1 for each spatial app on each worker) – console output produced by spatial apps. Each spatial app writes to its own log stream. The *spatial-app-id* is all characters after the trailing slash at the end of the *worker-id*.

```
/domain/spatial-domain-name/app/worker-worker-id/spatial-app-id
```

- **Custom app log streams** (1 for each custom app instance) – console output produced by custom apps. Each custom app instance writes to its own log stream.

```
/domain/custom-domain-name/app/custom-app-name/random-id
```

- **Service app log streams** (1 for each service app instance) – console output produced by service apps. Each service app writes to its own log stream. The *service-app-id* is all characters after the trailing slash at the end of the *service-app-name*.

```
/domain/service-domain-name/app/service-app-name/service-app-id
```

Monitoring SimSpace Weaver with Amazon CloudWatch

You can monitor SimSpace Weaver using Amazon CloudWatch, which collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the Amazon CloudWatch User Guide.

The SimSpace Weaver service reports the following metrics in the `AWS/simspaceweaver` namespace.

SimSpace Weaver metrics at the account level

The SimSpace Weaver namespace includes the following metrics related to activity at the AWS account level.

Metric	Description
`SimulationCount`	The number of simulations for the current account. Units: Count Dimensions: none Statistics: Average, Minimum, Maximum

Logging AWS SimSpace Weaver API calls using AWS CloudTrail

AWS SimSpace Weaver is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in SimSpace Weaver. CloudTrail captures all API calls for SimSpace Weaver as events. The calls captured include calls from the SimSpace Weaver console and code calls to the SimSpace Weaver API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for SimSpace Weaver. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to SimSpace Weaver, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

SimSpace Weaver information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in SimSpace Weaver, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing events with CloudTrail Event history.

For an ongoing record of events in your AWS account, including events for SimSpace Weaver, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

All SimSpace Weaver actions are logged by CloudTrail and are documented in the AWS SimSpace Weaver API Reference. For example, calls to the `ListSimulations`, `DescribeSimulation` and `DeleteSimulation` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

Understanding SimSpace Weaver log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, such as the date and time of the action, request parameters, and other details. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `ListSimulations` action.


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:aws-console-signin-utils",
```

```
        "arn": "arn:aws:sts::111122223333:assumed-role/ConsoleSigninRole/aws-console-
signin-utils",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/ConsoleSigninRole",
                "accountId": "111122223333",
                "userName": "ConsoleSigninRole"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-02-14T15:57:02Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-02-14T15:57:08Z",
    "eventSource": "simspaceweaver.amazonaws.com",
    "eventName": "ListSimulations",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.10",
    "userAgent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
 Gecko) Chrome/86.0.4240.0 Safari/537.36",
    "requestParameters": null,
    "responseElements": null,
    "requestID": "1234abcd-1234-5678-abcd-12345abcd123",
    "eventID": "5678abcd-5678-1234-ab12-123abc123abc",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

SimSpace Weaver endpoints and quotas

The following tables describe the service endpoints and service quotas for SimSpace Weaver. Service quotas, also referred to as *limits*, are the maximum number of service resources or operations for your AWS account. For more information, see AWS service quotas in the *AWS General Reference*.

Service endpoints

Region name	Region	Endpoint	Protocol
US East (N. Virginia)	us-east-1	simspaceweaver.us-east-1.amazonaws.com	HTTPS
US East (Ohio)	us-east-2	simspaceweaver.us-east-2.amazonaws.com	HTTPS
US West (Oregon)	us-west-2	simspaceweaver.us-west-2.amazonaws.com	HTTPS
Asia Pacific (Singapore)	ap-southeast-1	simspaceweaver.ap-southeast-1.amazonaws.com	HTTPS
Asia Pacific (Sydney)	ap-southeast-2	simspaceweaver.ap-southeast-2.amazonaws.com	HTTPS
Europe (Stockholm)	eu-north-1	simspaceweaver.eu-north-1.amazonaws.com	HTTPS

Region name	Region	Endpoint	Protocol
Europe (Frankfurt)	eu-central-1	simspaceweaver.eu-central-1.amazonaws.com	HTTPS
Europe (Ireland)	eu-west-1	simspaceweaver.eu-west-1.amazonaws.com	HTTPS
AWS GovCloud (US-East)	us-gov-east-1	simspaceweaver.us-gov-east-1.amazonaws.com	HTTPS
AWS GovCloud (US-West)	us-gov-west-1	simspaceweaver.us-gov-west-1.amazonaws.com	HTTPS

Service quotas

Name	Default	Adjuse	Description
Compute resource units for each app	Each supported Region: 4	No	The maximum number of compute resource units that you can allocate to each app.
Compute resource units for each worker	Each supported Region: 17	No	The number of compute resource units available for each worker.
Data fields for each entity	Each supported Region: 7	No	The maximum number of data (non-index) fields that an entity can have.

Name	Default	Adjustable	Description
Entities in a partition	Each supported Region: 8,192	No	The maximum number of entities in 1 partition.
Entity data field size	Each supported Region: 1,024 Bytes	No	The maximum size of a data (non-index) field of an entity.
Entity transfers between workers	Each supported Region: 25	No	The maximum number of entity transfers between workers, for each partition and each tick.
Entity transfers on the same worker	Each supported Region: 500	No	The maximum number of entity transfers on the same worker, for each partition and each tick.
Index fields for each entity	Each supported Region: 1	No	The maximum number of index fields that an entity can have.
Largest maximum duration (in days) for a simulation	Each supported Region: 14	No	The largest number of days that you can specify as the maximum duration for a simulation. All simulations have a maximum duration, even if you dont specify the value. A simulation automatically stops when it reaches its maximum duration.

Name	Default	Adjuse	Description
Memory for each compute resource unit	Each supported Region: 1 Gigabytes	No	The amount of random-access memory (RAM) that an app gets for each compute resource unit.
Remote subscriptions for each worker	Each supported Region: 24	No	The maximum number of remote subscriptions for each worker.
Simulation count	Each supported Region: 2	Yes	The maximum number of simulations with a target status of STARTED in your account. You can request a quota increase up to 10.
Workers for a simulation	Each supported Region: 2	Yes	The maximum number of workers that you can assign to 1 simulation. You can request a quota increase up to 10.
vCPUs for each compute resource unit	Each supported Region: 2	No	The number of virtual central processing units (vCPUs) that an app gets for each compute resource unit.

Clock rates

The simulation schema specifies the *clock rate* (also called the *tick rate*) for a simulation. The following table specifies the valid clock rates that you can use.

Name	Valid values	Description
Clock rate	Each supported region: "10", "15", "30", "unlimited"	The valid clock rates for a simulation.
Clock rate (versions 1.13 and 1.12)	Each supported region: 10, 15, 30	The valid clock rates for a simulation.

Service quotas for SimSpace Weaver Local

The following service quotas apply to SimSpace Weaver Local only. All other quotas also apply to SimSpace Weaver Local.

Name	Default	Adjustable	Description
Maximum partitions	SimSpace Weaver Local: 24	No	The maximum number of partitions for a simulation.
Maximum apps	SimSpace Weaver Local: 24	No	The maximum total number of apps (of any type) for a simulation.
Maximum domains	SimSpace Weaver Local: 24	No	The maximum total number of domains (of any type) for a simulation.
Entities per partition	SimSpace Weaver Local: 4,096	No	The maximum number of entities in each partition.
Fields per entity	SimSpace Weaver Local: 8	No	The maximum number of fields for each entity.

Name	Default	Adjustable	Description
Field size	SimSpace Weaver Local: 1024 bytes	No	The maximum size of an entity field.

Troubleshooting in SimSpace Weaver

Topics

AssumeRoleAccessDenied

You might receive the following error if your simulation fails to start:

```
Unable to assume role arn:aws:iam::111122223333:role/weaver-project-name-app-role;
 verify the role exists and has trust policy on SimSpace Weaver
```

You can receive this error if one of the following is true about the AWS Identity and Access Management (IAM) role for your simulation:

- The Amazon Resource Name (ARN) refers to an IAM role that doesn't exist.
- The trust policy for the IAM role that doesn't allow the name of the new simulation to assume the role.

