# Table of Contents

What Is AWS IoT Analytics? ............................................................................................................. 1
   Why Use AWS IoT Analytics? ........................................................................................................ 1
      Benefits ................................................................................................................................... 1
      Use Cases ............................................................................................................................... 2
      Key Features .......................................................................................................................... 2
   How to Use AWS IoT Analytics .................................................................................................. 4
   AWS IoT Analytics Components ................................................................................................. 4
      Accessing AWS IoT Analytics ............................................................................................... 5
   AWS IoT Analytics Service Limits ............................................................................................. 5
      About Amazon Web Services .................................................................................................. 6

Getting Started with AWS IoT Analytics .......................................................................................... 7
   Create a Data Store ..................................................................................................................... 7
   Create a Channel ....................................................................................................................... 7
   Create a Pipeline ....................................................................................................................... 7
   Create an IAM Role .................................................................................................................... 8
   Create an IoT Rule ...................................................................................................................... 9
   Send an IoT Message .................................................................................................................. 10
   Query Your Data ...................................................................................................................... 12
   Access the Query Results .......................................................................................................... 12
   Explore Your Data .................................................................................................................... 13

AWS IoT Analytics Console Quickstart Guide ................................................................................. 14
   Sign in to the AWS IoT Analytics Console ............................................................................... 14
   Create a Channel ....................................................................................................................... 14
   Create a Data store .................................................................................................................... 17
   Create a Pipeline ....................................................................................................................... 18
   Create a Data Set ....................................................................................................................... 21
   Send an AWS IoT Message ....................................................................................................... 23
   Access the Query Results .......................................................................................................... 25
   Explore Your Data .................................................................................................................... 28
   Notebook Templates .................................................................................................................. 30

Pipeline Activities .......................................................................................................................... 31
   Channel Activity ....................................................................................................................... 31
   Datastore Activity ...................................................................................................................... 31
   Lambda Activity ......................................................................................................................... 31
   AddAttributes Activity .............................................................................................................. 33
   RemoveAttributes Activity ....................................................................................................... 34
   SelectAttributes Activity .......................................................................................................... 35
   Filter Activity ............................................................................................................................ 35
   DeviceRegistryEnrich Activity ................................................................................................. 36
   DeviceShadowEnrich Activity ................................................................................................. 37
   Math Activity ............................................................................................................................. 39
   RunPipelineActivity Example ................................................................................................... 39

Reprocessing Channel Data ........................................................................................................... 41
   SQL Support .............................................................................................................................. 42

Visualizing AWS IoT Analytics Data with QuickSight .................................................................... 44

AWS IoT Analytics Troubleshooting Guide ..................................................................................... 46

Logging AWS IoT Analytics API Calls with CloudTrail ................................................................. 49
   AWS IoT Analytics Information in CloudTrail ........................................................................ 49
   Understanding AWS IoT Analytics Log File Entries .................................................................. 50

Setting Up CloudWatch Logs ......................................................................................................... 53
   Create a Logging Role .............................................................................................................. 53
   Configure and Enable Logging ............................................................................................... 54
      PutLoggingOptions ............................................................................................................... 54
      DescribeLoggingOptions ...................................................................................................... 56
### Namespace, Metrics and Dimensions

- Tagging Your AWS IoT Analytics Resources
  - Tag Basics
  - Tag Restrictions

### AWS IoT Analytics Commands

- BatchPutMessage
- CancelPipelineReprocessing
- CreateChannel
- CreateDataset
- CreateDatasetContent
- CreateDatastore
- CreatePipeline
- DeleteChannel
- DeleteDataset
- DeleteDatasetContent
- DeleteDatastore
- DeletePipeline
- DescribeChannel
- DescribeDataset
- DescribeDatastore
- DescribeLoggingOptions
- DescribePipeline
- GetDatasetContent
- ListChannels
- ListDatasets
- ListDatastores
- ListPipelines
- ListTagsForResource
- PutLoggingOptions
- RunPipelineActivity
- SampleChannelData
- StartPipelineReprocessing
- TagResource
- UntagResource
- UpdateChannel
- UpdateDataset
- UpdateDatastore
- UpdatePipeline
What Is AWS IoT Analytics?

AWS IoT Analytics allows you to collect large amounts of device data, process messages, and store them. You can then query the data and run sophisticated analytics on it. AWS IoT Analytics enables advanced data exploration through integration with Jupyter Notebooks and data visualization through integration with Amazon QuickSight.

Traditional analytics and business intelligence tools are designed to process structured data. IoT data often comes from devices that record noisy processes (such as temperature, motion, or sound). As a result, the data from these devices can have significant gaps, corrupted messages, and false readings that must be cleaned up before analysis can occur. Also, IoT data is often only meaningful in the context of other data from external sources.

AWS IoT Analytics automates the steps required to analyze data from IoT devices. AWS IoT Analytics filters, transforms, and enriches IoT data before storing it in a time-series data store for analysis. You can set up the service to collect only the data you need from your devices, apply mathematical transforms to process the data, and enrich the data with device-specific metadata such as device type and location before storing it. Then, you can analyze your data by running queries using the built-in SQL query engine, or perform more complex analytics and machine learning inference. AWS IoT Analytics includes pre-built models for common IoT use cases so you can answer questions like which devices are about to fail or which customers are at risk of abandoning their wearable devices.

Why Use AWS IoT Analytics?

Benefits

Run Queries on IoT Data

With AWS IoT Analytics, you can run simple, ad-hoc queries using the built-in AWS IoT Analytics SQL query engine. The service allows you to use standard SQL queries to extract data from the data store to answer questions like the average distance traveled for a fleet of connected vehicles or how many doors are locked after 7 P.M. in a smart building. These queries can be re-used even if connected devices, fleet size, and analytic requirements change.

Run Time-Series Analytics

AWS IoT Analytics also supports time-series analyses so you can analyze the performance of devices over time and understand how and where they are being used, continuously monitor device data to predict maintenance issues, and monitor sensors to predict and react to environmental conditions.

Data Storage Optimized for IoT

AWS IoT Analytics stores the processed device data in a time-series data store that is optimized to deliver fast response times on IoT queries that typically include time as a criteria. The raw data is also automatically stored for later processing or to reprocess it for another use case.

Prepares Your IoT Data for Analysis

AWS IoT Analytics includes data preparation techniques that allow you to prepare and process your data for analysis. AWS IoT Analytics is integrated with AWS IoT Core so it is easy to ingest device data directly from connected devices. It can clean false readings, fill gaps in the data, and perform mathematical transformations of message data. As the data is ingested, AWS IoT Analytics can process it using conditional statements, filter data to collect just the data you want to analyze, and enrich it with information from the AWS IoT Registry. You can also use AWS Lambda functions to
enrich your device data from external sources like the Weather Service, HERE Maps, Salesforce, or Amazon DynamoDB.

**Tools for Machine Learning**

AWS IoT Analytics allows you to apply machine learning to your IoT data with hosted Jupyter Notebooks. You can directly connect your IoT data to the notebook and build, train, and execute models right from the AWS IoT Analytics console without having to manage any of the underlying infrastructure. Using AWS IoT Analytics, you can apply machine learning algorithms to your device data to produce a health score for each device in your fleet.

**Use Cases**

**Smart Agriculture**

AWS IoT Analytics can enrich IoT device data with contextual metadata using AWS IoT Registry data or public data sources so that your analysis factors in time, location, temperature, altitude, and other environmental conditions. With that analysis, you can write models that output recommended actions for your devices to take in the field. For example, to determine when to water, irrigation systems might enrich humidity sensor data with data on rainfall, allowing more efficient water usage.

**Predictive Maintenance**

AWS IoT Analytics provides templates to build predictive maintenance models and apply them to your devices. For example, you can use AWS IoT Analytics to predict when heating and cooling systems will fail on connected cargo vehicles so the vehicles can be rerouted to prevent shipment damage. Or, an auto manufacturer can detect which of its customers have worn brake pads and alert them to seek maintenance for their vehicles.

**Proactive Replenishing of Supplies**

AWS IoT Analytics lets you build IoT applications that can monitor inventories in real time. For example, a food and drink company can analyze data from food vending machines and proactively reorder merchandise whenever the supply is running low.

