Implementation Guide

IoT Device Simulator

IoT Device Simulator: Implementation Guide

Table of Contents

Solution overview 1
Features and benefits 2
Use cases 4
Concepts and definitions 4
Architecture overview 6
Architecture diagram 6
AWS Well-Architected design considerations 7
Operational excellence 7
Security 7
Reliability 8
Performance efficiency 8
Cost optimization 8
Sustainability 8
Architecture details 9
AWS Step Functions workflow 9
Solution microservices 10
Web interface 10
Routes bucket 11
AWS services in this solution 11
Plan your deployment 13
Cost 13
Sample cost table 13
Security 14
IAM roles 14
AWS IoT Core policies 14
Amazon CloudFront 14
Amazon API Gateway 15
Supported AWS Regions 15
Quotas 16
Quotas for AWS services in this solution 16
Amazon Cognito 16
Simulation limits 16
Deploy the solution 17
Deployment process overview 17

AWS CloudFormation template 18
Step 1: Launch the stack 18
Step 2: Sign in to the web interface 20
Step 3: Create the device types 20
Step 4: Create the simulations 22
Step 5: Run and manage the simulations 24
View the simulations 24
Manage the device types 25
Manage the simulations 25
Step 6: Post-deployment configuration 25
Amazon CloudWatch dashboard and alarms 25
Additional security considerations 26
Monitor the solution with Service Catalog AppRegistry 27
Activate CloudWatch Application Insights 27
Confirm cost tags associated with the solution 29
Activate cost allocation tags associated with the solution 30
AWS Cost Explorer 30
Update the solution 31
Troubleshooting 33
Contact AWS Support 33
Create case 33
How can we help? 33
Additional information 33
Help us resolve your case faster 34
Solve now or contact us 34
Uninstall the solution 35
Using the AWS Management Console 35
Using AWS Command Line Interface 35
Deleting the Amazon S3 buckets 35
Deleting DynamoDB tables 36
Automotive demo 37
Routes 37
Running the automotive demo 37
Developer guide 39
Source code 39
Device type JSON structure 39

Reference 45
Anonymized data collection 45
Contributors 46
Revisions 48
Notices 51

Create and simulate hundreds of virtual connected devices without having to configure and manage physical devices

Publication date: *May 2018 (last update: April 2024)*

Amazon Web Services (AWS) helps customers build serverless IoT applications without having to manage any infrastructure. Customers can also use AWS to build a secure, agile, and scalable backend for their IoT applications, reducing backend resource costs and increasing productivity and innovation. However, testing IoT applications and backend services is costly and can be a challenge due to the large pool of physical, connected devices required.

IoT Device Simulator is designed to help customers more easily test device integration and IoT backend services, without the need for physical devices. This solution provides a web interface for you to create and simulate hundreds of connected devices, without having to configure and manage physical devices, or develop time-consuming scripts. Launch fleets of virtually connected devices from a user-defined template and then simulate them to publish data at regular intervals to AWS IoT. You can also monitor devices from the simulator or observe how backend services are processing the data.

This solution is designed to work out-of-the-box, or use this solution as a reference implementation to build a custom simulation engine for your specific use case.

This implementation guide provides an overview of the IoT Device Simulator solution, its reference architecture and components, considerations for planning the deployment, configuration steps for deploying the solution to the AWS Cloud.

The intended audience for using this solution's features and capabilities in their environment includes solution architects, business decision makers, DevOps engineers, data scientists, and cloud professionals.

> **Note**
>
> This solution is designed to simulate device data for testing. It is not recommended for use in production environments.

Use this navigation table to quickly find answers to these questions:

If you want to . . .	Read . . .
Know the cost for running this solution. The estimated cost for running this solution using the 100 automotive demo device types in a single simulation, sending a message every two seconds in the US East (N. Virginia) Region is USD **$3.05 per month** for a simulation running six hours per day.	Cost
Understand the security considerations for this solution.	Security
Know how to plan for quotas for this solution.	Quotas
Know which AWS Regions support this solution.	Supported AWS Regions
View or download the AWS CloudForm ation template included in this solution to automatically deploy the infrastructure resources (the “stack”) for this solution.	AWS CloudFormation template
Access the source code and optionally use the AWS Cloud Development Kit (AWS CDK) (AWS CDK) to deploy the solution.	GitHub repository

Features and benefits

The solution provides the following features:

Realistic simulation scenarios

IoT Device Simulator allows users to create and run realistic simulation scenarios tailored to their specific use cases. Users can define various parameters such as device types, behaviors, and data

patterns to accurately replicate real-world IoT device environments. This enables users to test their IoT applications under diverse and realistic conditions, helping them identify potential issues and optimize their solutions for performance and reliability.

Scalable simulation workloads

With the IoT Device Simulator solution, users can easily scale their simulation workloads to accommodate varying levels of demand. The solution leverages AWS services such as AWS Lambda and Amazon DynamoDB to dynamically adjust resources based on workload requirements.

Users can conduct large-scale simulations without worrying about infrastructure provisioning or capacity constraints. This flexibility allows for efficient testing of IoT applications across different scenarios and usage patterns.

Integration with AWS IoT

IoT Device Simulator seamlessly integrates with AWS IoT, allowing users to leverage the full capabilities of the IoT platform. Users can easily create IoT things, define thing types, and manage device connections directly from the simulator interface.

This integration streamlines the simulation setup process and enables users to leverage existing AWS IoT Core features and functionalities within their simulation scenarios. It also ensures compatibility and consistency with production IoT deployments on AWS.

Comprehensive monitoring and logging

This solution provides comprehensive monitoring and logging capabilities, allowing users to track simulation metrics and analyze simulation results in real time. Users can monitor key performance indicators (KPIs) such as message throughput, latency, and error rates to gain insights into simulation performance.

Integration with Service Catalog AppRegistry and Application Manager, a capability of AWS Systems Manager

This solution includes a Service Catalog AppRegistry resource to register the solution's CloudFormation template and its underlying resources as an application in both Service Catalog AppRegistry and Application Manager. With this integration, you can centrally manage the solution's resources and enable application search, reporting, and management actions.

Use cases

Load testing IoT infrastructure

Simulation scenarios can be configured to generate high volumes of traffic, simulating a large number of IoT devices interacting with the infrastructure simultaneously. This use case allows users to assess the scalability and performance of their IoT infrastructure under heavy loads. By analyzing metrics such as message throughput, latency, and error rates during load testing, users can identify potential bottlenecks and optimize their infrastructure for reliability and responsiveness.

Validating IoT application logic

The IoT Device Simulator enables users to simulate diverse scenarios to validate the logic and functionality of their IoT applications. Users can define custom behaviors for simulated devices, including data generation patterns, event triggers, and interaction flows. This facilitates comprehensive testing of application logic across various use cases and edge cases, helping users identify and address potential issues before deploying their applications in production. By validating IoT application logic through simulation, users can ensure the reliability and accuracy of their applications in real-world scenarios.

Training machine learning (ML) models

Simulation scenarios can be utilized to generate synthetic data for training ML models used in IoT applications. By simulating different environmental conditions, device behaviors, and data patterns, users can generate diverse datasets to train and validate ML algorithms. This use case enables users to develop and refine ML models in a controlled environment, without relying on costly or limited real-world data sources. By leveraging simulation for training ML models, users can accelerate the development process and improve the accuracy and robustness of their models for deployment in production IoT applications.

Concepts and definitions

This section describes key concepts and defines terminology specific to this solution:

device simulator

An AWS Step Functions state machine runs the device simulator. The state machine consists of an AWS Lambda function which provides the logic to create the device messages and send them to

the IoT endpoint. An architecture diagram of the workflow and a complete walkthrough can be found in the the section called "AWS Step Functions workflow" section.

microservices

The IoT Device Simulator microservices are a series of AWS Lambda functions that provide the business logic and data access layer for all device simulation operations. This includes create, read, update, and delete (CRUD) operations for the Amazon DynamoDB simulation and device type tables, as well as starting the step functions workflow. Each Lambda function assumes an AWS Identity and Access Management (IAM) role with least privilege access (minimum permissions necessary) to perform its designated functions.

web interface

The solution includes an intuitive web interface which is hosted in Amazon S3 and used to simulate the devices. You can use the interface to create and manage simulations and device types, and start device simulations to simulate devices and send messages to the AWS IoT endpoint.

The interface is designed to simulate devices that publish to an AWS IoT endpoint at regular intervals in order to test backend integration.

automotive demo

An Amazon S3 bucket is used to host the pre-defined routes for the automotive demo. The routes are used to provide a pathway for the simulated vehicle to follow. For more details about these routes, refer to Routes in this guide.