Check to make sure that the role exists. If the role exists, check your trust policy for the role. The `aws:SourceArn` in following example trust policy only allows a simulation (in account 111122223333) whose name begins with `MySimulation` to assume the role.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "simspaceweaver.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:simspaceweaver:us-
west-2:111122223333:simulation/MySimulation*"
                }
            }
        }
    ]
}
```

To allow another simulation whose name begins with `MyOtherSimulation` to assume the role, the trust policy must be modified as in the following edited example:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "simspaceweaver.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": [
                        "arn:aws:simspaceweaver:us-
west-2:111122223333:simulation/MySimulation*",
                        "arn:aws:simspaceweaver:us-
west-2:111122223333:simulation/MyOtherSimulation*"
                        ]
                }
```

```
            }
        }
    ]
}
```

InvalidBucketName

You might receive the following error while creating a project:

```
An error occurred (InvalidBucketName) when calling the CreateBucket operation: The
 specified bucket is not valid.
```

You received this error because the name that SimSpace Weaver passed to Amazon Simple Storage Service (Amazon S3) violated bucket naming rules (for more information, see Bucket naming rules in the *Amazon Simple Storage Service User Guide*).

The `create-project` script in the SimSpace Weaver app SDK creates bucket names using the project name that you provide to the script. The bucket names use the following formats:

- Version 1.13.x or later
  - `weaver-`*`lowercase-project-name`*`-`*`account-number`*`-`*`region`*
- Version 1.12.x
  - `weaver-`*`lowercase-project-name`*`-`*`account-number`*`-app-zips-`*`region`*
  - `weaver-`*`lowercase-project-name`*`-`*`account-number`*`-schemas-`*`region`*

For example, given the following project properties:

- Project name: `MyProject`
- AWS account number: `111122223333`
- AWS Region: `us-west-2`

The project would have the following buckets:

- Version 1.13.x or later
  - `weaver-myproject-111122223333-us-west-2`

- Version 1.12.x
  - `weaver-myproject-111122223333-app-zips-us-west-2`
  - `weaver-myproject-111122223333-schemas-us-west-2`

Your project name must not violate the Amazon S3 naming rules. You must also use a project name that is short enough so that the bucket names created by the `create-project` script do not exceed the name length limit for Amazon S3 buckets.

ServiceQuotaExceededException

You might receive the following error when you start a simulation:

```
An error occurred (ServiceQuotaExceededException) when calling the StartSimulation
 operation: Failed to start simulation due to: simulation quota has already been
 reached.
```

You will receive this error if you try to start a new simulation but your account currently has the maximum number of simulations with a target status of STARTED. This includes running simulations, failed simulations, and simulations that stopped because they reached their maximum duration. You can delete a stopped or failed simulation to allow you to start a new simulation. If all of your simulations are running, you can stop and delete a running simulation. You can also request an increase to your service quotas if you aren't already at the request limit. For more information, see SimSpace Weaver endpoints and quotas. For information about how to clean up unnecessary simulations, see Step 6: Stop and clean up your simulation in the quick start tutorial.

TooManyBuckets

You might receive the following error while creating a project:

```
An error occurred (TooManyBuckets) when calling the CreateBucket operation: You have
 attempted to create more buckets than allowed.
```

Amazon Simple Storage Service (Amazon S3) has a limit on the number of buckets that you can have in your AWS account (for more information, see Bucket restrictions and limitations in the *Amazon Simple Storage Service User Guide*).

You must do one of the following before you can continue:

- Delete 2 or more existing Amazon S3 buckets that you don't need.
- Request an Amazon S3 limit increase (for more information, see Bucket restrictions and limitations in the *Amazon Simple Storage Service User Guide*).
- Use a different AWS account.

> **Note**
>
> The `DeleteSimulation` API in SimSpace Weaver doesn't delete Amazon S3 resources associated with your simulation. We recommend that you remove all resources associated with your simulations when you no longer need them. For guidance on how to clean up your simulation, see Step 6: Stop and clean up your simulation in the quick start tutorial.

Permission denied during simulation start

When you start a simulation, you might get an error message indicating that permission was denied or that there was an error accessing your app artifacts. This problem can occur when you specify Amazon S3 buckets for your simulation that SimSpace Weaver didn't create for you (either through the console or the SimSpace Weaver app SDK scripts).

The following situations are the most likely root causes:

- **The service doesn't have permission to access one or more of the Amazon S3 buckets you specified in your simulation schema** – check your app role permissions policy, Amazon S3 bucket policies, and Amazon S3 bucket permissions to make sure that `simspaceweaver.amazonaws.com` has the correct permissions to access your buckets. For more information about the app role permissions policy, see Permissions that SimSpace Weaver creates for you.
- **Your Amazon S3 bucket could be in a different AWS Region than your simulation** – Your Amazon S3 buckets for your simulation artifacts must be in the same AWS Region as your simulation. Check your Amazon S3 console to see what AWS Region your bucket is in. If your Amazon S3 bucket is in a different AWS Region, select a bucket that is in the same AWS Region as your simulation.

Problems related to time when using Docker

If you are using Docker and you receive time-related errors while running scripts from the SimSpace Weaver app SDK, the cause could be that your Docker virtual machine clock is incorrect. This can happen if your computer was running Docker and then resumes from sleep or hibernation.

Solutions to try

- Restart Docker.
- Disable and then re-enable time synchronization in **Windows PowerShell**:

```
Get-VMIntegrationService -VMName DockerDesktopVM -Name "Time Synchronization" |
 Disable-VMIntegrationService
Get-VMIntegrationService -VMName DockerDesktopVM -Name "Time Synchronization" |
 Enable-VMIntegrationService
```

PathfindingSample console client fails to connect

You might get the following error from the console client when you connect to the `PathfindingSample` simulation described in the quick start tutorial and detailed tutorial. This error occurs because the client can't open a network connection to the `ViewApp` at the combined IP address and port number that you provided.

```
Fatal error in function nng_dial. Error code: 268435577. Error message: no link
```

For a simulation in the AWS Cloud

- **Is your network connection working correctly?** Verify that you can connect to other IP addresses or web sites that should work. Make sure that your web browser isn't loading a web site from its cache.
- **Is your simulation running?** You can use the **ListSimulations** API to get the status of your simulation. For more information, see Step 4: Get your IP address and port number. You can also use the SimSpace Weaver console to check the status of your simulations.
- **Are your apps running?** You can use the **DescribeApp** API to get the status of your apps. For more information, see Step 4: Get your IP address and port number. You can also use the SimSpace Weaver console to check the status of your simulations.

- **Are your apps running?** You can use the **DescribeApp** API to get the status of your apps. For more information, see Step 4: Get your IP address and port number. You can also use the SimSpace Weaver console to check the status of your simulations.
- **Did you use the correct IP address and port number?** When you connect over the internet, you must use the IP address and `Actual` port number of the `ViewApp`. You can find the IP `Address` and `Actual` port number in the `EndpointInfo` block of the **DescribeApp** API output. You can also use the SimSpace Weaver console to find the IP address (**URI**) and port number (**Ingress port**) for your `ViewApp` in the `MyViewDomain` detail page.
- **Is your network connection going through a firewall?** Your firewall might block your connection to the IP address or port number (or both). Check your firewall settings or check with your firewall administrator.

For a local simulation

- **Can you connect to your loopback address (127.0.0.1)?** If you have the `ping` command line tool in Windows, you can open a command prompt window and try to ping 127.0.0.1. Press **Ctrl-C** to end the ping.

```
ping 127.0.0.1
```

Example ping output

```
C:\>ping 127.0.0.1

Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time<1ms TTL=128
Reply from 127.0.0.1: bytes=32 time=1ms TTL=128

Ping statistics for 127.0.0.1:
    Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
Control-C
^C
C:\>
```

If the ping says it lost packets, you might have other software (such as a local firewall, security settings, or anti-malware programs) that is blocking your connection.