**Process Efficiency Scoring**

With AWS IoT Analytics, you can build applications that constantly monitor the efficiency of different processes and take action to improve the process. For example, a mining company can increase the efficiency of its ore trucks by maximizing the load for each trip. With AWS IoT Analytics, the company can identify the most efficient load for a location or truck over time, and then compare any deviations from the target load in real time, and better plan loading guidelines to improve efficiency.

**Key Features**

**Collect**

- **Integrated with AWS IoT Core** – AWS IoT Analytics is fully integrated with AWS IoT Core so it can receive messages from connected devices as they stream in.
- **Use a batch API to add data from any source** – AWS IoT Analytics can receive data from any source through HTTP. That means that any device or service that is connected to the internet can send data to AWS IoT Analytics. ([BatchPutMessage](p. 61))
- **Collect only the data you want to store and analyze** – You can use the AWS IoT Analytics console to configure AWS IoT Analytics to receive messages from devices through MQTT topic filters in various formats and frequencies. AWS IoT Analytics validates that the data is within specific parameters you define and creates channels. Then, the service routes the channels to appropriate pipelines for message processing, transformation, and enrichment.
Key Features

Process

- **Cleanse and filter** – AWS IoT Analytics lets you define AWS Lambda functions that are triggered when AWS IoT Analytics detects missing data, so you can run code to estimate and fill gaps. You can also define max/min filters and percentile thresholds to remove outliers in your data.

- **Transform** – AWS IoT Analytics can transform messages using mathematical or conditional logic you define, so you can perform common calculations like Celsius into Fahrenheit conversion.

- **Enrich** – AWS IoT Analytics can enrich data with external data sources such as a weather forecast, and then route the data to the AWS IoT Analytics data store.

Store

- **Time-Series Data Store** - AWS IoT Analytics stores the device data in an IoT optimized time-series data store for analysis. You can manage access permissions, implement data retention policies and export your data to external access points.

- **Store Processed and Raw Data** - AWS IoT Analytics stores the processed data. It also automatically stores the raw ingested data so you can process it at a later time.

Analyze

- **Run Ad hoc SQL Queries** - AWS IoT Analytics provides a SQL query engine so you can run ad-hoc queries and get results quickly. For example, you might want to run a quick query to find the number of active users for each device in your fleet.

- **Time-Series Analysis** - AWS IoT Analytics supports time-series analysis so you can analyze the performance of devices over time and understand how and where they are being used, continuously monitor device data to predict maintenance issues, and monitor sensors to predict and react to environmental conditions.

- **Hosted Notebooks for Sophisticated Analytics and Machine Learning** - AWS IoT Analytics includes support for hosted notebooks in Jupyter Notebooks for statistical analysis and machine learning. The service includes a set of notebook templates that contain AWS-authored machine learning models and visualizations to help you get started with IoT use cases related to device failure profiling, forecasting events such as low usage that might signal the customer will abandon the product, or segmenting devices by customer usage levels (for example heavy users, weekend users) or device health.

- **Prediction** - You can do statistical classification through a method called logistic regression. You can also use Long-Short-Term Memory (LSTM), which is a powerful neural network technique for predicting the output or state of a process that varies over time. The pre-built notebook templates also support the K-means clustering algorithm for device segmentation, which clusters your devices into cohorts of like devices. These templates are typically used to profile device health and device state such as HVAC units in a chocolate factory or wear and tear of blades on a wind turbine.

Visualize

- **QuickSight Integration** - AWS IoT Analytics provides a connector to Amazon QuickSight so you can visualize your data sets in a QuickSight dashboard. You can also visualize the results or your ad-hoc analysis in the embedded Jupyter Notebooks in the AWS IoT Analytics’ console.
How to Use AWS IoT Analytics

AWS IoT Analytics Components

Channel

A channel collects data from an MQTT topic and archives the raw, unprocessed messages before publishing the data to a pipeline. You can also send messages to a channel directly with the BatchPutMessage (p. 61) command.

Pipeline

A pipeline consumes messages from one or more channels and allows you to process the messages before storing them in a data store. The processing steps, called activities (see Pipeline Activities (p. 31)), perform transformations on your messages such as removing, renaming or adding message attributes, filtering messages based on attribute values, invoking your lambda functions on messages for advanced processing or performing mathematical transformations to normalize device data.

Data store

Pipelines store their processed messages in a data store. A data store is not a database, but it is a scalable and queryable repository of your messages. You can have multiple data stores for messages coming from different devices or locations, or filtered by message attributes depending on your pipeline configuration and requirements.

Data set

You retrieve data from a data store by creating a data set. A data set is similar to a materialized view from an SQL database. In fact, you create a data set by applying an SQL action. Data sets can be generated on a recurring schedule so the data you need is there when you need it without having to wait.

After you have a data set, you can explore and gain insights into your data through integration with Amazon QuickSight. Or you can perform more advanced analytical functions through integration with Jupyter Notebooks. Jupyter Notebooks provide powerful data science tools that can perform machine learning and a range of statistical analyses. For more information, see Notebook Templates (p. 30).
Accessing AWS IoT Analytics

As part of AWS IoT, AWS IoT Analytics provides the following interfaces to interact with your devices and the data they generate:

**AWS Command Line Interface (AWS CLI)**

Run commands for AWS IoT Analytics on Windows, OS X, and Linux. These commands allow you to create and manage things, certificates, rules, and policies. To get started, see the AWS Command Line Interface User Guide. For more information about the commands for AWS IoT, see `iot` in the AWS Command Line Interface Reference.

**Important**

Use the `aws iotanalytics` command to interact with AWS IoT Analytics using the CLI. Use the `aws iot` command to interact with other parts of the IoT system using the CLI.

**AWS IoT API**

Build your IoT applications using HTTP or HTTPS requests. These API actions allow you to create and manage things, certificates, rules, and policies. For more information about the API actions for AWS IoT, see Actions in the AWS IoT API Reference.

**AWS SDKs**

Build your AWS IoT Analytics applications using language-specific APIs. These SDKs wrap the HTTP/HTTPS API and allow you to program in any of the supported languages. For more information, see AWS SDKs and Tools.

**AWS IoT Device SDKs**

Build applications that run on your devices that send messages to AWS IoT Analytics. For more information see AWS IoT SDKs.

### AWS IoT Analytics Service Limits

<table>
<thead>
<tr>
<th>API</th>
<th>Limit Description</th>
<th>Adjustable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SampleChannelData</td>
<td>1 transaction per second per channel</td>
<td>yes</td>
</tr>
<tr>
<td>CreateDatasetContent</td>
<td>1 transaction per second per data set</td>
<td>yes</td>
</tr>
<tr>
<td>RunPipelineActivity</td>
<td>1 transaction per second</td>
<td>yes</td>
</tr>
<tr>
<td>other management APIs</td>
<td>20 transactions per second</td>
<td>yes</td>
</tr>
<tr>
<td>BatchPutMessage</td>
<td>1000 messages per second per channel; 100 messages per batch; 128Kb per message</td>
<td>yes; yes; no</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resource</th>
<th>Limit Description</th>
<th>Adjustable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>channel</td>
<td>50 per account</td>
<td>yes</td>
</tr>
<tr>
<td>data store</td>
<td>25 per account</td>
<td>yes</td>
</tr>
</tbody>
</table>
About Amazon Web Services

Amazon Web Services (AWS) is a collection of digital infrastructure services that developers can use when developing their applications. The services include computing, storage, database, and application synchronization (messaging and queuing).

AWS uses a pay-as-you-go service model. You are charged only for the services that you—or your applications—use. Also, to make AWS useful as a platform for prototyping and experimentation, AWS offers a free usage tier, in which services are free below a certain level of usage. For more information about AWS costs and the free usage tier see Test-Driving AWS in the Free Usage Tier.