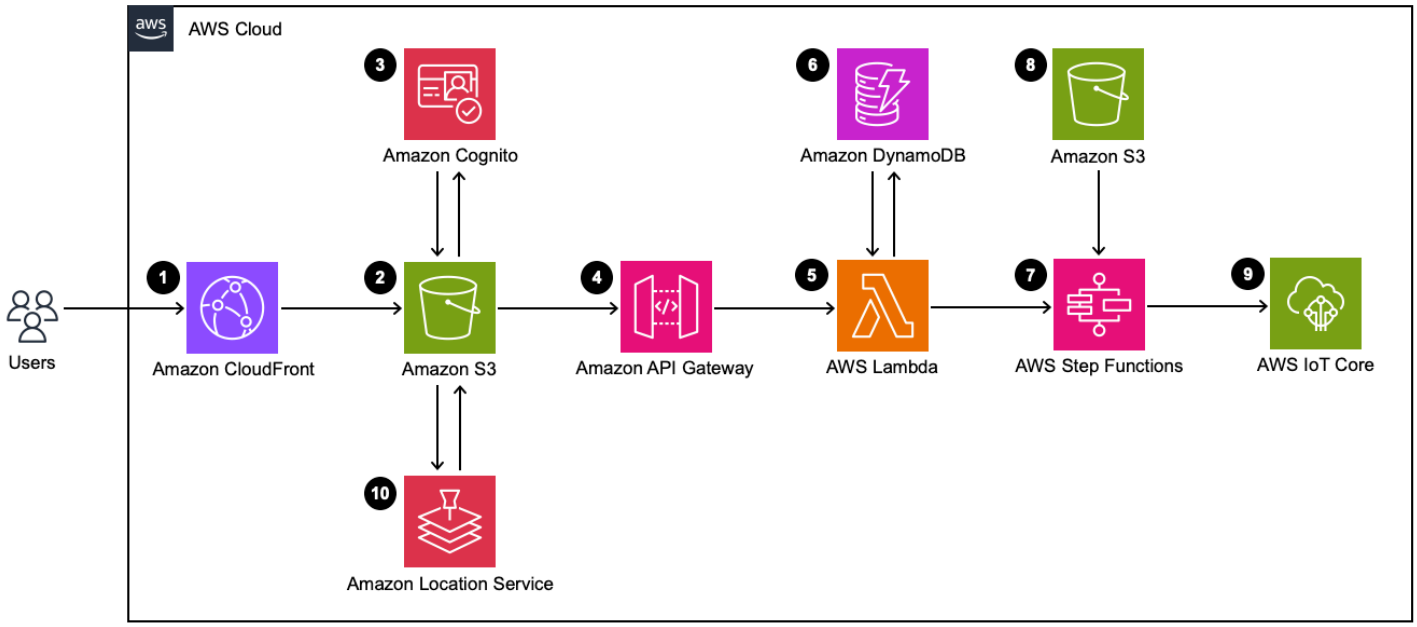
For a general reference of AWS terms, see the AWS Glossary.

Architecture overview

This section provides a reference implementation architecture diagram for the components deployed with this solution and summary of AWS Well-Architected design considerations.

Architecture diagram

Deploying this solution with the default parameters deploys the following components in your AWS account.



IoT Device Simulator architecture

> **Note**
>
> AWS CloudFormation resources are created from AWS Cloud Development Kit (AWS CDK) constructs.

The high-level process flow for the solution components deployed with the AWS CloudFormation template is as follows:

1. Amazon CloudFront to serve the web interface content from an Amazon Simple Storage Service (Amazon S3) bucket.

2. An Amazon S3 bucket hosts the web interface.
3. An Amazon Cognito user pool authenticates the API requests.
4. An Amazon API Gateway API provides the solution's API layer.
5. AWS Lambda serves as the solution's microservices and routes API requests.
6. Amazon DynamoDB stores simulation and device type information.
7. AWS Step Functions include an AWS Lambda simulator function to simulate devices and send messages.
8. An Amazon S3 bucket stores pre-defined routes that are used for the automotive demo.
9. AWS IoT Core serves as the endpoint to which messages are sent.
10. Amazon Location Service provides the map display showing the location of automotive devices for the automotive demo.

AWS Well-Architected design considerations

This solution uses the best practices from the AWS Well-Architected Framework, which helps customers design and operate reliable, secure, efficient, and cost-effective workloads in the cloud.

This section describes how the design principles and best practices of the Well-Architected Framework benefit this solution.

Operational excellence

This section describes how we architected this solution using the principles and best practices of the operational excellence pillar.

This solution pushes metrics to Amazon CloudWatch at various stages to provide observability into the infrastructure; Lambda functions, Amazon S3 buckets, and the rest of the solution components. Continuous integration and continuous delivery (CI/CD) and infrastructure deployment are managed by CloudFormation.

Security

This section describes how we architected this solution using the principles and best practices of the security pillar.

All internet accessible endpoints are protected by authentication. All databases are closed off from anything external to the AWS account. All data is encrypted at rest and in transit with rotating

encryption keys. Permissions are locked down to zero-trust principles or least-privilege; the most restrictive choice is made where possible.

Reliability

This section describes how we architected this solution using the principles and best practices of the reliability pillar.

IoT Device Simulator uses primarily serverless AWS services, which provides resiliency, uptime, and automatic scaling. All appropriate Amazon S3 buckets have versioning enabled and are backup protected. All DynamoDB tables have point-in-time recovery, and customer data is not deleted when you uninstall the solution.

Performance efficiency

This section describes how we architected this solution using the principles and best practices of the performance efficiency pillar.

All compute and performance efficiency relates to usage and not a base cost. Complex tasks are delegated to appropriate AWS services that provide built-in, efficient functionality. You can deploy in the section called “Supported AWS Regions” to keep your data closer to where it's being used and processed, minimizing delays.

Cost optimization

This section describes how we architected this solution using the principles and best practices of the cost optimization pillar.

AWS Billing and Cost Management provides cost observation and analysis. IoT Device Simulator follows a consumption model, so costs are driven by usage.

Sustainability

This section describes how we architected this solution using the principles and best practices of the sustainability pillar.

This solution uses managed and serverless services to minimize the environmental impact of the backend services.

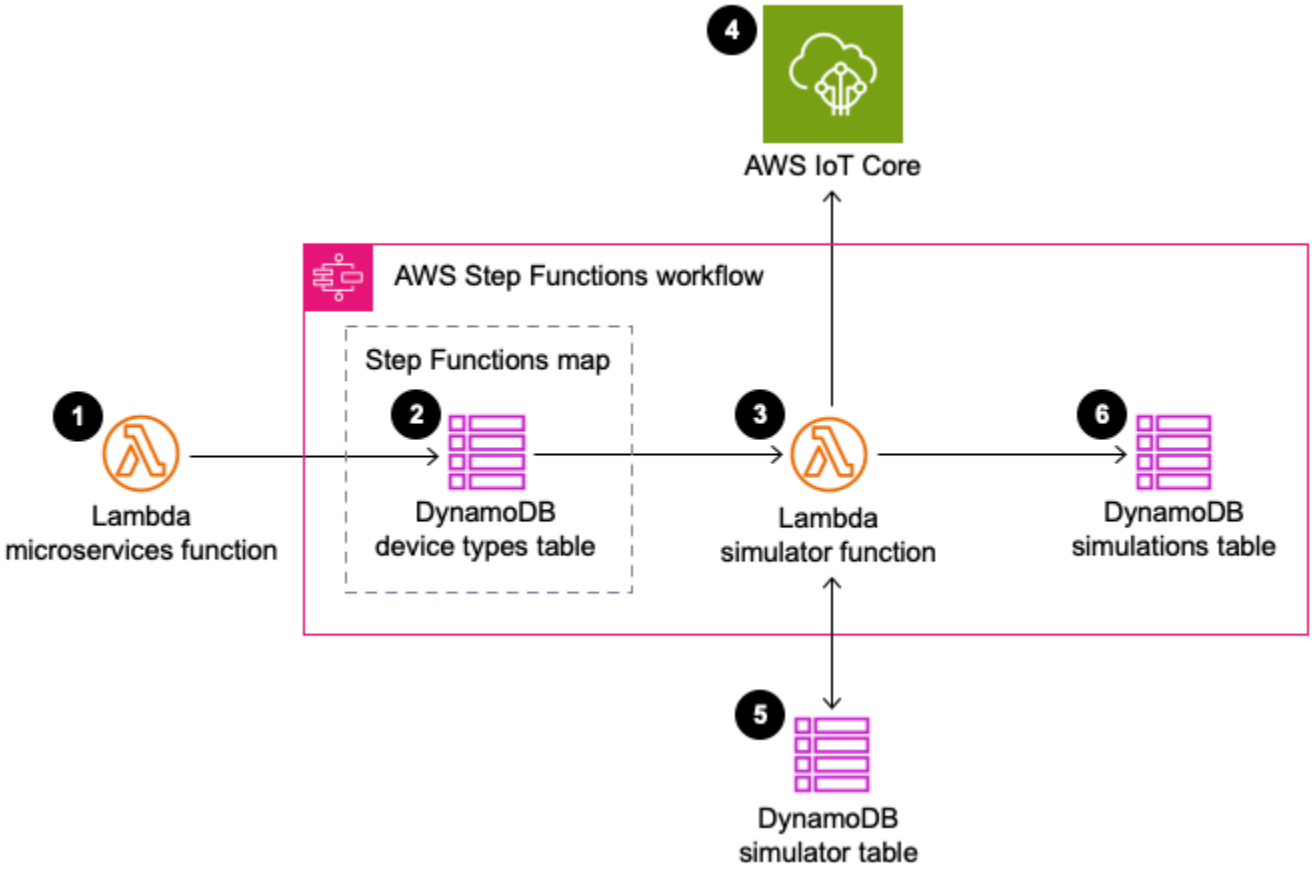
Architecture details

This section describes the components and AWS services that make up this solution and the architecture details on how these components work together.

AWS Step Functions workflow

A Step Functions state machine runs the device simulator. The state machine consists of a Lambda function that provides the logic to create the device messages and send them to the IoT endpoint.

The following detailed breakdown shows the steps involved in the AWS Step Functions state machine when running a simulation.