- **Are your apps running?** Your local simulation runs as separate windows for each app. Make sure the windows for your spatial apps and `ViewApp` are open. For more information, see Local development.
- **Did you use the correct IP address and port number?** You must use `tcp://127.0.0.1:7000` when you connect to a local simulation. For more information, see Local development.
- **Do you have local security software that could block your connection?** Check your security settings, local firewall, or anti-malware programs to see if they are blocking your connection to `127.0.0.1` on TCP port `7000`.

The AWS CLI doesn't recognize `simspaceweaver`

If the AWS CLI gives you errors that suggest that it doesn't know about SimSpace Weaver, run the following command.

```
aws simspaceweaver help
```

If you get an error that starts with the following lines and lists all available choices then your AWS CLI might be an older version.

```
usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

  aws help
  aws <command> help
  aws <command> <subcommand> help

aws: error: argument command: Invalid choice, valid choices are:
```

Run the following command to check the version of your AWS CLI.

```
aws --version
```

If the version number is earlier than 2.9.19 then you must update your AWS CLI. Note that the current version of the AWS CLI is later than 2.9.19.

To update your AWS CLI, see Install or update the latest version of the AWS CLI in the *AWS Command Line Interface User Guide for Version 2*.

SimSpace Weaver simulation schema reference

SimSpace Weaver uses a YAML file to configure the properties of a simulation. This file is called the *simulation schema* (or just *schema*). The sample simulation included in the SimSpace Weaver app SDK includes a schema that you can copy and edit for your own simulation.

Topics

Examples of a complete schema

The following examples show the YAML-formatted text file that describes a SimSpace Weaver simulation. These examples include dummy values for the properties. The format of the file varies based on the value of `sdk_version` specified in it. See Schema format for a complete description of the properties and their valid values.

1.15

```
sdk_version: "1.15"
simulation_properties:
  log_destination_resource_name: "MySimulationLogs"
  log_destination_service: "logs"
  default_entity_index_key_type: "Vector3<f32>"
  default_image: "111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-
repository:latest"
workers:
  MyComputeWorkers:
    type: "sim.c5.24xlarge"
    desired: 3
clock:
  tick_rate: "30"
partitioning_strategies:
  MyGridPartitioning:
    topology: "Grid"
    aabb_bounds:
      x: [-1000, 1000]
      y: [-1000, 1000]
    grid_placement_groups:
```

```
      x: 3
      y: 3
domains:
  MyCustomDomain:
    launch_apps_via_start_app_call: {}
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MyViewApp.zip"
      launch_command: ["MyViewApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports: [9000, 9001]
  MyServiceDomain:
    launch_apps_per_worker:
      count: 1
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/
MyConnectionServiceApp.zip"
      launch_command: ["MyConnectionServiceApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 9000
          - 9001
  MySpatialDomain:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 6
        y: 6
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MySpatialApp.zip"
      launch_command: ["MySpatialApp"]
      required_resource_units:
        compute: 1
  MySpatialDomainWithCustomContainer:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 6
        y: 6
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MySpatialApp2.zip"
```

```
      launch_command: ["MySpatialApp2"]
      required_resource_units:
        compute: 1
      image: "111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest"
placement_constraints:
  - placed_together: ["MySpatialDomain", "MySpatialDomainWithCustomContainer"]
    on_workers: ["MyComputeWorkers"]
```

1.14

```
sdk_version: "1.14"
simulation_properties:
  log_destination_resource_name: "MySimulationLogs"
  log_destination_service: "logs"
  default_entity_index_key_type: "Vector3<f32>"
  default_image: "111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-
repository:latest"
workers:
  MyComputeWorkers:
    type: "sim.c5.24xlarge"
    desired: 3
clock:
  tick_rate: "30"
partitioning_strategies:
  MyGridPartitioning:
    topology: "Grid"
    aabb_bounds:
      x: [-1000, 1000]
      y: [-1000, 1000]
    grid_placement_groups:
      x: 3
      y: 3
domains:
  MyCustomDomain:
    launch_apps_via_start_app_call: {}
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MyViewApp.zip"
      launch_command: ["MyViewApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports: [9000, 9001]
  MyServiceDomain:
```

```
    launch_apps_per_worker:
      count: 1
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/
MyConnectionServiceApp.zip"
      launch_command: ["MyConnectionServiceApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 9000
          - 9001
  MySpatialDomain:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 6
        y: 6
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MySpatialApp.zip"
      launch_command: ["MySpatialApp"]
      required_resource_units:
        compute: 1
  MySpatialDomainWithCustomContainer:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 6
        y: 6
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MySpatialApp2.zip"
      launch_command: ["MySpatialApp2"]
      required_resource_units:
        compute: 1
      image: "111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest"
placement_constraints:
  - placed_together: ["MySpatialDomain", "MySpatialDomainWithCustomContainer"]
    on_workers: ["MyComputeWorkers"]
```

1.13

```
sdk_version: "1.13"
simulation_properties:
```

```
  log_destination_resource_name: "MySimulationLogs"
  log_destination_service: "logs"
  default_entity_index_key_type: "Vector3<f32>"
workers:
  MyComputeWorkers:
    type: "sim.c5.24xlarge"
    desired: 3
clock:
  tick_rate: 30
partitioning_strategies:
  MyGridPartitioning:
    topology: "Grid"
    aabb_bounds:
      x: [-1000, 1000]
      y: [-1000, 1000]
    grid_placement_groups:
      x: 3
      y: 3
domains:
  MyCustomDomain:
    launch_apps_via_start_app_call: {}
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MyViewApp.zip"
      launch_command: ["MyViewApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports: [9000, 9001]
  MyServiceDomain:
    launch_apps_per_worker:
      count: 1
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/
MyConnectionServiceApp.zip"
      launch_command: ["MyConnectionServiceApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 9000
          - 9001
  MySpatialDomain:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
```

```
      grid_partition:
        x: 6
        y: 6
    app_config:
      package: "s3://weaver-myproject-111122223333-us-west-2/MySpatialApp.zip"
      launch_command: ["MySpatialApp"]
      required_resource_units:
        compute: 1
```

1.12

```
sdk_version: "1.12"
simulation_properties:
  log_destination_resource_name: "MySimulationLogs"
  log_destination_service: "logs"
  default_entity_index_key_type: "Vector3<f32>"
workers:
  MyComputeWorkers:
    type: "sim.c5.24xlarge"
    desired: 3
clock:
  tick_rate: 30
partitioning_strategies:
  MyGridPartitioning:
    topology: "Grid"
    aabb_bounds:
      x: [-1000, 1000]
      y: [-1000, 1000]
    grid_placement_groups:
      x: 3
      y: 3
domains:
  MyCustomDomain:
    launch_apps_via_start_app_call: {}
    app_config:
      package: "s3://weaver-myproject-111122223333-app-zips-us-west-2/MyViewApp.zip"
      launch_command: ["MyViewApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports: [9000, 9001]
  MyServiceDomain:
    launch_apps_per_worker:
```

```
      count: 1
    app_config:
      package: "s3://weaver-myproject-111122223333-app-zips-us-west-2/
MyConnectionServiceApp.zip"
      launch_command: ["MyConnectionServiceApp"]
      required_resource_units:
        compute: 1
      endpoint_config:
        ingress_ports:
          - 9000
          - 9001
  MySpatialDomain:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "MyGridPartitioning"
      grid_partition:
        x: 6
        y: 6
    app_config:
      package: "s3://weaver-myproject-111122223333-app-zips-us-west-2/
MySpatialApp.zip"
      launch_command: ["MySpatialApp"]
      required_resource_units:
        compute: 1
```

Schema format

The following example shows the overall structure of a schema. The order of properties at each level of the schema doesn't matter, as long as the parent-child relationships are the same. The order matters for elements in an array.

```
sdk_version: "sdk-version-number"
simulation_properties:
  simulation-properties
workers:
  worker-group-configurations
clock:
  tick_rate: tick-rate
partitioning_strategies:
  partitioning-strategy-configurations
domains:
  domain-configurations
placement_constraints:
```

```
  placement-constraints-configuration
```

Sections

SDK version

The `sdk_version` section (required) identifies the version of the SimSpace Weaver app SDK that supports this schema. Valid values: `1.15`, `1.14`, `1.13`, `1.12`

> **Important**
>
> The value of `sdk_version` only includes the major version number and first minor version number. For example, the value `1.12` specifies all versions `1.12.x`, such as `1.12.0`, `1.12.1`, and `1.12.2`.

```
sdk_version: "1.15"
```

Simulation properties

The `simulation_properties` section (required) specifies various properties of your simulation. Use this section to configure logging and specify a default container image. This section is required even if you don't configure logging or choose to specify a default container image.

```
simulation_properties:
  log_destination_resource_name: "log-destination-resource-name"
  log_destination_service: "log-destination-service"
  default_entity_index_key_type: "Vector3<f32>"
  default_image: "ecr-repository-uri"
```

Properties

`log_destination_resource_name`

Specifies the resource that SimSpace Weaver will write logs to.