If you don't have an AWS account, go to aws.amazon.com and choose Create an AWS Account.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Limit Description</th>
<th>Adjustable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>pipeline</td>
<td>100 per account</td>
<td>yes</td>
</tr>
<tr>
<td>activities</td>
<td>25 per pipeline</td>
<td>no</td>
</tr>
<tr>
<td>data set</td>
<td>100 per account</td>
<td>yes</td>
</tr>
<tr>
<td>minimum data set refresh interval</td>
<td>1 hour</td>
<td>yes</td>
</tr>
<tr>
<td>concurrent data set content generation</td>
<td>2 data sets simultaneously</td>
<td>no</td>
</tr>
</tbody>
</table>
Getting Started with AWS IoT Analytics

Follow these steps to collect, store, process and query your device data using AWS IoT Analytics.

Create a Data Store

1. Start by creating a data store to receive and store your messages. You can create multiple data stores to store data according to your needs, or you can start with a single data store to receive all of your AWS IoT messages:

   ```
   aws iotanalytics create-datastore --datastore-name mydatastore
   ```

2. To list the data stores you have already created:

   ```
   aws iotanalytics list-datastores
   ```

3. To get more information about a data store:

   ```
   aws iotanalytics describe-datastore --datastore-name mydatastore
   ```

Create a Channel

1. Incoming messages are sent to a channel, so the next step is to create a channel for your data. You will point a rule engine rule at this channel in a later step:

   ```
   aws iotanalytics create-channel --channel-name mychannel
   ```

2. To list the channels you have already created:

   ```
   aws iotanalytics list-channels
   ```

3. To get more information about a channel:

   ```
   aws iotanalytics describe-channel --channel-name mychannel
   ```

Create a Pipeline

To connect a channel to a data store, you need to create a pipeline. The simplest possible pipeline contains no activities other than specifying the channel that collects the data and identifying the data store to which the messages are sent. For information about more complicated pipelines, see Pipeline Activities (p. 31).
Create an IAM Role

You must create an IAM role that grants permission to send message data to an AWS IoT Analytics channel. First, create the role:

```
aws iam create-role --role-name myAnalyticsRole --assume-role-policy-document file://ardp.json
```

where the contents of the file `ardp.json` should look like this:

```
{
  "Version": "2012-10-17",
  "Statement": [
```

We recommend that you start with a pipeline that does nothing other than connect a channel to a data store. You can see how raw data flows to the data store before you introduce pipeline activities to process this data.

1. Create a pipeline:

```
aws iotanalytics create-pipeline --cli-input-json file://mypipeline.json
```

where the file `mypipeline.json` contains:

```
{
  "pipelineName": "mypipeline",
  "pipelineActivities": [
    {
      "channel": {
        "name": "mychannelactivity",
        "channelName": "mychannel",
        "next": "mystoreactivity"
      }
    },
    {
      "datastore": {
        "name": "mystoreactivity",
        "datastoreName": "mydatastore"
      }
    }
  ]
}
```

2. List your existing pipelines:

```
aws iotanalytics list-pipelines
```

3. View the configuration of an individual pipeline:

```
aws iotanalytics describe-pipeline --pipeline-name mypipeline
```

You now have a channel that routes data to a pipeline that stores data in a data store where it can be queried. To get device messages flowing into a channel, you must create a special rule in the AWS IoT platform to route messages from a topic into AWS IoT Analytics. (You can also use the "BatchPutMessage" command to send messages directly to a channel as shown in a later step.)
Create an IoT Rule

To connect a channel to a source of AWS IoT messages, create an AWS IoT rule that sends data to an AWS IoT Analytics channel:

```
aws iot create-topic-rule --rule-name analyticsTestRule --topic-rule-payload file://rule.json
```

The contents of the `rule.json` file should look like this:

```json
{
    "sql": "SELECT * FROM 'iot/test'",
    "ruleDisabled": false,
    "awsIotSqlVersion": "2016-03-23",
    "actions": [
        {
            "iotAnalytics": {
                "channelName": "mychannel",
                "roleArn": "arn:aws:iam::<your-account-number>:role/myAnalyticsRole"
            }
        }
    ]
}
```

where the channel name and the role are the ones you created in the previous steps.

Now you have joined a rule to a channel, a channel to a pipeline, and a pipeline to a data store. Any data matching the rule will now flow through AWS IoT Analytics to the data store ready to be queried.
Send an IoT Message

You can test the connections between the channel, pipeline, and data store by sending a message using either the AWS IoT console or the "BatchPutMessage" command (using the CLI).

**Option 1- Use the AWS IoT console to send an AWS IoT message:**

1. In the AWS IoT console, in the left navigation pane, choose **Test**.

2. On the MQTT client page, in the **Publish** section, in **Specify a topic**, type `iot/test`. In the message payload section, verify the following JSON contents are present, or type them if not:

   ```json
   {
     "message": "Hello from AWS IoT console"
   }
   ``

3. Choose **Publish to topic**.
This publishes a message that is routed to the data store you created earlier.

Option 2- Use the "BatchPutMessage" command to send an IoT message to the channel:

Note
This method does not require that you set up an IoT rule to route messages with a specific topic to your channel. But it does require that the device which sends its data/messages to the channel is capable of running software created with the AWS SDK or is capable of using the AWS CLI to call "BatchPutMessage".

1. Create a file "messages.json" which contains the messages to be sent (in this example only one message is sent):

```json
[
    { "messageId": "message01", "payload": "{ "message": "Hello from the CLI" }" }
]
```

2. Call batch-put-message:

```bash
aws iotanalytics batch-put-message --channel-name my_channel --messages file://messages.json
```

3. If there are no errors, you see the following output:

```json
{
    "batchPutMessageErrorEntries": []
}
```
Query Your Data

Now that you have data in a data store, you can query it to answer analytical questions. Although a data store is not a database, you use SQL expressions to query the data and produce results that are stored in a data set.

To query the data, you create a data set. A data set contains the SQL that you use to query the data store along with an optional schedule that repeats the query at a day and time you choose. You create the optional schedules using expressions similar to Amazon CloudWatch schedule expressions.

1. Create a data set:

   ```bash
   aws iotanalytics create-dataset --cli-input-json file://mydataset.json
   ```

   where the file `mydataset.json` contains:

   ```json
   {
     "datasetName": "mydataset",
     "actions": [
       {
         "actionName": "myaction",
         "queryAction": {
           "sqlQuery": "select * from mydatastore"
         }
       }
     ]
   }
   ```

2. Create the data set content by executing the query:

   ```bash
   aws iotanalytics create-dataset-content --dataset-name mydataset
   ```

   Wait a few minutes for the data set content to be created before you continue.

Access the Query Results

Data set contents are your results stored as a file, in CSV format. The file is made available to you through Amazon S3. The following example shows how you can check that your results are ready and download the file.

1. Call `get-dataset-content`:

   ```bash
   aws iotanalytics get-dataset-content --dataset-name mydataset
   ```

   If your data set contains any data, then the output from `get-dataset-content` will have "state": "SUCCEEDED" in the `status` field, like this:

   ```json
   {
     "timestamp": 1508189965.746,
     "entries": [
       {
         "entryName": "someEntry",
   ```
dataURI is a signed URL to the output results. It is valid for a short period of time (a few hours). Depending on your workflow, you might want to always call `get-dataset-content` before you access the content because calling this command generates a new signed URL.

**Explore Your Data**

AWS IoT Analytics provides direct integration with Amazon QuickSight. Amazon QuickSight is a fast business analytics service you can use to build visualizations, perform ad-hoc analysis, and quickly get business insights from your data. Amazon QuickSight enables organizations to scale to hundreds of thousands of users, and delivers responsive performance by using a robust in-memory engine (SPICE).

AWS IoT Analytics data sets can also be directly consumed by Jupyter Notebooks in order to perform advanced analytics and data exploration. Jupyter Notebooks is an open source solution. You can install and download from [http://jupyter.org/install.html](http://jupyter.org/install.html). Additional integration with SageMaker, an Amazon hosted notebook solution, is also available.
AWS IoT Analytics Console Quickstart Guide

This section shows you how to use the AWS IoT Analytics console to collect, store, process, and query your device data.

**Note**
Be aware as you enter names AWS IoT Analytics entities (channel, data set, data store, and pipeline) in the steps that follow, that any upper-case letters you use will be automatically changed to lower-case by the system. To avoid confusion, we recommend only using lower-case letters, underscores and digits when entering the names of these entities.

Sign in to the AWS IoT Analytics Console

If you do not have an AWS account, create one.

1. To create an AWS account, open the [AWS home page](https://aws.amazon.com) and choose **Create an AWS Account**.
2. Follow the online instructions. Part of the sign-up procedure involves receiving a phone call and entering a PIN using your phone’s keypad.
3. Sign in to the AWS Management Console and open the [AWS IoT Analytics console](https://aws.amazon.com/iot-analytics/).
4. On the [AWS IoT Analytics](https://aws.amazon.com/iot-analytics/) console home page, in the left navigation pane, choose **Prepare**.

Create a Channel

Incoming messages are sent to a channel.
1. In the left navigation pane, choose **Channels**, and then choose **Create a channel**: 

2. Enter the channel ID, then scroll down and choose **Next**.
3. Enter an IoT Core (MQTT) topic filter. Make a note of the topic filter you entered here, because you need it in a later step in order to create a message that will get picked up by your channel.

4. In the IAM role name area choose Create new. This will automatically create a new role with an appropriate policy attached to it. Then choose Create channel.
Create a Data store

A data store receives and stores your messages. You can create multiple data stores to store data according to your needs. For this example, you create a single data store to receive your AWS IoT messages:

1. In the left navigation pane, choose Data Stores. On the Create a data store page, choose Create a data store:

2. Enter the ID for your data store, and then choose Create data store:
Create a Pipeline

To connect a channel to a data store, you need to create a pipeline. The simplest possible pipeline contains no activities other than specifying the channel that collects the data and identifying the data store to which the messages are sent. We explore more complicated pipelines in Pipeline Activities (p. 31).

For this example, you create a pipeline that does nothing other than connect a channel to a data store. You can see how raw data flows to the data store. Later, you can introduce pipeline activities to process this data.

1. In the left navigation pane, choose Pipelines. On the Process messages with pipelines page, choose Create a pipeline:

2. Enter a pipeline ID, and then in Pipeline source, choose Edit:
3. Select the channel you created before, and then choose **Next**:
4. On the **Set attributes of your messages** page, enter an attribute name, select a type from the pull-down list, and enter an example value, then choose **Add new**. Repeat this for as many attributes as you want. When done, choose **Next**.

5. You won't be adding any pipeline activities right now, so on the **Enrich, transform, and filter messages** page, just choose **Next**.

6. On the **Save your processed messages in a data store** page, choose **Edit**, select the data store you created earlier, and then choose **Create pipeline**.
Create a Data Set

You now have a channel that routes data to a pipeline that stores data in a data store where it can be queried. To query the data, you create a data set. A data set contains SQL expressions that you use to query the data store along with an optional schedule that repeats the query at a day and time you choose. You can create the optional schedules by using expressions similar to Amazon CloudWatch schedule expressions.

1. On the left navigation pane, choose Analyze, and then choose Data Sets. On the Explore your data with a data set page, choose Create a data set:

2. On the Set ID and source page, enter an ID. In Select data store source choose Edit:
3. Select the data store you created earlier, and then choose **Next**.
4. On the **Author SQL Query** page, in the **Query** area, enter a SQL expression that selects your attributes, and then choose **Next**. Here is an example SQL expression that uses the attributes we specified previously in this tutorial:

```
SELECT thingid, temperature, humidity, datetime FROM my_datastore
```

5. You won't schedule a recurring run of the query at this point, so on the **Set query schedule** page just choose **Create data set**:

---

**Send an AWS IoT Message**

To generate some sample data, use the AWS IoT console to send an AWS IoT message.

1. In the **AWS IoT console**, in the left navigation pane, choose **Test**.
2. On the MQTT client page, in the Publish section, in **Specify a topic**, type `update/environment/dht1`. In the message payload section, enter the following JSON contents:

```json
{
    "thingid": "dht1",
    "temperature": 26,
    "humidity": 29,
    "datetime": "2018-01-26T07:06:01"
}
```

3. Choose **Publish to topic**.
If you have followed the example to this point, then this publishes a message that is captured by your channel, and then routed by your pipeline to your data store.

Access the Query Results

Data set contents are your results in a file, in CSV format.

1. In the AWS IoT Analytics console, in the left navigation pane, choose Analyze and then choose Data sets:

2. On the Data sets page, choose the data set you just created:
3. On the data set information page, in the upper-right corner, choose **Actions**, and then choose **Run query now**:

4. To check if the data set is ready, look for **SUCCEEDED** under the name of the data set in the upper left-hand corner. The **Details** section contains the query results:
5. In the left navigation pane, click **CSV**. Choose **Download** to view or save the CSV file that contains the query results.

It looks like this:

```
"thingid","temperature","humidity","datetime"
"dht1","26","29","2018-01-26T07:06:01"
```

6. Choose the left arrow in the upper-left corner to return to the main page of the **AWS IoT Analytics console**.
Explore Your Data

Jupyter Notebooks is an open source solution for advanced analyses and ad-hoc data exploration. Notebooks enable you to use templates and a scripting language, typically Python, to apply different transformations or normalizations to the data, aggregate metrics, and analyze and visualize data using data science libraries. You can even apply more complex analytics and machine learning, such as k-means clustering, to your data using these notebooks.

AWS IoT Analytics uses Amazon SageMaker notebook instances to host its Jupyter notebooks. Before you create a notebook instance, you must create a relationship between AWS IoT Analytics and Amazon SageMaker:

1. Go to the Amazon SageMaker console and create a notebook instance:
   a. Fill in the details, and then choose Create a new role. Make a note the role ARN.
   b. Create a notebook instance.

   This creates a role that Amazon SageMaker can use and a new notebook instance.

2. Go to the IAM console and modify the Amazon SageMaker role:
   a. Open the role. It should have one managed policy.
   b. Choose add inline policy, and then for Service, choose IoTAnalytics. Choose Select actions, and then type "GetDatasetContent" in the search box and select it. Choose Review policy.
   c. Review the policy for accuracy, give it a name, and then choose Create policy.

   This gives the newly created role permission to read a data set from AWS IoT Analytics.

3. Return to the AWS IoT Analytics console, and in the left navigation pane, choose Analyze, and then Notebooks:

4. On the Notebooks page, choose Create:

5. On the Select method page, choose Blank Notebook:
6. On the **Set up notebook** page, enter a name for the notebook. In **Select data set sources**, choose **Select**, and then select the data set you created earlier. In **Select a Notebook Instance**, choose **Select**, and then choose the notebook instance you created in Amazon SageMaker. Choose **Create Notebook**:

7. On the **Notebooks** page, use the triangles to open your notebook instance and the **IoTAnalytics** folder. Use the links to explore your data in Jupyter Notebooks:
You can download and install Jupyter Notebooks on your computer. Additional integration with an Amazon hosted notebook solution is also available.

**Notebook Templates**

The AWS IoT Analytics notebook templates contain AWS-authored machine learning models and visualizations to help you get started with IoT analytics use cases. These notebook templates can be explored as-is for educational purposes, or re-purposed to fit your data and deliver immediate value.

AWS IoT Analytics provides the following notebook templates:

1. Detecting Contextual Anomalies: Application of contextual anomaly detection in measured wind speed with a PEWMA model for time series data.
2. Solar Panel Output Forecasting: Application of piecewise, seasonal, linear time series models with trend to predicting the output of solar panels.
4. Smart Home Customer Segmentation: Application of k-means and PCA analysis to detect different customer segments in smart home usage data.
5. Smart City Congestion Forecasting: Application of LSTM to predict the utilization rates for city highways.
6. Smart City Air Quality Forecasting: Application of LSTM to predict particulate pollution in city centers.

You can find more information about notebook templates in the AWS IoT Analytics console under **Analyze/Notebooks**.
Pipeline Activities

The simplest functional pipeline connects a channel to a data store, which makes it a pipeline with two activities: a channel activity and a datastore activity. You can achieve more powerful message processing by adding additional activities to your pipeline.

AWS IoT Analytics provides the RunPipelineActivity command which allows you to simulate the results of running a pipeline activity on a message payload you provide. You might find this helpful when you are developing and debugging your pipeline activities. RunPipelineActivity Example demonstrates how it is used.

Channel Activity

The first activity in a pipeline must be the channel activity which determines the source of the messages to be processed.