Simulation workflow

1. The `microservices` AWS Lambda function receives the run simulation request and passes the simulation information to AWS Step Functions.

2. The Lambda function retrieves device type information from the corresponding DynamoDB table for each device type specified in the simulation.
3. The simulation and device type information are passed to the `simulator` Lambda function.
4. The `simulator` Lambda function creates messages and sends them to the AWS IoT Core endpoint. The Lambda function restarts every 15 minutes until it has run for the specified duration.
5. Every 30 seconds, the `simulator` Lambda function polls the `simulator` DynamoDB table to check if the simulation has been stopped externally.
6. When the simulation has finished, the corresponding simulation in the `simulations` DynamoDB table is updated.

Solution microservices

The IoT Device Simulator microservices are a series of Lambda functions that provide the business logic and data access layer for all device operations. This includes create, read, update, and delete (CRUD) operations for the DynamoDB simulation and device type tables, as well as starting the step functions workflow. Each Lambda function assumes an AWS Identity and Access Management (IAM) role with least privilege access (minimum permissions necessary) to perform its designated functions.

Web interface

The solution includes a web interface, which is hosted in Amazon S3 and used to simulate the devices. You can use the interface to create and manage simulations and device types, and start device simulations to simulate devices and send messages to the AWS IoT endpoint.

The interface is designed to simulate devices that publish to an AWS IoT endpoint at regular intervals in order to test backend integration.

> **Note**
>
> Device creation and starting and stopping device simulations are routed through the Amazon API Gateway.

Routes bucket

The solution includes an IoT device simulator automotive demo. An Amazon S3 bucket hosts the pre-defined routes for the automotive demo. The routes are used to provide a pathway for the simulated vehicle to follow. For more details about these routes, see Routes in this guide.

AWS services in this solution

The solution uses the following services. Core services are required to use the solution.

AWS service	Description
Amazon API Gateway	**Core.** Provides endpoints for entire solution.
Amazon CloudFront	**Core.** Serves the web interface content from an Amazon S3 bucket.
Amazon Cognito	**Core.** User pool authenticates the API requests.
AWS IoT Core	**Core.** Serves as the endpoint to which messages are sent.
AWS Lambda	**Core.** Serves as the solution's microservices and routes API requests.
Amazon Location Service	**Core.** Provides the map display showing the location of automotive devices for the automotive demo.
Amazon S3	**Core.** Stores pre-defined routes that are used for the automotive demo.
AWS Step Functions	**Core.** Includes a Lambda simulator function to simulate devices and send messages.
AWS CloudFormation	**Supporting.** Manages deployments for the solution infrastructure.

AWS service	Description
Amazon CloudWatch	**Supporting.** Provides observability into all solution components.
Amazon DynamoDB	**Supporting.** Stores simulation and device type information.
IAM	**Supporting.** IAM manages access permissions between the resources in this solution, such as allowing Lambda functions to read and write to Amazon DynamoDB, publish to an IoT endpoint, read from the Amazon S3 bucket, and start the Step Functions state machine.

Plan your deployment

This section describes the cost, security, Regions, and quota considerations prior to deploying the solution.

Cost

You are responsible for the cost of the AWS services used while running this solution. As of this revision, the estimated cost for running the IoT Device Simulator solution using the 100 automotive demo device types in a single simulation, sending a message every two seconds in the US East (N. Virginia) Region is **$3.05 per month** for a simulation running six hours per day, **$6.11 per month** for a simulation running 12 hours per day, and **$12.22 per month** for a simulation running 24 hours per day. This includes estimated charges for Amazon API Gateway, AWS Lambda, AWS Step Functions, Amazon DynamoDB, and Amazon IoT Core. These costs are for the resources shown in the Sample cost table.

We recommend creating a budget through AWS Cost Explorer to help manage costs. Prices are subject to change. For full details, see the pricing webpage for each AWS services used in this solution.

Sample cost table

The following table provides a sample cost breakdown for deploying this solution with the default parameters in the US East (N. Virginia) Region for one month.

AWS service	6 hours per day	12 hours per day	24 hours per day
Amazon API Gateway	$0.000105	$0.000105	$0.000105
AWS Step Functions	$0.02	$0.04	$0.08
AWS Lambda	$2.70	5.40	$10.80
Amazon DynamoDB	$0.01	$0.02	$0.04
AWS IoT Core messaging	$0.32	$0.65	$1.30

AWS service	6 hours per day	12 hours per day	24 hours per day
Total monthly cost [USD]:	**$3.05***	**$6.11***	**$12.22***

*Cost to run 100 device simulations per month.

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared responsibility model reduces your operational burden because AWS operates, manages, and controls the components including the host operating system, the virtualization layer, and the physical security of the facilities in which the services operate. For more information about AWS security, visit AWS Cloud Security.

IAM roles

IAM roles allow customers to assign granular access policies and permissions to services and users on the AWS Cloud. This solution creates IAM roles that grant the solution's AWS Lambda functions access to read and write to Amazon DynamoDB, publish to an IoT endpoint, read from the Amazon S3 bucket used to host routes, and start the AWS Step Functions state machine.

AWS IoT Core policies

AWS IoT Core policies allow you to control access to the AWS IoT data plane. The AWS IoT data plane consists of operations that allow you to connect to the AWS IoT message broker and send and receive MQ Telemetry Transport (MQTT) messages. The IoT Device Simulator solution creates an AWS IoT policy which allows the web interface to connect to AWS IoT Core, subscribe, and receive MQTT messages.

Amazon CloudFront

This solution deploys a web interface hosted in an Amazon S3 bucket. To help reduce latency and improve security, this solution includes an Amazon CloudFront distribution with an origin access identity, which is a CloudFront user that provides public access to the solution's website bucket contents. For more information, refer to Restricting Access to Amazon S3 Content by Using an Origin Access Identity in the *Amazon CloudFront Developer Guide*.

Amazon API Gateway

This solution deploys an API Gateway REST API and uses the default API endpoint and Secure Sockets Layer (SSL) certificate. The default API endpoint supports only the TLSv1 protocol. To use a later version of Transport Layer Security (TLS), use your own domain name and custom SSL certificate. For more information, refer to Choosing a minimum TLS version for a custom domain in API Gateway in the *Amazon API Gateway Developer Guide*.

Supported AWS Regions

This solution uses Amazon Location Service, which is not currently available in all AWS Regions. For the most current availability of AWS services by Region, see the AWS Regional Services List.

IoT Device Simulator is available in the following AWS Regions:

Region name	
US East (Ohio)	Canada (Central)
US East (N. Virginia)	China (Beijing)
US West (Northern California)	China (Ningxia)
US West (Oregon)	Europe (Frankfurt)
Africa (Cape Town)	Europe (Ireland)
Asia Pacific (Hong Kong)	Europe (London)
Asia Pacific (Hyderabad)	Europe (Paris)
Asia Pacidic (Jakarta)	Europe (Spain)
Asia Pacific (Melbourne)	Europe (Stockholm)
Asia Pacific (Mumbai)	Europe (Zurich)
Asia Pacific (Osaka)	Middle East (Bahrain)
Asia Pacific (Seoul)	Middle East (UAE)

Region name	
Asia Pacific (Singapore)	South America (São Paulo)
Asia Pacific (Sydney)	AWS GovCloud (US-East)
Asia Pacific (Tokyo)	AWS GovCloud (US-West)

Quotas

Service quotas, also referred to as limits, are the maximum number of service resources or operations for your AWS account.

Quotas for AWS services in this solution

Make sure you have sufficient quota for each of the services implemented in this solution. For more information, see AWS service quotas.

Use the following links to go to the page for that service. To view the service quotas for all AWS services in the documentation without switching pages, view the information in the Service endpoints and quotas page in the PDF instead.

Amazon Cognito

This solution uses Amazon Cognito user pools to manage users. Amazon Cognito sends an email every time you create a user, change a password, or reset a password. Amazon Cognito limits the number of emails sent daily per user pool to 50. For customers who plan to use this solution for a large number of users, we recommend using Amazon Simple Email Service (Amazon SES) for these emails. For more information, refer to Email settings for Amazon Cognito user pools in the *Amazon Cognito Developer Guide*.

Simulation limits

A simulation is limited to running up to 100 devices; however, multiple simulations can be run concurrently. The number of simulations that can be run concurrently is limited by the number of Lambda functions concurrently running. For more information on AWS Lambda service quotas, refer to Lambda quotas in the *AWS Lambda Developer Guide*.

Deploy the solution

This solution uses AWS CloudFormation templates and stacks to automate its deployment. The CloudFormation template specifies the AWS resources included in this solution and their properties. The CloudFormation stack provisions the resources that are described in the template.

Deployment process overview

Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Before you launch the solution, review the cost, architecture, network security, and other considerations discussed earlier in this guide.

> **Important**
>
> You cannot update version 2.x or earlier versions of this solution to version 3.x using the AWS CloudFormation console due to changes with how resources are deployed. To use version 3.x, you must launch a new stack using version 3.x of the AWS CloudFormation template. You can uninstall your previous version of this solution.

Time to deploy: Approximately 10 minutes

Step 1: Launch the stack

Step 2. Sign in to the web interface

Step 3. Create the device types

Step 4. Create the simulations

Step 5. Run the simulations

Step 6: Post-deployment configuration

> **Important**
>
> This solution includes an option to send anonymized operational metrics to AWS. We use this data to better understand how customers use this solution and related services and

products. AWS owns the data gathered though this survey. Data collection is subject to the AWS Privacy Notice.
To opt out of this feature, download the template, modify the AWS CloudFormation mapping section, and then use the AWS CloudFormation console to upload your updated template and deploy the solution. For more information, see the Anonymized data collection section of this guide.