Required: No. If this property isn't included, SimSpace Weaver won't write logs for the simulation.

Type: String

Valid values:

- The name of an CloudWatch Logs log group (for example, `MySimulationLogs`)
- The Amazon Resource Name (ARN) of a CloudWatch Logs log group (for example, `arn:aws:logs:us-west-2:111122223333:log-group/MySimulationLogs`)

> **Note**
>
> SimSpace Weaver only supports a log destination in the same account and AWS Region as the simulation.

`log_destination_service`

Indicates the type of logging destination resource when you specify a `logging_destination resource_name` that isn't an ARN.

Required: You must specify this property if the `log_destination_resource_name` is specified and isn't an ARN. You can't specify this property if the `log_destination_resource_name` isn't specified or is an ARN.

Type: String

Valid values:

- `logs`: The log destination resource is a log group.

`default_entity_index_key_type`

Specifies the data type for the index key field of simulation entities.

Required: Yes

Type: String

Valid values: `Vector3<f32>`

`default_image`

Specifies the default container image for your simulation (not supported for version `1.13` and `1.12`). If this property is specified, domains that don't specify an `image` use the `default_image`.

Required: No

Type: String

Valid values:

- The URI of a repository in Amazon Elastic Container Registry (Amazon ECR) (for example, `111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest`)

Workers

The `workers` section (required) specifies configurations for *worker groups* (groups of workers). SimSpace Weaver uses this information together with `placement_constraints` to configure the underlying infrastructure of your simulation. Only 1 worker group is currently supported.

To specify the properties for a worker group, replace *`worker-group-name`* with a name of your choice. The name must be 3-64 characters long and can contain the characters **A** -**Z** , **a** -**z** , **0** -**9** , and **_** **-** (hyphen). Specify the properties of the worker group after the name.

```
workers:
  worker-group-name:
    type: "sim.c5.24xlarge"
    desired: number-of-workers
```

Properties

`type`

Specifies worker type.

Required: Yes

Type: String

Valid values: `sim.c5.24xlarge`

`desired`

Specifies the desired number of workers for this worker group.

Required: Yes

Type: Integer

Valid values: 1-3. Your service quota (limit) for the number of workers for your simulations determines the maximum value of this property. For example, if your service quota is 2 then the maximum value for this property is 2. You can request an increase to your service quota. For more information, see SimSpace Weaver endpoints and quotas.

Clock

The `clock` section (required) specifies the properties of the simulation clock.

```
clock:
  tick_rate: tick-rate
```

Properties

`tick_rate`

Specifies the number of ticks per second that the clock publishes to apps.

Required: Yes

Type:

- *Version `1.14` and `1.15`:* String
- *Version `1.13` and `1.12`:* Integer

Valid values:

- *Version `1.14` and `1.15`:* `"10"` | `"15"` | `"30"` | `"unlimited"`
  - `"unlimited"`: the clock sends the next tick as soon as all apps finish their commit operations for the current tick.
- *Version `1.13` and `1.12`:* `10` | `15` | `30`

Partitioning strategies

The `partitioning_strategies` section (required) specifies the organization of partitions for a spatial domain.

> **Note**
>
> SimSpace Weaver only supports 1 partitioning strategy.

To specify the properties for a partitioning strategy, replace *`partitioning-strategy-name`* with a name of your choice. The name must be 3-64 characters long and can contain the characters **A** -**Z** , **a** -**z** , **0** -**9** , and **_** **-** (hyphen). Specify the properties of the partitioning strategy after the name.

```
partitioning_strategies:
  partitioning-strategy-name:
    topology: "Grid"
    aabb_bounds:
      x: [aabb-min-x, aabb-max-x]
      y: [aabb-min-y, aabb-max-y]
    grid_placement_groups:
      x: number-of-placement-groups-along-x-axis
      y: number-of-placement-groups-along-y-axis
```

Properties

`topology`

Specifies the topology (partition arrangement scheme) for this partitioning strategy.

Required: Yes

Type: String

Valid values: `"Grid"`

`aabb_bounds`

Specifies the bounds of the main axis-aligned bounding box (AABB) for your simulation. You specify the bounds as 2-element ordered arrays that describe the minimum and maximum values (in that order) for each axis (x and y).

Required: Conditional. This property is required (and can only be specified) if the topology is set to `"Grid"`.

Type: `Float` array (for each axis)

Valid values: `-3.4028235e38-3.4028235e38`

`grid_placement_groups`

Specifies the number of placement groups along each axis (x and y) in a grid topology. A placement group is a collection of partitions (in the same domain) that are spatially adjacent.

Required: Conditional. This property is required (and can only be specified) if the topology is set to `"Grid"`. If you don't specify a placement groups configuration, SimSpace Weaver will calculate one for you. Any domain that uses a partitioning strategy without a placement groups configuration must specify a `grid_partition` (see Spatial domain partitioning strategy).

Type: Integer (for each axis)

Valid values: `1-20`. We recommend that x * y is equal to the desired number of workers. Otherwise, SimSpace Weaver will attempt to balance your placement groups across the available workers.

Domains

The `domains` section (required) specifies the properties for each of your domains. All simulations must have at least one section for a spatial domain. You can create multiple sections for additional domains. Each type of domain has its own configuration format.

Important

Versions `1.13` and `1.12` don't support multiple spatial domains.

Important

SimSpace Weaver supports up to 5 domains for each simulation. This includes all spatial, custom, and service domains.

```
domains:
  domain-name:
    domain-configuration
  domain-name:
    domain-configuration
  ...
```

Domain configuration

- Spatial domain configuration
- Custom domain configuration
- Service domain configuration

Spatial domain configuration

To specify the properties for a spatial domain, replace *spatial-domain-name* with a name of your choice. The name must be 3-64 characters long and can contain the characters **A** -**Z** , **a** -**z** , **0** -**9** , and **_** **-** (hyphen). Specify the properties of the spatial domain after the name.

```
  spatial-domain-name:
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "partitioning-strategy-name"
      grid_partition:
        x: number-of-partitions-along-x-axis
        y: number-of-partitions-along-y-axis
    app_config:
      package: "app-package-s3-uri"
      launch_command: ["app-launch-command", "parameter1", ...]
      required_resource_units:
        compute: app-resource-units
    image: "ecr-repository-uri"
```

Spatial domain partitioning strategy

The `launch_apps_by_partitioning_strategy` section (required) specifies the partitioning strategy and dimensions (in number of partitions) of the simulation space.

```
    launch_apps_by_partitioning_strategy:
      partitioning_strategy: "partitioning-strategy-name"
      grid_partition:
        x: number-of-partitions-along-x-axis
```

```
        y: number-of-partitions-along-y-axis
```

Properties

`partitioning_strategy`

Specifies the partitioning strategy for this spatial domain.

Required: Yes

Type: String

Valid values: The value of this property must match the name of a partitioning strategy defined in the `partitioning_strategies` section. For more information, see Partitioning strategies.