```
{
    "channel": {
        "name": "MyChannelActivity",
        "channelName": "mychannel",
        "next": "MyLambdaActivity"
    }
}
```

Datastore Activity

The datastore activity, which specifies where to store the processed data, is the last activity.

```
{
    "datastore": {
        "name": "MyDatastoreActivity",
        "datastoreName": "mydatastore"
    }
}
```

Lambda Activity

A lambda activity can be used to perform more complex processing on the message. Examples include enriching the message with data from the output of external APIs or filtering the message based on logic from DynamoDB. However, you can use this activity to perform any sort of message-based processing, including filtering which messages are stored in the data store.

The AWS Lambda function used in this activity must receive and return an array of JSON objects. In the following example, the Lambda function modifies, and then returns, its event parameter.

The batchSize determines how many messages your Lambda function receives on each invocation. When you set it, keep in mind that an AWS Lambda function has a maximum timeout of five minutes. So the Lambda function must be able to process all messages in the batch in less than five minutes.
Lambda Activity

You will need to add a function policy to allow AWS IoT Analytics to invoke your Lambda function. Use the following CLI command:

```
aws lambda add-permission --function-name <lambda-function-name> --statement-id <your-statement> --principal iotanalytics.amazonaws.com --action lambda:InvokeFunction
```

Here is an example of a Lambda function used in AWS IoT Analytics. Given a device that publishes a message with a payload similar to:

```
{
    "thingid": "00001234abcd",
    "temperature": 26,
    "humidity": 29,
    "location": {
        "lat": 52.4332935,
        "lon": 13.231694
    },
    "ip": "192.168.178.54",
    "datetime": "2018-02-15T07:06:01"
}
```

and the following pipeline definition:

```
{
    "pipeline": {
        "activities": [
            {
                "channel": {
                    "channelName": "foobar_channel",
                    "name": "foobar_channel_activity",
                    "next": "lambda_foobar_activity"
                }
            },
            {
                "lambda": {
                    "lambdaName": "MyAnalyticsLambdaFunction",
                    "batchSize": 5,
                    "name": "lambda_foobar_activity",
                    "next": "foobar_store_activity"
                }
            },
            {
                "datastore": {
                    "datastoreName": "foobar_datastore",
                    "name": "foobar_store_activity"
                }
            }
        ],
        "name": "foobar_pipeline",
        "arn": "arn:aws:iotanalytics:eu-west-1:123456789012:pipeline/foobar_pipeline"
    }
}
```
The following Lambda Python function (MyAnalyticsLambdaFunction) adds the GMaps URL and the temperature, in Fahrenheit, to the message:

```python
import logging
import sys

# Configure logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
streamHandler = logging.StreamHandler(stream=sys.stdout)
formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')
streamHandler.setFormatter(formatter)
logger.addHandler(streamHandler)

def c_to_f(c):
    return 9.0/5.0 * c + 32

def lambda_handler(event, context):
    logger.info("event before processing: ").format(event)
    maps_url = 'N/A'

    for e in event:
        if 'location' in e:
            lat = e['location']['lat']
            lon = e['location']['lon']
            maps_url = "http://maps.google.com/maps?q={},{}".format(lat, lon)

        if 'temperature' in e:
            e['temperature_f'] = c_to_f(e['temperature'])

    logger.info("maps_url: ").format(maps_url)
    e['maps_url'] = maps_url

    logger.info("event after processing: ").format(event)

    return event
```

AddAttributes Activity

An addAttributes activity adds attributes based on existing attributes in the message. This lets you alter the shape of the message before it is stored, for example, to normalize data coming from different generations of device firmware.

This is best explained by example. Consider the input message:

```json
{
    "device": {
        "id": "device-123",
        "coord": [ 47.6152543, -122.3354883 ]
    }
}
```

and an addAttributes activity that looks like this:

```json
{
    "addAttributes": {
        "name": "MyAddAttributesActivity",
        "attributes": {
```
RemoveAttributes Activity

A removeAttributes activity removes attributes from a message. For example, given the message that was the result of the addAttributes activity:

```json
{  
  "device": {  
    "id": "device-123",  
    "coord": [ 47.6, -122.3 ]  
  },  
  "id": "device-123",  
  "lat": 47.6,  
  "lon": -122.3  
}
```

To normalize that message so that it includes only the required data at the root level, use the following removeAttributes activity:

```json
{  
  "removeAttributes": {  
    "name": "MyRemoveAttributesActivity",  
    "attributes": [  
      "device"  
    ],  
    "next": "MyDatastoreActivity"  
  }  
}
```

This results in the following message flowing along the pipeline:

```json
{  
  "id": "device-123",  
  "device": {  
    "id": "device-123",  
    "coord": [ 47.6, -122.3 ]  
  }  
}
```
SelectAttributes Activity

The selectAttributes activity creates a new message using only the specified attributes from the original message. Every other attribute is dropped. selectAttributes creates new attributes under the root of the message only. So given this message:

```
{
    "device": {
        "id": "device-123",
        "coord": [ 47.6152543, -122.3354883 ],
        "temp": 50,
        "hum": 40
    },
    "light": 90
}
```

and this activity:

```
{
    "selectAttributes": {
        "name": "MySelectAttributesActivity",
        "attributes": [
            "device.temp",
            "device.hum",
            "light"
        ],
        "next": "MyDatastoreActivity"
    }
}
```

The result is the following message flowing through the pipeline:

```
{
    "temp": 50,
    "hum": 40,
    "light": 90
}
```

Again, selectAttributes can only create root-level objects.

Filter Activity

A filter activity filters a message based on its attributes. The expression used in this activity looks like an SQL WHERE clause which must return a boolean:

```
{
    "filter": {
        "name": "MyFilterActivity",
        "filter": "temp > 40 AND hum < 20",
        "next": "MyDatastoreActivity"
    }
}
```
DeviceRegistryEnrich Activity

The `deviceRegistryEnrich` activity allows you to add data from the AWS IoT device registry to your message payload. For example, given the following message:

```json
{
  "temp": 50,
  "hum": 40,
  "device": {
    "thingName": "my-thing"
  }
}
```

and a `deviceRegistryEnrich` activity that looks like this:

```json
{
  "deviceRegistryEnrich": {
    "name": "MyDeviceRegistryEnrichActivity",
    "attribute": "metadata",
    "thingName": "device.thingName",
    "roleArn": "arn:aws:iam::<your-account-number>:role:MyEnrichRole",
    "next": "MyDatastoreActivity"
  }
}
```

The output message now looks like this:

```json
{
  "temp": 50,
  "hum": 40,
  "device": {
    "thingName": "my-thing"
  },
  "metadata": {
    "defaultClientId": "my-thing",
    "thingName": "my-thing",
    "thingArn": "arn:aws:iot:us-east-1:<your-account-number>:thing/my-thing",
    "version": 1,
    "thingName": "my-thing",
    "attributes": {},
    "thingId": "aaabbbccc-dddeefghh-jjkk-llmmnoopp"
  }
}
```

You must specify a role in the `roleArn` field of the activity definition that has the appropriate permissions attached. The role must have a permissions policy that looks like:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:DescribeThing"
      ]
    }
  ]
}
```
DeviceShadowEnrich Activity

A deviceShadowEnrich activity adds information from the AWS IoT Device Shadows service to a message. For example, given the message:

```json
{
  "temp": 50,
  "hum": 40,
  "device": { "thingName": "my-thing" }
}
```

and the following deviceShadowEnrich activity:

```json
{
  "deviceShadowEnrich": {
    "name": "MyDeviceShadowEnrichActivity",
    "attribute": "shadow",
    "thingName": "device.thingName",
    "roleArn": "arn:aws:iam::<your-account-number>:role:MyEnrichRole",
    "next": "MyDatastoreActivity"
  }
}
```

the result is a message that looks like this:

```json
{
  "temp": 50,
  "hum": 40,
  "device": { "thingName": "my-thing"
  },
  "shadow": {
}
You must specify a role in the `roleArn` field of the activity definition that has the appropriate permissions attached. The role must have a permissions policy that looks like:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "iot:GetThingShadow"
         ],
         "Resource": [
            "arn:aws:iot:<region>:<account-id>:thing/<thing-name>
         ]
      }
   ]
}
```

and a trust policy that looks like:

```json
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "",
         "Effect": "Allow",
         "Principal": {
            "Service": "iotanalytics.amazonaws.com"
         },
         "Action": [
            "sts:AssumeRole"
         ]
      }
   ]
}
```
Math Activity