AWS CloudFormation template

You can download the CloudFormation template for this solution before deploying it.

View template

iot-device-simulator.template - Use this template to launch the solution and all associated components. The default configuration deploys the core and supporting services found in the AWS services in this solution section, but you can customize the template to meet your specific needs.

> **Note**
>
> AWS CloudFormation resources are created from AWS Cloud Development Kit (AWS CDK) constructs.

Step 1: Launch the stack

Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately 10 minutes

1. Sign in to the AWS Management Console and select the button to launch the `iot-device-simulator` AWS CloudFormation template.

   Launch solution

2. The template launches in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the Region selector in the console navigation bar.

> **Note**
>
> This solution uses Amazon Location Service, which is not currently available in all AWS Regions. You must launch this solution in an AWS Region where Amazon Location Service is available. For the most current availability by Region, refer to the AWS Regional Services List.

3. On the **Create stack** page, verify that the correct template URL is in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack. For information about naming character limitations, see IAM and AWS STS quotas, name requirements, and character limits in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
User email	*<Requires input>*	The email used to sign in to the IoT Device Simulator web interface.

6. Select **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review and create** page, review and confirm the settings. Select the box acknowledging that the template will create IAM resources.
9. Choose **Submit** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a CREATE_COMPLETE status in approximately 10 minutes.

> **Note**
>
> In addition to the primary AWS Lambda functions (`microservices` and `simulator`), this solution also includes the `custom resources helper` Lambda function, which runs only during initial configuration or when resources are updated or deleted.

When you run this solution, you will notice both Lambda functions in the AWS console. Only the `microservices` and `simulator` functions are regularly active. However, you must not delete the `custom resources helper` function, as it is necessary to manage associated resources.

Step 2: Sign in to the web interface

After the AWS CloudFormation stack is created, the resources for the web interface are deployed. You should also receive an email containing the URL for the web interface, your admin credentials, and a temporary password.

Use the following procedure to sign in to the web interface for the first time:

1. Open the email, note your username and temporary password, and select the URL link.
2. On the IoT Device Simulator sign in page, enter the username and temporary password.
3. On the **Change Password** page, enter a new password.

Note

Password requirements: minimum of 12 characters, requiring at least one upper case character, one number, and one symbol.

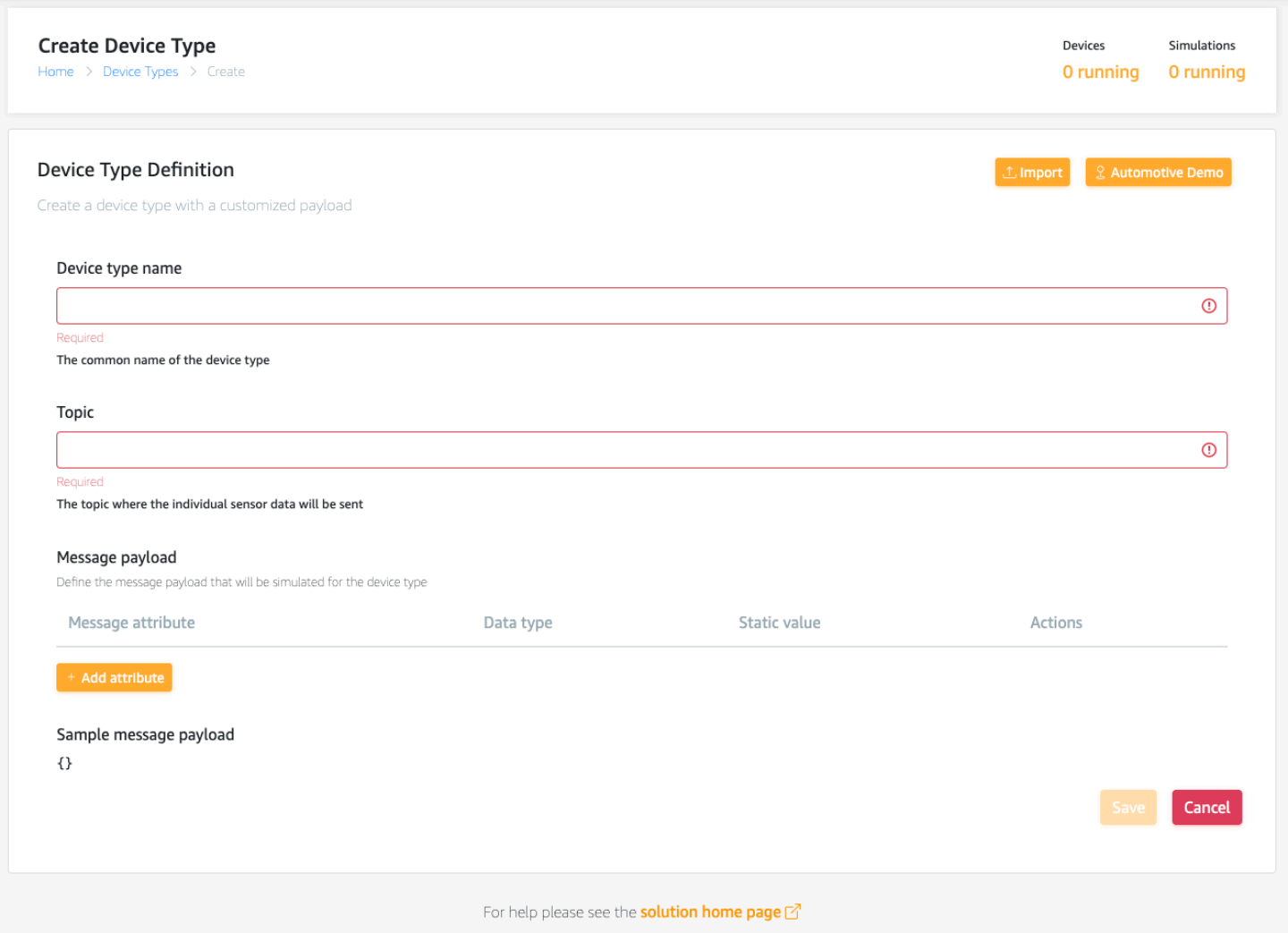
After you sign in to the web interface, follow the remaining steps to create the device types, simulations, and other activities.

Step 3: Create the device types

To define your device types, use the following procedure:

1. Navigate to the **Device Types** page.
2. Choose **Add Device Type**.

On this page, you can either manually add device types or import your device types by uploading a JSON file containing the necessary attributes. To view the structure for the JSON, see the Device Type JSON structure section.



Web interface - create a device type

Use the following procedure to manually create a device type:

1. On the **Create Device Type** page, take the following actions:

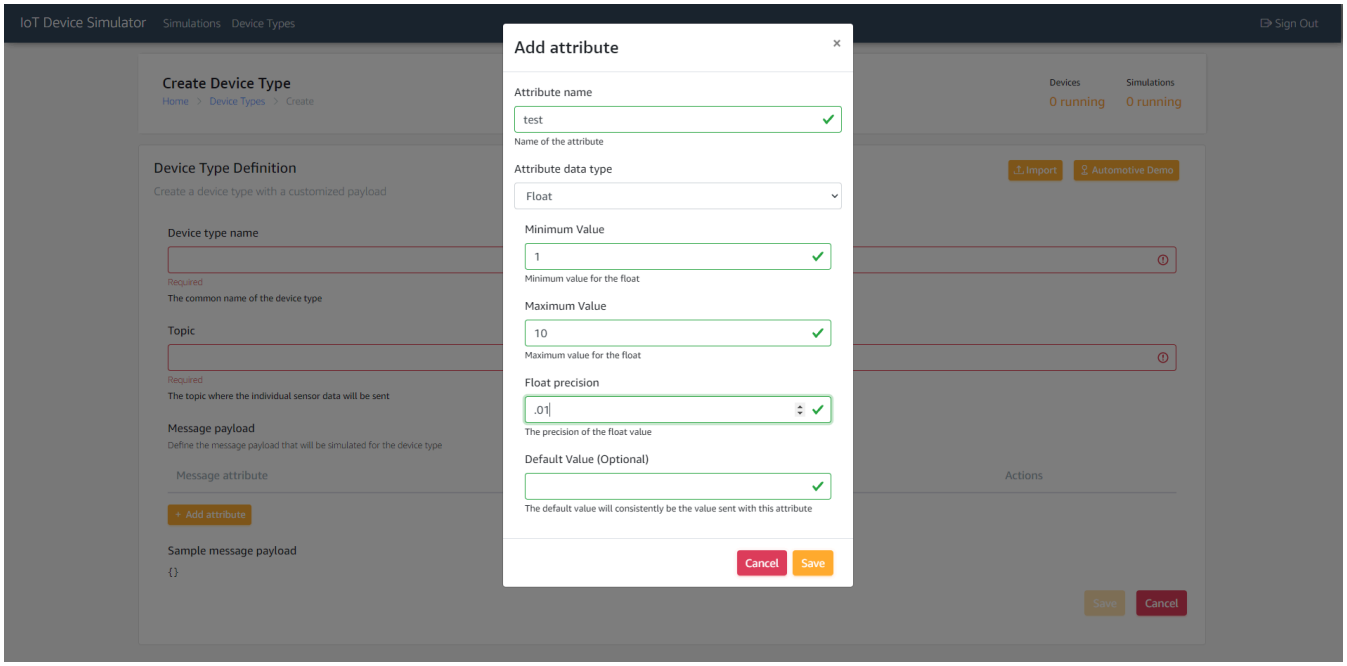
   a. In the **Device type name** field, enter a name.

   b. In the **Topic** field, enter the topic where the device type will publish to.

   c. Under **Message payload**, choose **Add Attribute**.