`grid_partition`

Specifies the number of partitions along each axis (x and y) in a grid topology. These dimensions describe the total simulation space for this domain.

Required: Conditional. This property can only be specified if the topology is set to `"Grid"`. This property depends on the `grid_placement_groups` property of the specified partitioning strategy for this domain:

- This property is required if this domain's partitioning strategy doesn't specify a `grid_placement_groups` configuration.
- If there is a `grid_placement_groups` configuration but you don't specify `grid_partition` , then SimSpace Weaver will use the same dimensions as the `grid_placment_groups` configuration.
- If you specify both `grid_placement_groups` and `grid_partition` , the dimensions of `grid_partition` must be multiples of the dimensions of `grid_placement_groups` (for example, if your `grid_placement_groups` dimensions are 2x2, then some valid dimensions for `grid_partition` are 2x2, 4x4, 6x6, 8x8, 10x10).

Type: Integer (for each axis)

Valid values: `1-20`

Spatial app configuration

The `app_config` section (required) specifies the package, launch configuration, and resource requirements for apps in this domain.

```
app_config:
  package: "app-package-s3-uri"
  launch_command: ["app-launch-command", "parameter1", ...]
  required_resource_units:
    compute: app-resource-units
```

Properties

`package`

Specifies the package (zip file) that contains the app executable/binary. The package must be stored in an Amazon S3 bucket. Only zip file format is supported.

Required: Yes

Type: String

Valid values: The Amazon S3 URI of the package in an Amazon S3 bucket. For example, `s3://weaver-myproject-111122223333-app-zips-us-west-2/MySpatialApp.zip`.

`launch_command`

Specifies the executable/binary file name and command-line parameters to launch the app. Each command-line string token is an element in the array.

Required: Yes

Type: String array

`required_resource_units`

Specifies the number of resource units that SimSpace Weaver should allocate to each instance of this app. A *resource unit* is a fixed amount of virtual central processing units (vCPUs) and random-access memory (RAM) on a worker. For more information about resource units, see Endpoints and service quotas. The `compute` property specifies a resource unit allocation for the `compute` family of workers, and is currently the only valid type of allocation.

Required: Yes

Type: Integer

Valid values: 1-4

Custom container image

The `image` property (optional) specifies the location of a container image that SimSpace Weaver uses to run apps in this domain (not supported in versions `1.13` and `1.12`). Provide the URI to a repository in Amazon Elastic Container Registry (Amazon ECR) that contains the image. If this property isn't specified but the `default_image` is specified in the top level `simulation_properties` section, apps in this domain use the `default_image`. For more information, see Custom containers.

```
    image: "ecr-repository-uri"
```

Properties

`image`

Specifies the location of a container image to run apps in this domain.

Required: No

Type: String

Valid values:

- The URI of a repository in Amazon Elastic Container Registry (Amazon ECR) (for example, `111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest`)

Custom domain configuration

To specify the properties for a custom domain, replace *`custom-domain-name`* with a name of your choice. The name must be 3-64 characters long and can contain the characters **A** -**Z** , **a** -**z** , **0** -**9** , and **_** **-** (hyphen). Specify the properties of the custom domain after the name. Repeat this process for each custom domain.

```
custom-domain-name:
  launch_apps_via_start_app_call: {}
  app_config:
    package: "app-package-s3-uri"
    launch_command: ["app-launch-command", "parameter1", ...]
    required_resource_units:
      compute: app-resource-units
    endpoint_config:
```

```
        ingress_ports: [port1, port2, ...]
    image: "ecr-repository-uri"
```

Properties

`launch_apps_via_start_app_call`

This property is required to launch your custom apps using the **StartApp** API.

Required: Yes

Type: N/A

Valid values: {}

Custom app configuration

The `app_config section` (required) specifies the package, launch configuration, resource requirements, and network ports for apps in this custom domain.

```
    app_config:
      package: "app-package-s3-uri"
      launch_command: ["app-launch-command", "parameter1", ...]
      required_resource_units:
        compute: app-resource-units
      endpoint_config:
        ingress_ports: [port1, port2, ...]
```

Properties

`package`

Specifies the package (zip file) that contains the app executable/binary. The package must be stored in an Amazon S3 bucket. Only zip file format is supported.

Required: Yes

Type: String

Valid values: The Amazon S3 URI of the package in an Amazon S3 bucket. For example, `s3://weaver-myproject-111122223333-app-zips-us-west-2/MyCustomApp.zip`.

`launch_command`

Specifies the executable/binary file name and command-line parameters to launch the app. Each command-line string token is an element in the array.

Required: Yes

Type: String array

`required_resource_units`

Specifies the number of resource units that SimSpace Weaver should allocate to each instance of this app. A *resource unit* is a fixed amount of virtual central processing units (vCPUs) and random-access memory (RAM) on a worker. For more information about resource units, see Endpoints and service quotas. The `compute` property specifies a resource unit allocation for the `compute` family of workers, and is currently the only valid type of allocation.

Required: Yes

Type: Integer

Valid values: 1-4

`endpoint_config`

Specifies the network endpoints for apps in this domain. The value of `ingress_ports` specifies the ports that your custom apps bind to for incoming client connections. SimSpace Weaver maps dynamically allocated ports to your specified ingress ports. Ingress ports are both TCP and UDP. Use the **DescribeApp** API to find the actual port number to connect your clients.

Required: No. If you don't specify endpoint configuration then your custom apps in this domain won't have network endpoints.

Type: Integer array

Valid values: `1024-49152`. Values must be unique.

Custom container image

The `image` property (optional) specifies the location of a container image that SimSpace Weaver uses to run apps in this domain (not supported in versions `1.13` and `1.12`). Provide

the URI to a repository in Amazon Elastic Container Registry (Amazon ECR) that contains the image. If this property isn't specified but the `default_image` is specified in the top level `simulation_properties` section, apps in this domain use the `default_image`. For more information, see Custom containers.

```
    image: "ecr-repository-uri"
```

Properties

`image`

Specifies the location of a container image to run apps in this domain.

Required: No

Type: String

Valid values:

- The URI of a repository in Amazon Elastic Container Registry (Amazon ECR) (for example, `111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest`)

Service domain configuration

To specify the properties for a service domain, replace *`service-domain-name`* with a name of your choice. The name must be 3-64 characters long and can contain the characters **A** -**Z** , **a** -**z** , **0** -**9** , and **_** **-** (hyphen). Specify the properties of the service domain after the name. Repeat this process for each service domain.

```
service-domain-name:
  launch_apps_per_worker:
    count: number-of-apps-to-launch
  app_config:
    package: "app-package-s3-uri"
    launch_command: ["app-launch-command", "parameter1", ...]
    required_resource_units:
      compute: app-resource-units
    endpoint_config:
      ingress_ports: [port1, port2, ...]
  image: "ecr-repository-uri"
```

Launch apps per worker

The `launch_apps_per_worker` section (required) indicates that this is a service domain configuration, and specifies the number of service apps to launch per worker.

```
  launch_apps_per_worker:
    count: number-of-apps-to-launch
```

Properties

`count`

This property specifies the number of service apps to launch per worker.

Required: Yes

Type: Integer

Valid values: `{}` | 1 | 2. A value of `{}` specifies the default value of 1.

Service app configuration

The `app_config section` (required) specifies the package, launch configuration, resource requirements, and network ports for apps in this service domain.

```
  app_config:
    package: "app-package-s3-uri"
    launch_command: ["app-launch-command", "parameter1", ...]
    required_resource_units:
      compute: app-resource-units
    endpoint_config:
      ingress_ports: [port1, port2, ...]
```

Properties

`package`

Specifies the package (zip file) that contains the app executable/binary. The package must be stored in an Amazon S3 bucket. Only zip file format is supported.

Required: Yes

Type: String

Valid values: The Amazon S3 URI of the package in an Amazon S3 bucket. For example, `s3://weaver-myproject-111122223333-app-zips-us-west-2/MyServiceApp.zip`.

`launch_command`

Specifies the executable/binary file name and command-line parameters to launch the app. Each command-line string token is an element in the array.