A math activity computes an arithmetic expression using the message's attributes. The expression must return a number. For example, given the following input message:

```
{
  "tempF": 50,
}
```

after processing by the following math activity:

```
{
  "math": {
    "name": "MyMathActivity",
    "math": "((tempF - 32) * 5.0) / 9.0",
    "attribute": "tempC",
    "next": "MyDatastoreActivity"
  }
}
```

the resulting message looks like:

```
{
  "tempF": 50,
  "tempC": 9
}
```

RunPipelineActivity Example

Here is an example of how you would use the "RunPipelineActivity" command to test a pipeline activity. For this example we test a Math activity:

1. Create a file "maths.json" which contains the definition of the pipeline activity you want to test:

```
{
  "math": {
    "name": "MyMathActivity",
    "math": "((temp - 32) * 5.0) / 9.0",
    "attribute": "tempC"
  }
}
```

2. Create a file "payloads.json" which contains the example payloads that are used to test the pipeline activity:

```
[
  "{\"humidity\": 52, \"temp\": 68 }",
  "{\"humidity\": 52, \"temp\": 32 }"
]
```

3. Call "RunPipelineActivities" from the command line:

4. This produces the following results:

```json
{
  "logResult": "",
  "payloads": [
    "eyJodW1pZGl0eSI6NTIsInRlbXAiOjY4LCJ0ZW1wQyI6MjB9",
    "eyJodW1pZGl0eSI6NTIsInRlbXAiOjMyLCJ0ZW1wQyI6MH0="
  ]
}
```

5. The "payloads" listed in the results are Base64-encoded strings. When these strings are decoded, you get the following results:

```json
{"humidity":52,"temp":68,"tempC":20}
{"humidity":52,"temp":32,"tempC":0}
```
Reprocessing Channel Data

AWS IoT Analytics allows you to reprocess channel data or, to put it another way, to replay existing raw data through a pipeline. This can be useful if:

- You want to replay existing ingested data rather than starting over.
- You make an update to a pipeline and want to bring existing data up-to-date with the changes.

To trigger the reprocessing of existing raw data, use the `StartPipelineReprocessing` (p. 121) command. Note the following:

- The "startTime" and "endTime" parameters specify when the raw data was ingested, but these are rough estimates. You can round to the nearest hour. The "startTime" is inclusive, but the "endTime" is exclusive.
- The command launches the reprocessing asynchronously and returns immediately.
- There is no guarantee that reprocessed messages are processed in the order they were originally received. It is roughly the same, but not exact.

To cancel the reprocessing of existing raw data, use the `CancelPipelineReprocessing` (p. 63) command.

Use the `DescribePipeline` (p. 93) command to check the status of the reprocessing. See the "reprocessingSummaries" field in the response.
## SQL Support

Data sets are generated using SQL expressions on data in a data store. AWS IoT Analytics uses the same SQL queries, functions and operators as Amazon Athena.

This means that AWS IoT Analytics supports a subset of ANSI standard SQL syntax as follows:

```sql
SELECT [ ALL | DISTINCT ] select_expression [, ...] 
[ FROM from_item [, ...] ] 
[ WHERE condition ] 
[ GROUP BY [ ALL | DISTINCT ] grouping_element [, ...] ] 
[ HAVING condition ] 
[ UNION [ ALL | DISTINCT ] union_query ] 
[ ORDER BY expression [ ASC | DESC ] [ NULLS FIRST | NULLS LAST] [, ...] ] 
[ LIMIT [ count | ALL ] ]
```

However, AWS IoT Analytics does not support `JOIN` or `WITH` clauses.

Amazon Athena and AWS IoT Analytics SQL functionality are based on Presto 0.172, so they support the following functions:

- Logical Operators
- Comparison Functions and Operators
- Conditional Expressions
- Conversion Functions
- Mathematical Functions and Operators
- Bitwise Functions
- Decimal Functions and Operators
- String Functions and Operators
- Binary Functions
- Date and Time Functions and Operators
- Regular Expression Functions
- JSON Functions and Operators
- URL Functions
- Aggregate Functions
- Window Functions
- Color Functions
- Array Functions and Operators
- Map Functions and Operators
- Lambda Expressions and Functions
- Teradata Functions

However, AWS IoT Analytics and Amazon Athena do not support:

- User-defined functions (UDFs or UDAFs).
- Stored procedures.
- Some data types.
- `CREATE TABLE AS SELECT` statements.
• INSERT INTO statements.
• Prepared statements. You cannot run EXECUTE with USING.
• CREATE TABLE LIKE.
• DESCRIBE INPUT and DESCRIBE OUTPUT.
• EXPLAIN statements.
• Federated connectors.

These are the supported data types:

• primitive_type
  • TINYINT
  • SMALLINT
  • INT
  • BIGINT
  • BOOLEAN
  • DOUBLE
  • FLOAT
  • STRING
  • TIMESTAMP
  • DECIMAL[(precision, scale)]
  • DATE
  • CHAR (fixed-length character data with a specified length)
  • VARCHAR (variable-length character data with a specified length)

• array_type
  • ARRAY<data_type>

• map_type
  • MAP<primitive_type, data_type>

• struct_type
  • STRUCT<col_name:data_type[COMMENT col_comment][,...]>
Visualizing AWS IoT Analytics Data with QuickSight

AWS IoT Analytics provides direct integration with Amazon QuickSight. Amazon QuickSight is a fast business analytics service you can use to build visualizations, perform ad-hoc analysis, and quickly get business insights from your data. Amazon QuickSight enables organizations to scale to hundreds of thousands of users, and delivers responsive performance by using a robust in-memory engine (SPICE). You can select your AWS IoT Analytics data sets in the QuickSight console and start creating dashboards and visualizations.

To get started with your QuickSight visualizations, you must create a QuickSight account. Make sure you give QuickSight access to your AWS IoT Analytics data when you set up your account. If you already have an account, give QuickSight access to your AWS IoT Analytics data by choosing Admin, Manage QuickSight, Account Settings. Choose Edit AWS Permissions, and then select Amazon IoT Analytics.

After your account is set up, from the main QuickSight console page choose New Analysis and New data set, and then select AWS IoT Analytics as the source. Type a name for your data source, select a data set to import, and then choose Create data source.

After your data source is created, you can create visualizations in QuickSight:
For information about QuickSight dashboards and data sets, see the QuickSight documentation.
AWS IoT Analytics Troubleshooting Guide

Q: Are my messages getting into AWS IoT Analytics?

- Check if the rule to inject data into the channel through the rules-engine is configured correctly:

  ```bash
  aws iot get-topic-rule --rule-name <your-rule-name>
  ```

  The response should look like:

  ```json
  {
    "ruleArn": "arn:aws:iot:us-west-2:<your-account-id>:rule/<your-rule-name>",
    "rule": {
      "awsIoTSqlVersion": "2016-03-23",
      "sql": "SELECT * FROM 'iot/<your-rule-name>'",
      "ruleDisabled": false,
      "actions": [
        {
          "iotAnalytics": {
            "channelArn": "arn:aws:iotanalytics:<region>:<your_account_number>:channel/<your-channel-name>"
          }
        }
      ],
      "ruleName": "<your-rule-name>"
    }
  }
  ```

  Make sure the region and channel name used in the rule are correct. To ensure your data is reaching the rules engine and the rule is being executed correctly, you might want to add a new target to store incoming messages in the S3 bucket temporarily.

Q: Why is my pipeline losing messages? How do I fix it?

- An activity has received an invalid JSON input:

  All activities, except Lambda activities, specifically require a valid JSON string as input. If the JSON received by an activity is invalid, then the message is dropped and does not make its way into the data store. Make sure you are ingesting valid JSON messages into the service. In case of binary input, make sure the first activity in your pipeline is a Lambda activity that converts the binary data to valid JSON before passing it to the next activity or storing it in the data store.