      You will define the payload of the device type which will serve as a template for the messages that devices of the device type will send.

   d. In the **Add attribute** dialog box, enter an attribute name and complete the additional data fields as needed. You can specify multiple attributes with different data types to structure your payload.



Web Interface – add an attribute

> **Note**
>
> The fields are dynamic based on the Attribute data type that you select.

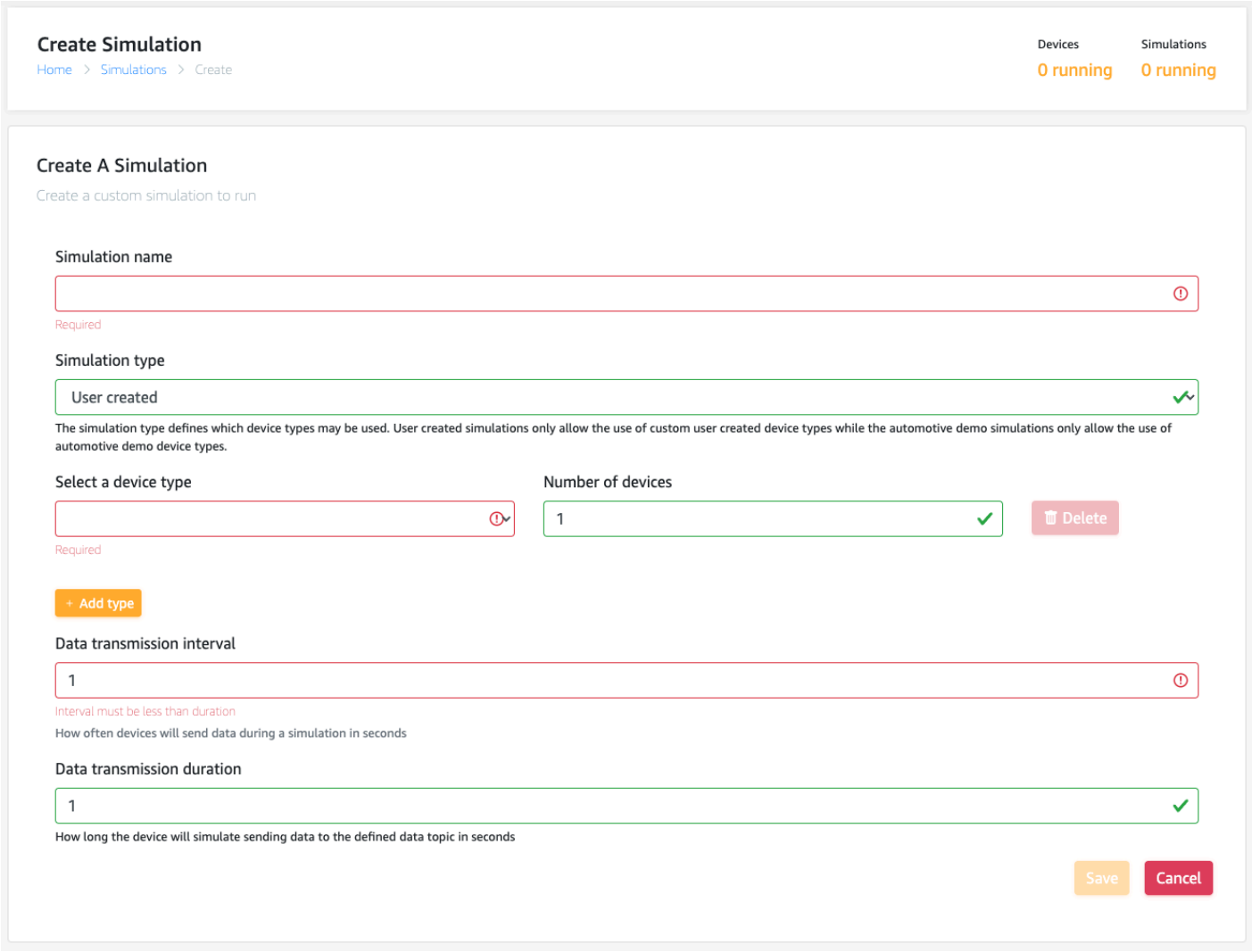
e. Choose **Save**.

2. Enter additional attributes, as needed.
3. Choose **Save**.

Step 4: Create the simulations

To define the simulations to run, use the following procedure:

1. Navigate to the **Simulations** page.
2. Choose **Add Simulation**.
3. On the **Create Simulation** page, take the following actions:



Web interface - create a simulation

a. In the **Simulation name** field, enter a name.

b. In the **Select a device type** drop-down menu, select the device type you want to simulate. To select more than one device type, choose **Add type**.

c. In the **Number of devices** drop-down menu, select the number of devices you want to simulate.

d. In the **Data transmission interval** field, enter the interval time that the devices will send data.

> **Note**
>
> Data transmission interval is a key cost driver of AWS IoT Core messaging expense.

e. In the **Data transmission duration** field, enter the length of time that the simulation will run.

4. Choose **Save**.

Step 5: Run and manage the simulations

To run one or more simulations, use the following procedure:

1. Navigate to the **Simulations** page.
2. On the **Simulations** page, select the checkbox for the simulations you want to run.
3. Choose **Start Simulations** to run one or more simulations.

View the simulations

Use the following procedure to view simulations from the **Simulations** page:

1. Select the **View** button that corresponds to the simulation you want to view. This will take you to the **Simulation Details** page
2. If the simulation is running, you will be able to view the incoming messages in the **Messages** section.

> **Note**
>
> If you are running the automotive simulation demo, you will see a map with the locations of each automotive device.

3. You can filter the messages by device using the filter button or by topic by selecting the corresponding topic.
4. Alternatively, you can also start or stop a simulation from the **Simulation Details** page using the **Start** or **Stop** buttons.

> **Note**
>
> An attribute labeled **_id_** is automatically added to each device in order to identify the device.

Manage the device types

Device types are used to define the type of data your simulated IoT devices will send. The IoT Device Simulator provides a web interface to help you manage your device types, letting you view and edit your device types when needed. You can manage the device types from the **Device Types** page.

On this page, you can view your device types. A list of all device types associated with your account is displayed on this page. You can also review and update the details of a specific device type. To make updates, find the applicable device type and select **Edit**. The **Device Type Edit** page shows the device type definition details including the name, the data topic, and the message payload. To make updates, change the applicable values, and choose **Save**. You can remove existing attributes or add new attributes to the message payload.

Manage the simulations

Simulations define which devices run, how long they run, and at what intervals they send messages to the IoT topic. You can create and delete simulations, view the simulation details, and view the messages a simulation is sending.

To manage your simulations, navigate to the **Simulation** page. A list of all simulations associated with your account are displayed.

- To start one or more simulations, select the checkbox next to each applicable simulation, then choose **Start Simulations**. To select all simulations on the page, use the select all checkbox in the table header.
- To stop simulations, choose **Stop Simulations**.
- To delete a simulation, select the simulation you want to delete and choose **Delete**.

Step 6: Post-deployment configuration

This section provides recommendations for configuring the solution after deployment.

Amazon CloudWatch dashboard and alarms

We recommend creating CloudWatch alarms and adding notifications based on the use case. In addition, you can create a CloudWatch dashboard to monitor your resources in a single view.

Additional security considerations

We recommend reviewing the following additional security best practices:

- **Customer managed AWS KMS encryption keys** - The solution uses AWS managed AWS KMS keys by default because these are available at no additional cost. Review your use case to determine if you need to update the solution to use customer managed AWS KMS keys.
- **Activate AWS CloudTrail** - As a recommended security practice, consider activating AWS CloudTrail in the AWS account where the solution is deployed to log API calls in the AWS account. For more details, see the AWS CloudTrail User Guide.
- **Activate AWS WAF** – As an additional security measure, activate AWS WAF to protect against common web exploits and bots that can affect availability, compromise security, or consume excessive resources.