Required: Yes

Type: String array

`required_resource_units`

Specifies the number of resource units that SimSpace Weaver should allocate to each instance of this app. A *resource unit* is a fixed amount of virtual central processing units (vCPUs) and random-access memory (RAM) on a worker. For more information about resource units, see Endpoints and service quotas. The `compute` property specifies a resource unit allocation for the `compute` family of workers, and is currently the only valid type of allocation.

Required: Yes

Type: Integer

Valid values: `1-4`

`endpoint_config`

Specifies the network endpoints for apps in this domain. The value of `ingress_ports` specifies the ports that your service apps bind to for incoming client connections. SimSpace Weaver maps dynamically allocated ports to your specified ingress ports. Ingress ports are both TCP and UDP. Use the **DescribeApp** API to find the actual port number to connect your clients.

Required: No. If you don't specify endpoint configuration then your service apps in this domain won't have network endpoints.

Type: Integer array

Valid values: `1024-49152`. Values must be unique.

Custom container image

The `image` property (optional) specifies the location of a container image that SimSpace Weaver uses to run apps in this domain (not supported in versions `1.13` and `1.12`). Provide the URI to a repository in Amazon Elastic Container Registry (Amazon ECR) that contains the image. If this property isn't specified but the `default_image` is specified in the top level `simulation_properties` section, apps in this domain use the `default_image`. For more information, see Custom containers.

```
    image: "ecr-repository-uri"
```

Properties

`image`

Specifies the location of a container image to run apps in this domain.

Required: No

Type: String

Valid values:

- The URI of a repository in Amazon Elastic Container Registry (Amazon ECR) (for example, `111122223333.dkr.ecr.us-west-2.amazonaws.com/my-ecr-repository:latest`)

Placement constraints

The `placement_constraints` section (optional) specifies which spatial domains SimSpace Weaver should place together on the same workers. For more information, see Configuring spatial domains.

> **Important**
>
> Versions `1.13` and `1.12` don't support `placement_constraints`.

```
placement_constraints:
  - placed_together: ["spatial-domain-name", "spatial-domain-name", ...]
```

```
    on_workers: ["worker-group-name"]
```

Properties

`placed_together`

Specifies the spatial domains that SimSpace Weaver should place together.

Required: Yes

Type: String array

Valid values: Names of spatial domains specified in the schema

`on_workers`

Specifies the worker group that that SimSpace Weaver should place the domains on.

Required: Yes

Type: 1-element string array

Valid values: Name of a worker group specified in the schema

SimSpace Weaver API references

SimSpace Weaver has 2 different sets of application programming interfaces (APIs):

- **Service APIs** – These APIs control the service and service resources, such as your simulations, clocks, and apps. They are part of the main AWS software development kit (SDK) and you can use the AWS Command Line Interface (CLI) to call them. You can also use the convenience script in the tools folder for your project and platform (for example, *project-folder*\tools\windows\weaver-*project-name*-cli.bat) to call these APIs. For more information about the servcie APIs, see the SimSpace Weaver API reference.
- **App SDK APIs** – These APIs control the data within your simulation. You use them in your app code to do things like read and write entity field data, work with subscriptions, and monitor events in the simulation. For more information, see the SimSpace Weaver app SDK documentation in your unzipped app SDK folder: *sdk-folder*\SimSpaceWeaverAppSdk-1.15.3\documentation

> **Note**
>
> *sdk-folder* is the folder where you unzipped the SimSpaceWeaverAppSdkDistributable package. If *sdk-folder*\SimSpaceWeaverAppSdk-1.15.3 doesn't exist, make sure that you have version 1.15.3 of the SimSpace Weaver app SDK. The SimSpaceWeaverAppSdkDistributable doesn't contain the entire SimSpace Weaver app SDK. The first time you use the SimSpace Weaver app SDK scripts to create a project, the scripts create the SimSpaceWeaverAppSdk-1.15.3 folder inside your *sdk-folder* and download the rest of the SimSpace Weaver app SDK into it.

AWS SimSpace Weaver versions

We continuously improve AWS SimSpace Weaver. You must download the latest SimSpace Weaver app SDK when we release a new version if you want to take advantage of new features and feature updates. To run an existing simulation with a newer version, you might have to update its schema and code, then start a new instance of the simulation. You don't have to upgrade, and can continue to run existing simulations with previous versions. You can check this page to see what's different between the versions. All versions are currently supported.

Latest version

The latest version is: 1.15.3

How to find your current version

If you created a simulation with the SimSpace Weaver app SDK, the `create-project` script downloaded a version of the SDK libraries into a subdirectory in your *`sdk-folder`*. The subdirectory that contains the SDK libraries has a name that includes the SDK version number: `SimSpaceWeaverAppSdk-`*`sdk-version`*. For example, the libraries for version 1.15.2 are in `SimSpaceWeaverAppSdk-1.15.2`.

You can also find the version of the SimSpace Weaver app SDK distributable package in the text file `app_sdk_distributable_version.txt` in your *`sdk-folder`*.

Download the latest version

Use one of the following links to download the latest version.

- Complete app SDK distributable package
- Only the app SDK libraries

You can also download the complete SimSpace Weaver app SDK distributable package from the SimSpace Weaver console in the AWS Management Console. Choose **Download app SDK** from the navigation pane.

> **Warning**
>
> Don't use the AWS CLI to download anything that appears to be the SimSpace Weaver app SDK distributable package. Only use the download links on this page or the download link in the console. Any other download method or location isn't supported and might contain obsolete, incorrect, or malicious code.

Troubleshooting app SDK downloads

We use Amazon CloudFront (CloudFront) to distribute the app SDK .zip files. You might experience some of the following situations.

- **The downloaded package isn't the latest version**
  - If the .zip file that you downloaded doesn't contain the latest version, it's possible that the cache at your CloudFront edge location hasn't updated yet. Download again after 24 hours.
- **You get a HTTP 4xx or 5xx error using a download link**
  - Try again after 24 hours. If you get the same error, use the **Feedback** link at the bottom of the SimSpace Weaver console to report the problem. Select **Report an issue** as the **Type** of feedback.
- **Your browser reports that it can't load the page**
  - You might have a local network or browser configuration problem. Verify that you can load other pages. Clear your browser cache and try again. Make sure that you don't have firewall rules that might block the download URL.
- **You get an error when you try to save the file**
  - Check your local file system permissions to make sure that you have the correct permissions to save the file.
- **Your browser displays AccessDenied**
  - If you manually entered the URL into your browser, check that it is correct. If you used a download link, make sure something didn't interfere with the URL in your browser; use the link again.

Install the latest version

To install the latest version

1. Download the latest version.
2. Unzip the SimSpaceWeaverAppSdkDistributable.zip to a folder.
3. Run `docker-create-image.bat` (or `docker-create-image.sh` for WSL) from the unzipped latest version SimSpace Weaver app SDK folder.
4. Use the unzipped latest version SimSpace Weaver app SDK folder instead of the previous version.

Service versions

Version	Notes	Release date	App SDK download
1.15.3	**SimSpace Weaver Local update:** • We changed SimSpace Weaver Local to more closely align it with development for the AWS Cloud. These changes impact C++, Python, Unity, and Unreal Engine projects and workflows for SimSpace Weaver Local. For more information, see Local development differences in version 1.15.3.	December 4, 2023	• Complete package • Libraries only See Troubleshooting.