- A Lambda function invoked by a Lambda activity has insufficient permissions:

  Make sure that each Lambda function in a Lambda activity has permission to be invoked from the AWS IoT Analytics service. You can use the following CLI command to grant permission:

  ```bash
  aws lambda add-permission --function-name <name> --region <region> --statement-id <id> --principal iotanalytics.amazonaws.com --action lambda:InvokeFunction
  ```

- A filter or removeAttribute activity is incorrectly defined:
Make sure the definitions of any `filter` or `removeAttribute` activities are correct. If you filter out a message or remove all attributes from a message, that message is not added to the data store.

**Q: Why is there no data in my data store?**

- There is a delay between data ingestion and data availability:
  
  It might take several minutes after data is ingested into a channel before that data is available in the data store. The time varies based on the number of pipeline activities and the definition of any custom lambda activities in your pipeline.

- Messages are being filtered out in your pipeline:

  Make sure you are not dropping messages in the pipeline. (See the previous question and response.)

- Your data set query is incorrect:

  Make sure the query that generates the data set from the data store is correct. Remove any unnecessary filters from the query to ensure your data reaches your data store.

**Q: Why does my data set just show `:code:`__dt`?`**

- This column is added by the service automatically and contains the approximate ingestion time of the data. It may be used to optimize your queries. If your data set contains nothing but this, see the previous question and response.

**Q: How do I code an event driven by the data set completion?**

- You will have to set up polling based on the `describe-dataset` command to check if the status of the data set with a particular timestamp is `SUCCEEDED`.

**Q: How do I correctly configure my notebook instance to use the IoTAnalytics Service?**

- Follow these steps to make sure the IAM role you are using to create the notebook instance has the required permissions:
  
  1. Go to the Amazon SageMaker console and create a notebook instance.
  2. Fill in the details and choose **create a new role**. Make a note of the role ARN.
  3. Create the notebook instance. This also creates a role that Amazon SageMaker can use.
  4. Go to the IAM console and modify the newly created Amazon SageMaker role. When you open that role, it should have a managed policy.
  5. Click **add inline policy**, choose `IotAnalytics` as the service, and under read permission, select `GetDatasetContent`.
  6. Review the policy, add a policy name, and then **create** it. The newly created role now has policy permission to read a data set from AWS IoT Analytics.
  7. Go to the AWS IoT Analytics console and create notebooks in the notebook instance.
  8. Wait for the notebook instance to be in the "In Service" state.
  9. Choose **create notebooks**, and select the notebook instance you created. This creates a Jupyter notebook with the selected template that can access your data sets.

**Q: Why can't I create notebooks in an instance?**

- Make sure you create a notebook instance with the correct IAM policy. (Follow the steps in the previous question.)
• Make sure the notebook instance is in the "In Service" state. (When you create an instance, it starts in a "Pending" state. It usually takes about five minutes for it to go into the "In Service" state. If the notebook instance goes into the "Failed" state after about five minutes, check the permissions again.)

Q: Why is my notebook unable to pull data from my data sets?
• Make sure the notebook instance has all the required permissions as described earlier. The notebook instance must have read permission to the data set's content.

Q: Why am I not seeing my data sets in QuickSight?
• Follow these steps to make sure you have given QuickSight read permission for data set content:
  1. Click the icon in the upper-right corner (mentioning the account name) and choose Manage QuickSight
  2. Choose Account settings, and then choose Edit AWS permissions
  3. Choose AWS IoT Analytics from the list. This gives QuickSight read permissions to your data sets.
  4. Apply the permissions and try again to visualize your data.
Logging AWS IoT Analytics API Calls with CloudTrail

AWS IoT Analytics is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS IoT Analytics. CloudTrail captures a subset of API calls for AWS IoT Analytics as events, including calls from the AWS IoT Analytics console and from code calls to the AWS IoT Analytics APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS IoT Analytics. 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 AWS IoT Analytics, 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.

AWS IoT Analytics Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS IoT Analytics, 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 AWS IoT Analytics, 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 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:

- 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

AWS IoT Analytics supports logging the following actions as events in CloudTrail log files:

- CancelPipelineReprocessing (p. 63)
- CreateChannel (p. 64)
- CreateDataset (p. 66)
- CreateDatasetContent (p. 69)
- CreateDatastore (p. 70)
- CreatePipeline (p. 72)
- DeleteChannel (p. 79)
- DeleteDataset (p. 80)
- DeleteDatasetContent (p. 81)
- DeleteDatastore (p. 82)
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 Amazon Identity & 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 AWS IoT Analytics Log File Entries

A trail is a configuration that enables delivery of events as log files to an 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, the date and time of the action, request parameters, and so on. CloudTrail log files are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the CreateChannel action.

```json
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "ABCDE12345FGHIJ67890B:AnalyticsChannelTestFunction",
    "arn": "arn:aws:sts::123456789012:assumed-role/AnalyticsRole/AnalyticsChannelTestFunction",
    "accountId": "123456789012",
    "accessKeyId": "ABCDE12345FGHIJ67890B",
    "sessionContext": {
```

```json
```
"attributes": {  
  "mfaAuthenticated": "false",  
  "creationDate": "2018-02-14T23:12Z"
},  
"sessionIssuer": {  
  "type": "Role",  
  "principalId": "ABCDE12345FGHIJ67890B",  
  "arn": "arn:aws:iam::123456789012:role/AnalyticsRole",  
  "accountId": "123456789012",  
  "userName": "AnalyticsRole"
}  
}

"eventTime": "2018-02-14T23:55:14Z",  
"eventSource": "iotanalytics.amazonaws.com",  
"eventName": "CreateChannel",  
"awsRegion": "us-east-1",  
"sourceIPAddress": "198.162.1.0",  
"userAgent": "aws-internal/3 exec-env/AWS_Lambda_java8",  
"requestParameters": {  
  "channelName": "channel_channeltest"
},  
"responseElements": {  
  "retentionPeriod": {  
    "unlimited": true
  },  
  "channelName": "channel_channeltest",  
  "channelArn": "arn:aws:iotanalytics:us-east-1:123456789012:channel/channel_channeltest"
},  
"requestID": "7f871429-11e2-11e8-9eee-0781b50ac59",  
"eventID": "17885899-6977-41be-a6a0-74bb95a78294",  
"eventType": "AwsApiCall",  
"recipientAccountId": "123456789012"
}

The following example shows a CloudTrail log entry that demonstrates the CreateDataset action.

{
  "eventVersion": "1.05",  
  "userIdentity": {  
    "type": "AssumedRole",  
    "principalId": "ABCDE12345FGHIJ67890B:AnalyticsDatasetTestFunction",  
    "arn": "arn:aws:sts::123456789012:assumed-role/AnalyticsRole/AnalyticsDatasetTestFunction",  
    "accountId": "123456789012",  
    "accessKeyId": "ABCDE12345FGHIJ67890B",  
    "sessionContext": {  
      "attributes": {  
        "mfaAuthenticated": "false",  
        "creationDate": "2018-02-14T23:41:36Z"
      },  
      "sessionIssuer": {  
        "type": "Role",  
        "principalId": "ABCDE12345FGHIJ67890B",  
        "arn": "arn:aws:iam::123456789012:role/AnalyticsRole",  
        "accountId": "123456789012",  
        "userName": "AnalyticsRole"
      }
    },  
    "eventTime": "2018-02-14T23:53:39Z",  
    "eventSource": "iotanalytics.amazonaws.com",  
    "eventName": "CreateDataset",  
    "awsRegion": "us-east-1",  
    "sourceIPAddress": "198.162.1.0",
  }
}
"userAgent": "aws-internal/3 exec-env/AWS_Lambda_java8",
"requestParameters": {
  "datasetName": "dataset_datasettest"
},
"responseElements": {
  "datasetArn": "arn:aws:iotanalytics:us-east-1:123456789012:dataset/
dataset_datasettest",
  "datasetName": "dataset_datasettest"
},
"requestID": "46ee8dd9-11e2-11e8-979a-6198b668c3f0",
"eventID": "5abe21f6-ee1a-48ef-afc5-c77211235303",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"}
Setting Up CloudWatch Logs

AWS IoT Analytics supports logging with Amazon CloudWatch. You can enable and configure Amazon CloudWatch logging for AWS IoT Analytics using the `PutLoggingOptions` command. This section describes that command and the permissions you must set using AWS Identity and Access Management (IAM) and Amazon CloudWatch.