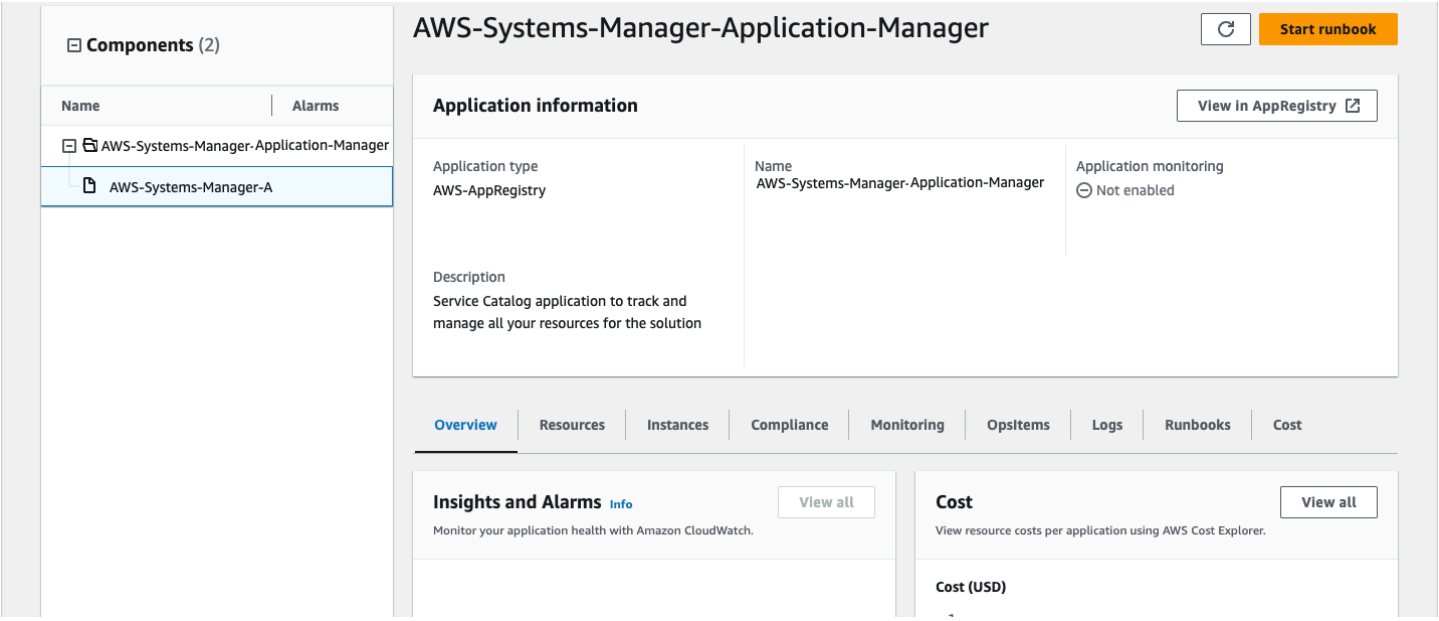
Monitor the solution with Service Catalog AppRegistry

The solution includes a Service Catalog AppRegistry resource to register the CloudFormation template and underlying resources as an application in both Service Catalog AppRegistry and AWS Systems Manager Application Manager.

AWS Systems Manager Application Manager gives you an application-level view into this solution and its resources so that you can:

- Monitor its resources, costs for the deployed resources across stacks and AWS accounts, and logs associated with this solution from a central location.
- View operations data for the resources of this solution in the context of an application. For example, deployment status, CloudWatch alarms, resource configurations, and operational issues.

The following figure depicts an example of the application view for the solution stack in Application Manager.



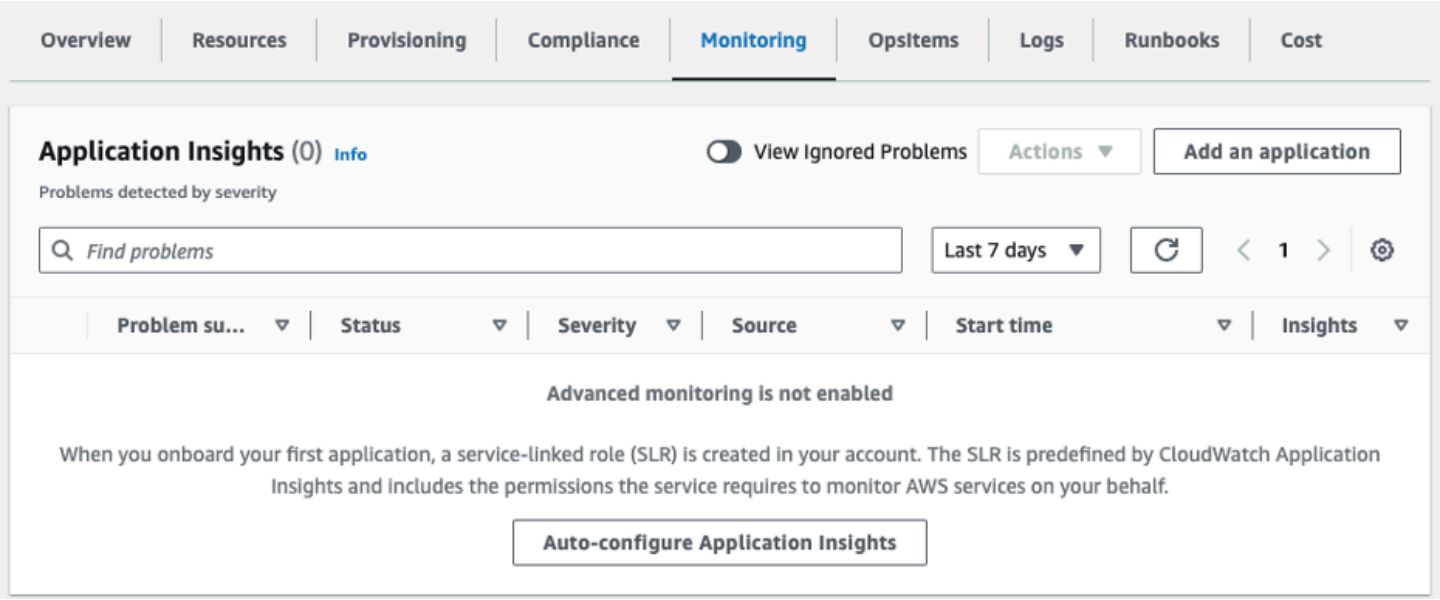
Solution stack in Application Manager

Activate CloudWatch Application Insights

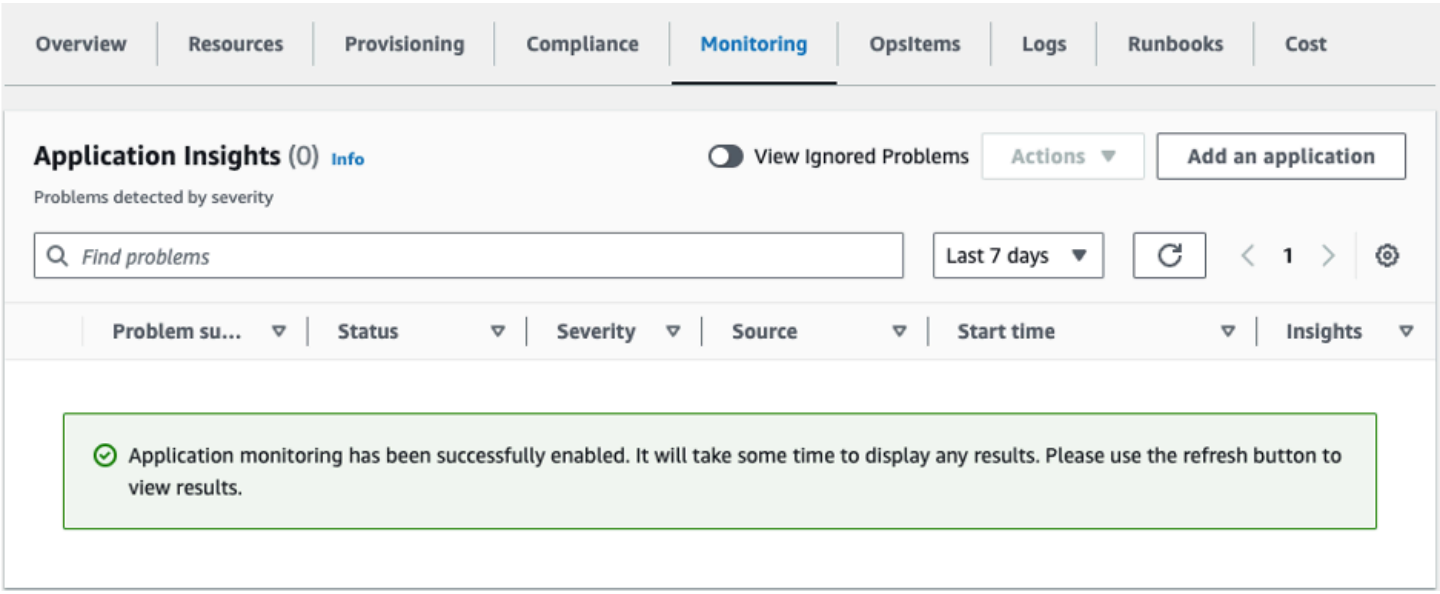
1. Sign in to the Systems Manager console.

2. In the navigation pane, choose **Application Manager**.
3. In **Applications**, search for the application name for this solution and select it.

   The application name will have **App Registry** in the **Application Source** column, and will have a combination of the solution name, Region, account ID, or stack name.
4. In the **Components** tree, choose the application stack you want to activate.
5. In the **Monitoring** tab, in **Application Insights**, select **Auto-configure Application Insights**.