Version	Notes	Release date	App SDK download
1.15.2	**App SDK distributable package update:** • We updated the `Dockerfile` to use the specific required version of cmake. Docker container builds might fail without this change.	November 2, 2023	Not available for download
1.15.1	**Feature update:** • **Python SDK:** This release fixes an issue that caused Python-based simulations to fail in the AWS Cloud. For more information, see the version notes.	September 22, 2023	Not available for download
1.15.0	**New feature:** • **Python SDK:** You can now develop your simulations in Python. The SimSpace Weaver app SDK distributable package includes a template for a sample Python project and its Python view client. For more information, see Working with Python.	August 31, 2023	Not available for download

Version	Notes	Release date	App SDK download
1.14.0	**New features:** • **Custom containers:** Create your own Amazon Linux 2 (AL2) based container image, store it in Amazon Elastic Container Registry (Amazon ECR), and use it to run your SimSpace Weaver apps in the AWS Cloud. For more information, see Custom containers. • **Multiple spatial domains:** Create more than 1 spatial domain in a simulation. Separate simulation logic instead of combining it all into a single spatial app. Allocate different resources to spatial domains based on their requirements. For more information, see Configuring spatial domains. • **Unlimited tick rate:** Enable your simulation to run as fast as your code can execute. Set your simulation's clock	July 26, 2023	Not available for download

Version	Notes	Release date	App SDK download
	so that it sends the next tick as soon as all apps finish their commit operations for the current tick. For more information, see Clock. **SimSpace Weaver app SDK:** • The value of `tick_rate` is now a string. The value must include double quotes ("). The tick rate for earlier versions is still an integer. For more information, see Clock.		
1.13.1	**SimSpace Weaver app SDK:** • Feature update: Project creation now works correctly with the `PathfindingSampleUnreal` template.	June 7, 2023	Not available for download

Version	Notes	Release date	App SDK download
1.13.0	**SimSpace Weaver service APIs:** • • New CreateSnapshot action • Changes to the StartSimulation action: • Added a SnapshotS3Location parameter to start from a snapshot. • The SchemaS3Location parameter is now optional. • Changes to DescribeSimulation output: • SchemaError is deprecated. • Added a StartError field. • Added a SnapshotS3Location field. • Added a SNAPSHOT_	April 29, 2023	Not available for download

Version	Notes	Release date	App SDK download

	`IN_PROGRESS` simulation status.
	• New `S3Destination` data type
	
	SimSpace Weaver console:
	• New functionality to create snapshots.
	• New functionality to start a simulation from a snapshot.
	
	SimSpace Weaver app SDK:
	• New scripts to support snapshots
	  • `create-snapshot-`*`project-name`*`.bat`
	  • `start-from-snapshot-`*`project-name`*`.bat`
	  • `quick-start-from-snapshot-`*`project-name`*`-cli.bat`
	  • `list-snapshots-`*`project-name`*`.bat`		

Version	Notes	Release date	App SDK download
	• Projects now use a single Amazon S3 bucket per project: weaver-*lowercase-project-name* -*account-number* -*region* For more information about snapshots, see Snapshots.		

Version	Notes	Release date	App SDK download

1.12.3	**SimSpace Weaver app SDK:**
	• The following scripts now support the `--maximum-duration` parameter:
	  • `quick-start-` *project-name* `-cli.bat`
	  • `quick-start-` *project-name* `-cli.sh`
	  • `start-simulation-` *project-name* `.bat`
	  • `start-simulation-` *project-name* `.sh`
	  • `run-` *project-name* `.bat`
	  • `run-` *project-name* `.sh`
	For more information, see Maximum duration of a simulation.
		March 27, 2023	Not available for download

1.12.2	**SimSpace Weaver app SDK:** • Bug fix: `docker-create-image.bat` now runs correctly.	March 1, 2023	Not available for download

Version	Notes	Release date	App SDK download
1.12.1	**SimSpace Weaver app SDK:** • The scripts now accept an AWS CLI profile to use for AWS authentication. • The scripts now support AWS IAM Identity Center for AWS authentication. **SimSpace Weaver Local:** • Bug fix: `Api::BeginUpdateWillBlock` now correctly returns `true` if all of the spatial apps have not joined the simulation.	February 28, 2023	Not available for download
1.12.0	Release for general availability (GA)	November 29, 2022	Not available for download

AWS SimSpace Weaver version 1.15.1

This release is a **required** update for the Python SDK that was originally released in SimSpace Weaver version 1.15.0. It fixes a version mismatch issue that caused Python-based simulations to fail in the AWS Cloud. Use this version instead of 1.15.0.

Update an existing Python project to 1.15.1

If you have an existing Python project that you created with the version 1.15.0 Python SDK, you must perform the following steps to update it to 1.15.1 so that it can run in the AWS Cloud.

Instead of following this procedure, you can also create a new Python project with the 1.15.1 Python SDK and move your custom code to the new project.

To update a 1.15.0 Python project to 1.15.1

1. Go to your Python project's folder.
2. In `src/PythonBubblesSample/bin/run-python` change the following line:

   ```
   export PYTHONPATH=$PYTHONPATH:/roapp/lib
   ```

   To the following:

   ```
   export PYTHONPATH=$PYTHONPATH:$LD_LIBRARY_PATH:/roapp/lib
   ```

3. In `CMakeLists.txt` delete the following lines:

   - ```
     file(COPY "${SDK_PATH}/libweaver_app_sdk_python_v1_$ENV{PYTHON_VERSION}.so"
      DESTINATION "${ZIP_FILES_DIR}/lib/weaver_app_sdk_v1")
     ```
   - ```
     file(RENAME "${ZIP_FILES_DIR}/lib/weaver_app_sdk_v1/libweaver_app_sdk_python_v1_
     $ENV{PYTHON_VERSION}.so" "${ZIP_FILES_DIR}/lib/weaver_app_sdk_v1/
     libweaver_app_sdk_python_v1.so")
     ```
   - ```
     message("    * COPYING WEAVER PYTHON SDK TO BUILD DIR ${ZIP_FILES_DIR}....")
     ```
   - ```
     file(COPY ${SDK_DIR} DESTINATION ${ZIP_FILES_DIR}/lib/weaver_app_sdk_v1)
     ```

Troubleshooting for version 1.15.1

After updating a 1.15.0 Python simulation, it fails to start in the AWS Cloud

Symptoms: After approximately 5-10 minutes after you start the simulation, the simulation management log reports an `internal error` and the simulation status is `FAILED`.

This can happen if a library file from the 1.15.0 Python SDK is included in an app zip file. Make sure that you completed the steps to update your project, and make sure that `libweaver_app_sdk_python_v1.so` isn't in your zip files or referenced in any way.

Frequently asked questions about version 1.15.1

Does this release affect anything other than the Python SDK?

No.

Do I have to update to version 1.15.1?

You don't have to update to 1.15.1 if you don't intend to use Python for your spatial apps. If you updated to 1.15.0, your Python-based simulations won't run in the AWS Cloud. We recommend that you update to 1.15.1 if you use 1.15.0.

What is `$LD_LIBRARY_PATH`?

It's the location of the Python SDK when your simulation runs in the AWS Cloud. This is new for 1.15.1. We made this change to avoid Python version problems in the future. Linking to that directory is functionally the same as linking to `libweaver_app_sdk_python_v1.so` in 1.15.0.

Document history for AWS SimSpace Weaver

The following table describes important changes to the SimSpace Weaver documentation.

Date	Change	Documentation updates	API versions updated
May 20, 2025	End of support notice	End of support notice: On May 20, 2026, AWS will end support for AWS SimSpace Weaver. After May 20, 2026, you will no longer be able to access the SimSpace Weaver console or SimSpace Weaver resources. For more information, see AWS SimSpace Weaver end of support.	N/A
December 4, 2023	Updated content	Updated the AWS SimSpace Weaver versions chapter for the version 1.15.3 release.	N/A
December 4, 2023	Updated content	Updated the AWS SimSpace Weaver versions chapter to include installation instructions for the latest version.	N/A
December 4, 2023	Updated content	Updated the Service quotas for SimSpace Weaver Local.	N/A

Date	Change	Documentation updates	API versions updated
December 4, 2023	New and updated content	Restructured the Local development section and added a new page that describes the differenc es for SimSpace Weaver Local introduced in version 1.15.3.	N/A
November 7, 2023	Updated content	Updated the instructions to set up for Docker and WSL to use the direct download link/URL for the app SDK. For more information, see Set up your local environment for SimSpace Weaver.	N/A
November 2, 2023	Updated content	Updated the service versions page for the 1.15.2 release. For more information, see Service versions.	N/A

Date	Change	Documentation updates	API versions updated
October 23, 2023	Updated content	Updated the service versions page with new instructions to download the app SDK distribut able package. Customers should now only use one of our approved direct download links and not use the AWS CLI to download the app SDK distributable package. For more information, see Download the latest version.	N/A
September 22, 2023	New content	Added a version notes page with update instructions for the 1.15.1 release. For more information, see AWS SimSpace Weaver version 1.15.1.	N/A
September 10, 2023	New content	Added a troubleshooting section for situation s where the AWS CLI doesn't recognize SimSpace Weaver. For more information, see The AWS CLI doesn't recognize `simspacew eaver` .	N/A

Date	Change	Documentation updates	API versions updated
September 10, 2023	Updated content	Updated installation instructions for the AWS CLI in WSL. For more information, see Set up Amazon Linux 2 (AL2) in Windows Subsystem for Linux (WSL).	N/A
September 7, 2023	API update	BucketName and ObjectKey are now required for the S3Location data type. BucketName is now required for the S3Destination data type.	AWS SDK: 2023-09-07
August 31, 2023	New content	Added a new section for release 1.15.0: Working with Python.	N/A
August 15, 2023	Updated content	Updated download instructions in AWS SimSpace Weaver versions to only list official SimSpace Weaver Amazon S3 buckets. Other download locations are not controlled by AWS and might contain malicious code.	N/A
July 26, 2023	Updated content	Updated Clock.	N/A

Date	Change	Documentation updates	API versions updated
July 26, 2023	Updated content	Updated Configuring spatial domains.	N/A
July 26, 2023	New content	Added a new section: Custom containers.	N/A
July 26, 2023	Updated content	Updated AWS SimSpace Weaver versions for release 1.14.0.	N/A
July 6, 2023	New content	Added a new section: PathfindingSample console client fails to connect.	N/A
June 7, 2023	Updated content	Updated AWS SimSpace Weaver versions for release 1.13.1.	N/A
May 15, 2023	New content	Added a new section: Using snapshots with AWS CloudFormation.	N/A
April 29, 2023	New content	Added content for release 1.13.0. For more information, see AWS SimSpace Weaver versions.	AWS SDK: 2023-04-28

Date	Change	Documentation updates	API versions updated
March 27, 2023	New content	Added a section that describes the maximum duration of simulations. Added notes in the tutorials for release 1.12.3, which added support for the `--maximum-duration` parameter to SimSpace Weaver app SDK scripts.	N/A
March 9, 2023	Changed content	Clarified that we only provide instructions for Docker on Windows and for Windows Subsystem for Linux (WSL), and that WSL (and any other Linux environment) is unsupported.	N/A
February 28, 2023	New content	Added a chapter that describes SimSpace Weaver versions.	N/A
February 28, 2023	Changed content	Changed content about authentication to include the use of AWS IAM Identity Center and named profiles for the AWS Command Line Interface (AWS CLI).	N/A

Date	Change	Documentation updates	API versions updated
February 17, 2023	New content	Added a section about managing your resources with AWS CloudForm ation.	N/A
January 23, 2023	New content	Added instructions for debugging local simulations.	N/A
November 29, 2022	Service launch	Released the User Guide and API Reference for SimSpace Weaver.	AWS SDK: 2022-11-29

Glossary

This glossary defines terms that are specific to AWS SimSpace Weaver.

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.

A

app	Executable code (also called *binaries*) that you create. The term *app* can refer to the code or a running instance of that code. An app encapsulates simulation behavior. Apps create, delete, read, and update entities.
app SDK	A software development kit (SDK) that you use to integrate an app with SimSpace Weaver. The SDK provides APIs for reading and writing entity data and tracking simulation time. For more information, see SimSpace Weaver app SDK.

C

client	Processes (or their definitions) that exist outside of SimSpace Weaver and interact with the simulation through a custom app or service app. You can use a client to view or change the simulation state.
clock	An abstraction of SimSpace Weaver's internal scheduling processes. The clock publishes ticks to apps to maintain time synchronization. Each simulation has its own clock.
clock rate	The number of ticks per second that the clock publishes to apps. For more information about supported clock rates, see SimSpace Weaver endpoints and quotas.
clock tick rate	See clock rate.
compute resource unit	A unit of compute resources (processor and memory) on a worker. A single instance of an app is normally allocated 1 compute resource unit. You can allocate more than 1 compute resource unit for each app.
custom app	A type of app that you use to read and interact with the state of the simulation. Custom apps can create entities in the simulation but don't own them. When a custom app creates an entity, it must transfer the

	entity to the spatial domain. You control the lifecycle of a custom app using the app APIs. For more information about the SimSpace Weaver APIs, see SimSpace Weaver API references.
custom domain	A domain that contains custom apps.
custom partition	The partition of a custom app.

D

deadline	An actual time by which an operation (such as processing for a tick) should be complete.
domain	A group of app instances that run the same executable code (app binary) and have the same launch options.

E

endpoint (service)	A fully-qualified domain name (FQDN) that programs (such as the AWS Command Line Interface) use to connect to the SimSpace Weaver service.
endpoint (simulation)	An IP address and port number that clients use to connect to connect to a simulation. You can configure endpoints on custom apps and service apps.
entity	Customer data objects (or their definitions). Entities can be static (remain in one location) or dynamic (move through the simulation space). For example, people and buildings in a simulation.

I

index (simulation)	A description of the spatial properties of a simulation, including its spatial boundaries and coordinate system.

L

lifecycle (of an app)	A description of the expected logical steps that an app goes through during a simulation. Lifecycles are either *managed* (SimSpace Weaver starts and stops the app) or *unmanaged* (you start and stop the app).

load (entity field data)	Read entity field data from the State Fabric.

P

partition	A segment of shared memory on a worker. Each partition contains a discrete subset of entities within a domain. Each app has an assigned partition. An app owns all of the entities in its partition. When an app creates an entity, it creates it in its partition. When entities move from one partition to another partition, ownership transfers from the source partition's app to the destination partition's app.

R

resource unit	See ???.

S

schema	A YAML or JSON document that describes the configuration of a simulation. SimSpace Weaver uses a schema to create a simulation resource.
service app	A type of app that you use to read and interact with the state of the simulation. Service apps can create entities in the simulation but must transfer them to the spatial domain. SimSpace Weaver manages the lifecycle of a service app, and starts 1 (or more, as specified in your simulation schema) on each worker in your simulation.
service domain	A domain that contains service apps.
service partition	The partition of a service app.
simulation (resource)	An abstraction of a compute cluster that runs a simulated virtual space. You can have multiple simulations. You configure a simulation using a schema.
spatial app	A type of app that encapsulates the core simulation logic. Each spatial app owns 1 (and only 1) partition.
spatial domain	A domain that contains spatial apps.

spatial partition	The partition of a spatial app.
State Fabric	SimSpace Weaver's in-memory database. The State Fabric stores the state of simulations, including entities and internal SimSpace Weaver data.
store (entity field data)	Write entity field data to the State Fabric.
subscription	A long-running request for a specific app instance to receive data from a subscription area. The subscribing app uses a subscription to discover changes to entities inside the subscription area.
subscription area	A 2-dimensional region of the simulation space. A subscription refers to a subscription area. A subscription area can span more than 1 partition, and also include parts of partitions. A subscription area is continuous within its defined bounds.

T

tick	A discrete value for time (either wall-clock time or simulation time). Apps can iterate faster than the tick duration, but are expected to write specified ticks within specific deadlines. All operations for all apps for a given tick must complete before the next tick can start.
tick rate	See clock rate.
time (actual)	The current time from the perspective of reality. SimSpace Weaver uses a 64-bit POSIX timestamp which is the number of nanoseconds since the Unix epoch (January 1, 1970, 00:00:00 UTC).
time (simulation)	The current time from the perspective of the simulation. SimSpace Weaver uses a 64-bit integer logical tick counter, which might not directly correspond to the actual time.

W

worker	An Amazon Elastic Compute Cloud (Amazon EC2) instance that runs simulation code.