Monitoring for your applications is now activated and the following status box appears:

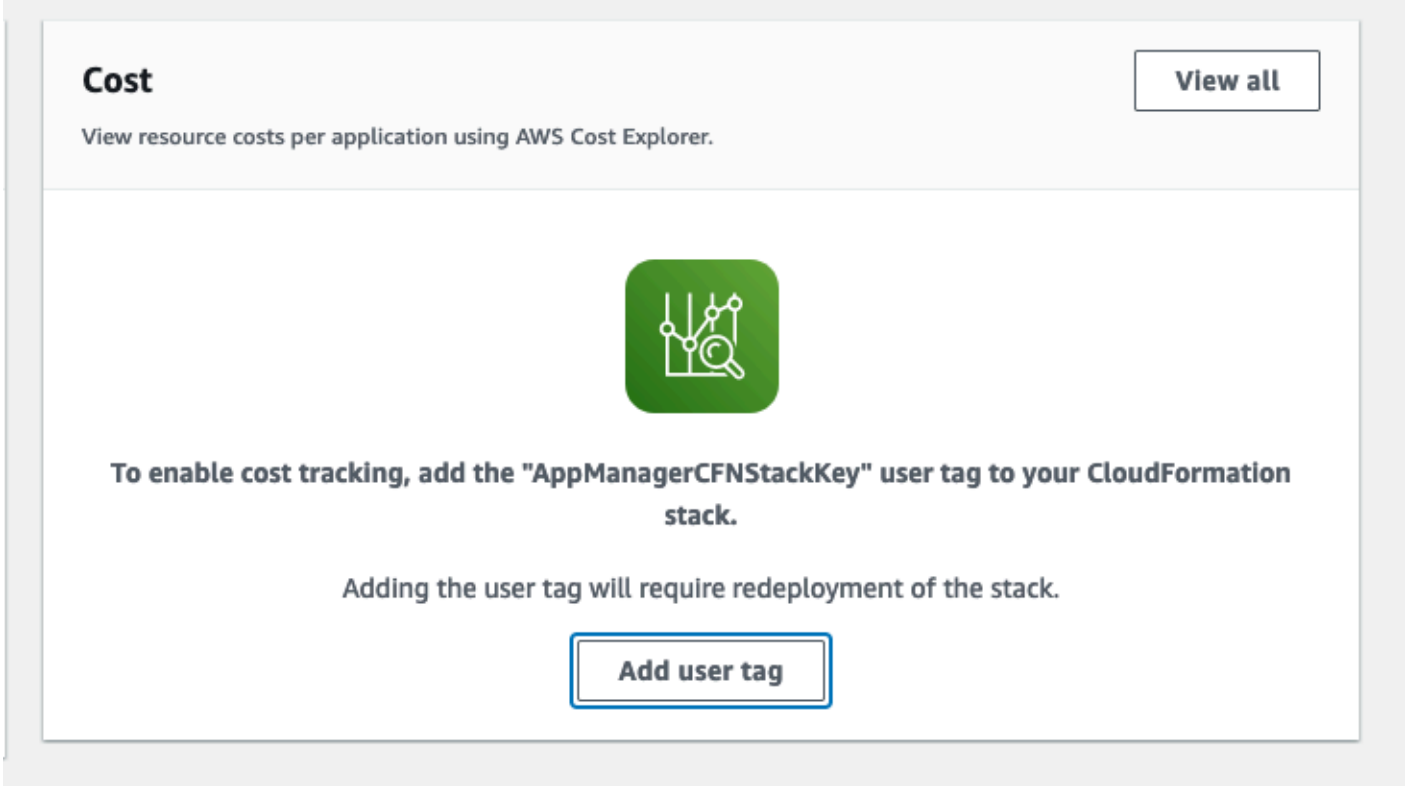


Confirm cost tags associated with the solution

After you activate cost allocation tags associated with the solution, you must confirm the cost allocation tags to see the costs for this solution. To confirm cost allocation tags:

1. Sign in to the Systems Manager console.
2. In the navigation pane, choose **Application Manager**.
3. In **Applications**, choose the application name for this solution and select it.

   The application name will have **App Registry** in the **Application Source** column, and will have a combination of the solution name, Region, account ID, or stack name.
4. In the **Overview** tab, in **Cost**, select **Add user tag**.



5. On the **Add user tag** page, enter `confirm`, then select **Add user tag**.

The activation process can take up to 24 hours to complete and the tag data to appear.

Activate cost allocation tags associated with the solution

After you activate Cost Explorer, you must activate the cost allocation tags associated with this solution to see the costs for this solution. The cost allocation tags can only be activated from the management account for the organization. To activate cost allocation tags:

1. Sign in to the AWS Billing and Cost Management and Cost Management console.
2. In the navigation pane, select **Cost Allocation Tags**.
3. On the **Cost allocation tags** page, filter for the AppManagerCFNStackKey tag, then select the tag from the results shown.
4. Choose **Activate**.

AWS Cost Explorer

You can see the overview of the costs associated with the application and application components within the Application Manager console through integration with AWS Cost Explorer, which must be first activated. Cost Explorer helps you manage costs by providing a view of your AWS resource costs and usage over time. To activate Cost Explorer for the solution:

1. Sign in to the AWS Cost Management console.
2. In the navigation pane, select **Cost Explorer** to view the solution's costs and usage over time.

Update the solution

This implementation guide contains information about how to set up and configure IoT Device Simulator version 3.x. You cannot update version 2.x or earlier versions of this solution to version 3.x using the AWS CloudFormation console due to changes with how resources are deployed. To use version 3.x, you must launch a new stack using version 3.x of the AWS CloudFormation template. You can uninstall your previous version of this solution.

> **Note**
>
> If you have device types saved in a previous version of this solution, we recommend recreating these types using the web interface.

If you have previously deployed an older version 3.x of the solution, follow this procedure to update the solution's CloudFormation stack to get the latest version of the solution's framework.

1. Sign in to the CloudFormation console, select your existing IoT Device Simulator CloudFormation stack, and select **Update**.
2. Select **Replace current template**.
3. Under Specify template:
   a. Select Amazon S3 URL.
   b. Copy the link of the `iot-device-simulator.template` the section called "AWS CloudFormation template".
   c. Paste the link in the **Amazon S3 URL** box.
   d. Verify that the correct template URL shows in the **Amazon S3 URL** text box, and choose **Next**. Choose **Next** again.
4. Under **Parameters**, review the parameters for the template and modify them as necessary. For details about the parameters, see Step 1. Launch the Stack.
5. Choose **Next**.
6. On the **Configure stack options** page, choose **Next**.
7. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create IAM resources.
8. Choose **View change set** and verify the changes.

9. Choose **Update stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a UPDATE_COMPLETE status in approximately 10 minutes.

Troubleshooting

If you need help with this solution, contact AWS Support to open a support case for this solution.

Contact AWS Support

If you have AWS Developer Support, AWS Business Support, or AWS Enterprise Support, you can use the Support Center to get expert assistance with this solution. The following sections provide instructions.

Create case

1. Sign in to Support Center.
2. Choose **Create case**.

How can we help?

1. Choose **Technical**.
2. For **Service**, select **Solutions**.
3. For **Category**, select **Other Solutions**.
4. For **Severity**, select the option that best matches your use case.
5. When you enter the **Service**, **Category**, and **Severity**, the interface populates links to common troubleshooting questions. If you can't resolve your question with these links, choose **Next step: Additional information**.

Additional information

1. For **Subject**, enter text summarizing your question or issue.
2. For **Description**, describe the issue in detail.
3. Choose **Attach files**.
4. Attach the information that AWS Support needs to process the request.

Help us resolve your case faster

1. Enter the requested information.
2. Choose **Next step: Solve now or contact us**.

Solve now or contact us

1. Review the **Solve now** solutions.
2. If you can’t resolve your issue with these solutions, choose **Contact us**, enter the requested information, and choose **Submit**.

Uninstall the solution

You can uninstall the IoT Device Simulator solution from the AWS Management Console or by using the AWS Command Line Interface. You must manually delete the Amazon S3 buckets, and Amazon DynamoDB tables created by this solution. AWS Solutions Implementations do not automatically delete these resources in case you have stored data to retain.

Using the AWS Management Console

1. Sign in to the CloudFormation console.
2. On the **Stacks** page, select this solution's installation stack.
3. Choose **Delete**.

Using AWS Command Line Interface

Determine whether the AWS Command Line Interface (AWS CLI) is available in your environment. For installation instructions, see What Is the AWS Command Line Interface in the *AWS CLI User Guide*. After confirming that the AWS CLI is available, run the following command:

```
$ aws cloudformation delete-stack --stack-name <installation-stack-name>
```

Deleting the Amazon S3 buckets

This solution is configured to retain the solution-created Amazon S3 bucket (for deploying in an opt-in Region) if you decide to delete the AWS CloudFormation stack to prevent accidental data loss. After uninstalling the solution, you can manually delete this S3 bucket if you do not need to retain the data. Follow these steps to delete the Amazon S3 bucket.

1. Sign in to the Amazon S3 console.
2. Choose **Buckets** from the left navigation pane.
3. Locate the *<stack-name>* S3 buckets.
4. Select the S3 bucket and choose **Delete**.

To delete the S3 bucket using AWS CLI, run the following command:

```
$ aws s3 rb s3://<bucket-name> --force
```

Deleting DynamoDB tables

This solution is configured to retain the solutions Amazon DynamoDB tables if you decide to delete the AWS CloudFormation stack to prevent accidental data loss. After uninstalling the solution, you can manually delete the Amazon DynamoDB tables if you do not need to retain the data. Follow these steps to delete the Amazon DynamoDB tables.

1. Sign in to the Amazon DynamoDB console.
2. Choose **Tables** from the left navigation pane.
3. Select the *<stack-name>* Amazon DynamoDB table you want to delete and choose **Delete table**.

Automotive demo

Routes

The solution includes an Amazon S3 bucket to store pre-defined routes for the automotive demo. The routes are used to define the path the vehicle takes, and some routes include random triggers that could arise throughout the route, such as high oil temperature. The route locations are defined in the stages. Each stage contains a start and end position. The automotive demo device moves along the stages based on various calculations, such as the speed of the device. The latitude and longitude of the device is reflective of which stage the device is currently navigating. The device location updates each time it moves between a stage. It might take multiple messages before a device moves from one stage to the next.

Running the automotive demo

The solution contains an automotive demo which simulates a connected vehicle. The automotive demo uses one of 18 defined routes that exist in an Amazon S3 bucket that is created when the solution is launched. It then uses various calculations in the `simulator` Lambda function to simulate data such as fuel consumption, vehicle speed, acceleration and more. With the automotive demo simulation, you can view a map with the location of the running devices.

To use the automotive demo, complete the following steps:

1. Create an Automotive Demo device type.
   a. Navigate to the **Device Types** page.
   b. Choose **Create Device Type**.
   c. Choose **Automotive Demo**.
   d. The payload is auto populated. Enter the rest of the fields such as **Name** and **Topic**.
   e. Choose **Save**.
2. Create a Simulation.
   a. Navigate to the **Simulations** page.
   b. Choose **Create Simulation**.
   c. Change the **Simulation Type** field to **Automotive**.
   d. In the dropdown field for devices, you can view your automotive demo device types.

e. Enter the required fields.

f. Choose **Save**.

3. Run the simulation.

   a. Run the simulation from the Simulations page by checking the desired simulations then choose **Start Simulations**.

   b. Alternatively, choose **View** next to the simulation you want to run, then choose **Start** to run the simulation.

4. View the simulation.

   a. Choose **View** next to the simulation you want to view.

   b. If the simulation is running, you can view a map with the locations of the devices, and up to 100 of the most recent messages sent to the IoT topic.

Developer guide

This section provides the source code for the solution and JSON structure for importing devices into the simulator.

Source code

Visit our GitHub repository to download the source files for this solution and to share your customizations with others. The IoT Device Simulator templates are generated using the AWS CDK. See the README.md file for additional information.

Device type JSON structure

The structure of the JSON to import a device type should contain the following:

Parameter	Required	Type	Description
name	Yes	String	The name of the device type.
topic	Yes	String	The topic to which the device type sends its messages.
payload	Yes	Array of attribute objects	The payload of the device type. See the attribute structure table below to see valid payload contents.

General attribute parameters that are required in all attribute objects:

Parameter	Required	Type	Description
name	Yes	String	The name of the attribute.
type	Yes	String	The type of attribute , must be one of the values listed in the attribute table below.

Parameters that are specific to each attribute:

Type	Parameters			
id	**Name**	**Required**	**Type**	**Description**
	charSet	No	String	An alphabet to limit the characters that will be used in the ID.
	length	No	Number	The length of the ID.
	static	No	Boolean	If true, the value will only be generated once per simulation.
bool	**Name**	**Required**	**Type**	**Description**
	default	No	Number	A default value to be used instead of generating a value.

Type	**Parameters**			
decay	**Name**	**Required**	**Type**	**Description**
	min	Yes	Number	The floor for the decay.
	max	Yes	Number	The starting value for the decay.
	default	No	Number	The static value to be used.
float	**Name**	**Required**	**Type**	**Description**
	min	Yes	Number	The minimum value to be generated.
	max	Yes	Number	The maximum value to be generated.
	precision	Yes	Number	The decimal precision of the float (for example, .01).
	default	No	Number	A default value to be used instead of generating a value.
int	**Name**	**Required**	**Type**	**Description**

Type	Parameters			
	min	Yes	Number	The minimum of the range from which a number will be generated.
	max	Yes	Number	The maximum of the range form which a number will be generated.
	default	No	Number	A default value to be used rather than have one generated.
location	**Name**	**Required**	**Type**	**Description**
	lat	Yes	Number	The center position latitude.
	long	Yes	Number	The center position longitude.
	radius	Yes	Number	The radius (in meters) from the center position for the random coordinates to be generated.
object	**Name**	**Required**	**Type**	**Description**

Type	**Parameters**			
	payload	Yes	Array	An array of attribute objects.
string	**Name**	**Required**	**Type**	**Description**
	min	Yes	Number	The minimum length of the string.
	max	Yes	Number	The maximum length of the string.
	static	No	Boolean	If true, the value will be generated once per simulation.
	default	No	String	A default value to be used instead of generating a value.
sinusoidal	**Name**	**Required**	**Type**	**Description**
	min	Yes	Number	The minimum value.
	max	Yes	Number	The maximum value.

Type	Parameters			
	default	No	Number	A default value to be used instead of generating a value.
timestamp	**Name**	**Required**	**Type**	**Description**
	tsformat	Yes	String	The timestamp format to be used. Must be one of the following: `default` or `unix`.
	default	No	Number	A default value to be used instead of generating a value.
pickOne	**Name**	**Required**	**Type**	**Description**
	arr	Yes	Array	An array of strings from which a value will be chosen.
	static	No	Boolean	If true, the value will be generated only once per simulation.

Reference

This section includes information about an optional feature for collecting unique metrics for this solution and a list of builders who contributed to this solution.

Anonymized data collection

This solution includes an option to send anonymized operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When invoked, the following information is collected and sent to AWS:

- **Solution ID** - The AWS solution identifier
- **Unique ID (UUID)** - Randomly generated, unique identifier for each IoT Device Simulator deployment
- **Timestamp** - Data-collection timestamp
- **Device Type Data** - The type of attributes used when defining a device type payload

  Example data:

```
{
  eventType: "create device type"
  uniquePayloadAttrs: [string, float, sinusoidal]
}
```

- **Simulation Creation Data** - The number of devices, and the duration of the simulation

  Example data:

```
{
  eventType: "create simulation",
  duration: 120,
  numDevices: 70
}
```

- **Simulation Run Data** - The number of devices, and the duration of the simulation

  Example data:

```
{
```

```
  eventType: "start simulation",
  duration: 120,
  numSimulations: 2
  type: 'autoDemo'
}
```

AWS owns the data gathered though this survey. Data collection is subject to the Privacy Notice. To opt out of this feature, complete the following steps before launching the AWS CloudFormation template.

1. Download the `iot-device-simulator.template` the section called "AWS CloudFormation template" to your local hard drive.
2. Open the AWS CloudFormation template with a text editor.
3. Modify the AWS CloudFormation template mapping section from:

   ```
   SendAnonymousUsage: "Yes"
   ```

   to:

   ```
   SendAnonymousUsage: "No"
   ```

4. Sign in to the AWS CloudFormation console.
5. Select Create stack.
6. On the Create stack page, Specify template section, select Upload a template file.
7. Under **Upload a template file**, choose **Choose file** and select the edited template from your local drive.
8. Choose **Next** and follow the steps in Launch the stack in the Deploy the solution section of this guide.

Contributors

- George Lenz
- Ajay Swamy
- Manish Jangid
- Abhishek Patil

- Alex Sansone

Revisions

Date	Change
May 2018	Initial release
December 2018	Added information about the Amazon CloudFront distribution for the static website hosted in the Amazon S3 bucket
March 2019	Added information about Amazon DynamoDB on-demand, the Amazon ECS service-linked role, additional device type attributes and functionality, and managing device types, widgets, and users
December 2019	Added information on support for Node.js update
July 2020	Added cost considerations for Amazon ECS; updated information on support for Node.js update and AWS Lambda
November 2021	Release v3.0.0: Used AWS Cloud Developme nt Kit to create the AWS CloudFormation template; migrated UI to React and simplifie d the UI; added the ability to import/export device types; changed simulator from running on Amazon ECS to AWS Lambda with AWS Step Functions; removed widgets and changed to a device type/simulation workflow to run devices; added Amazon Location Service as map provider; changed the UI and custom resource Lambda to Typescript; and aggregate d automotive demo messages. For more information, refer to the CHANGELOG.md file in the GitHub repository.

Date	Change
April 2023	Release v3.0.1: Upgraded to Node.js 18, added AppRegistry integration, upgraded to AWS CDK v2, upgraded UI build system to React Scripts 5, upgraded to Axios1, and upgraded to use ES2022 for all TypeScript modules. Additionally, mitigated impact caused by new default settings for S3 Object Ownership (ACLs disabled) for all new S3 buckets. For more information, refer to the CHANGELOG.md file in the GitHub repository.
June 2023	Release v3.0.2: Fixed fast-xml-parser vulnerability, and added deployment details in README.md. For more information, refer to the CHANGELOG.md file in the GitHub repository.
August 2023	Release v3.0.3: Upgraded AWS Amplify from v4.x to v5.x, migrated AWS SDK for JavaScript from v2.x to v3.x, library version updates, security patches, UI bug fixes, and removed CDK bootstrap requirement for provisioning the CloudFormation stack. For more information, refer to the CHANGELOG.md file in the GitHub repository. Added information to documentation about CloudWatch alarms creation and additional security recommendations.
October 2023	Release v3.0.4: Updated package versions to resolve security vulnerabilities. For more information, refer to the CHANGELOG.md file in the GitHub repository.

Date	Change
November 2023	Documentation update: Added Confirm cost tags associated with the solution to the Monitoring the solution with Service Catalog AppRegistry section.
February 2024	Release v3.0.5: CDK updates. For more information, refer to the CHANGELOG.md file in the GitHub repository.
April 2024	Release v3.0.6: Updated package versions to resolve security vulnerabilities. For more information, refer to the CHANGELOG.md file in the GitHub repository.
April 2024	Documentation update: Migration to new template to address customer feedback.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents AWS current product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers, or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. AWS responsibilities and liabilities to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

IoT Device Simulator is licensed under the terms of the Apache License Version 2.0.