For more information about CloudWatch Logs see the Amazon CloudWatch Logs User Guide. For more information about AWS IAM, see the AWS Identity and Access Management User Guide.

**Note**
Before you enable AWS IoT Analytics logging, make sure you understand the CloudWatch Logs access permissions. Users with access to CloudWatch Logs can see your debugging information. See Authentication and Access Control for Amazon CloudWatch Logs.

Create a Logging Role

First, you must create an AWS IAM role with permissions to perform the logging. Use the AWS IAM console or the AWS IAM commands:

- `CreateRole`
  ```
  aws iam create-role ...
  ```
- `PutRolePolicy`
  ```
  aws iam put-role-policy ...
  ```

You will use the ARN of this role later when you call the AWS IoT Analytics `PutLoggingOptions` command. Here are the trust policy (used in `CreateRole`) and the role policy (used in `PutRolePolicy`) you should attach to this AWS IAM role.

**trust policy**:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": [
        {"Service": "iotanalytics.amazonaws.com"},
        "Action": "sts:AssumeRole"
      ]
    }
  ]
}
```

**role policy**:

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
```
"Effect": "Allow",
"Action": [
    "logs:CreateLogGroup",
    "logs:CreateLogStream"
],
"Resource": [
    "arn:aws:logs:*:*:*"
]
}
]
}

In addition, you must give AWS IoT Analytics permission to put log events to Amazon CloudWatch using the Amazon CloudWatch command:

- **PutResourcePolicy**

  ```
  aws logs put-resource-policy ...
  ```

Use the following resource policy:

```
resource policy:
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "Service": "iotanalytics.amazonaws.com"
         },
         "Action": "logs:PutLogEvents",
         "Resource": "*"
      }
   ]
}
```

## Configure and Enable Logging

Configure and Enable Logging

Use the `PutLoggingOptions` command to configure and enable Amazon CloudWatch logging for AWS IoT Analytics. The `roleArn` in the `loggingOptions` field should be the ARN of the role you created in the previous section. You can also use the `DescribeLoggingOptions` command to check your logging options settings.

### PutLoggingOptions

Sets or updates the AWS IoT Analytics logging options.

Note that if you update the value of any `loggingOptions` field, it takes up to one minute for the change to take effect. Also, if you change the policy attached to the role you specified in the `roleArn` field (for example, to correct an invalid policy) it takes up to 5 minutes for that change to take effect.

**CLI Synopsis:**

```
aws iotanalytics put-logging-options
--logging-options <value>
```
cli-input-json format:

```json
{
  "loggingOptions": {
    "roleArn": "string",
    "level": "string",
    "enabled": "boolean"
  }
}
```

fields:

- loggingOptions
  
  *type*: `LoggingOptions`

  The new values of the AWS IoT Analytics logging options.

- roleArn
  
  *type*: `string`; (length- max:2048 min:20)

  The ARN of the role that grants permission to AWS IoT Analytics to perform logging.

- level
  
  *type*: `string`

  The logging level. Currently, only "ERROR" is supported. enum: ERROR

- enabled
  
  *type*: `boolean`

  If true, logging is enabled for AWS IoT Analytics.

Output:

None

Errors:

- `InvalidRequestException`
  
  The request was not valid.

  HTTP response code: 400

- `InternalFailureException`
  
  There was an internal failure.

  HTTP response code: 500

- `ServiceUnavailableException`
  
  The service is temporarily unavailable.

  HTTP response code: 503

- `ThrottlingException`
The request was denied due to request throttling.

HTTP response code: 429

**DescribeLoggingOptions**

Retrieves the current settings of the AWS IoT Analytics logging options.

*CLI Synopsis:*

```bash
aws iotanalytics describe-logging-options
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

*cli-input-json format:*

```json
{}
```

*fields:*

**Output:*

```json
{
  "loggingOptions": {
    "roleArn": "string",
    "level": "string",
    "enabled": "boolean"
  }
}
```

*fields:*

- **loggingOptions**
  
  *type: LoggingOptions*
  
  The current settings of the AWS IoT Analytics logging options.

- **roleArn**
  
  *type: string; (length- max:2048 min:20)*
  
  The ARN of the role that grants permission to AWS IoT Analytics to perform logging.

- **level**
  
  *type: string*
  
  The logging level. Currently, only "ERROR" is supported. enum: ERROR

- **enabled**
  
  *type: boolean*
  
  If true, logging is enabled for AWS IoT Analytics.

*Errors:*
• InvalidRequestException
  The request was not valid.
  HTTP response code: 400
• ResourceNotFoundException
  A resource with the specified name could not be found.
  HTTP response code: 404
• InternalFailureException
  There was an internal failure.
  HTTP response code: 500
• ServiceUnavailableException
  The service is temporarily unavailable.
  HTTP response code: 503
• ThrottlingException
  The request was denied due to request throttling.
  HTTP response code: 429

Namespace, Metrics and Dimensions

AWS IoT Analytics puts the following metrics into the Amazon CloudWatch repository:

<table>
<thead>
<tr>
<th>Namespace</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS/IoTAnalytics</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionExecution</td>
<td>The number of actions executed.</td>
</tr>
<tr>
<td>ActivityExecutionError</td>
<td>The number of errors generated while executing the pipeline activity.</td>
</tr>
<tr>
<td>IncomingMessages</td>
<td>The number of messages coming into the channel.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionType</td>
<td>The type of action that is being monitored.</td>
</tr>
<tr>
<td>ChannelName</td>
<td>The name of the channel that is being monitored.</td>
</tr>
<tr>
<td>DatasetName</td>
<td>The name of the data set that is being monitored.</td>
</tr>
<tr>
<td>DatastoreName</td>
<td>The name of the data store that is being monitored.</td>
</tr>
</tbody>
</table>
### Namespace, Metrics and Dimensions

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PipelineActivityName</td>
<td>The name of the pipeline activity that is being monitored.</td>
</tr>
<tr>
<td>PipelineActivityType</td>
<td>The type of the pipeline activity that is being monitored.</td>
</tr>
<tr>
<td>PipelineName</td>
<td>The name of the pipeline that is being monitored.</td>
</tr>
</tbody>
</table>
Tagging Your AWS IoT Analytics Resources

To help you manage your channels, data sets, data stores and pipelines, you can optionally assign your own metadata to each of these resources in the form of tags. This chapter describes tags and shows you how to create them.

Tag Basics

Tags enable you to categorize your AWS IoT Analytics resources in different ways, for example, by purpose, owner, or environment. This is useful when you have many resources of the same type — you can quickly identify a specific resource based on the tags you've assigned to it. Each tag consists of a key and optional value, both of which you define. For example, you could define a set of tags for your channels that helps you track the type of device responsible for each channel's message source. We recommend that you devise a set of tag keys that meets your needs for each resource type. Using a consistent set of tag keys makes it easier for you to manage your resources. You can search and filter the resources based on the tags you add.

You can also use tags to categorize and track your costs. When you apply tags to channels, data sets, data stores, or pipelines, AWS generates a cost allocation report as a comma-separated value (CSV) file with your usage and costs aggregated by your tags. You can apply tags that represent business categories (such as cost centers, application names, or owners) to organize your costs across multiple services. For more information about using tags for cost allocation, see Use Cost Allocation Tags in the AWS Billing and Cost Management User Guide.

For ease of use, use the Tag Editor in the AWS Management Console, which provides a central, unified way to create and manage your tags. For more information, see Working with Tag Editor in Getting Started with the AWS Management Console.

You can also work with tags using the AWS CLI and the AWS IoT Analytics API. You can associate tags with channels, data sets, data stores and pipelines when you create them; use the `Tags` field in the following commands:

- CreateChannel (p. 64)
- CreateDataset (p. 66)
- CreateDatastore (p. 70)
- CreatePipeline (p. 72)

You can add, modify, or delete tags for existing resources; use the following commands:

- TagResource (p. 122)
- ListTagsForResource (p. 110)
- UntagResource (p. 124)

You can edit tag keys and values, and you can remove tags from a resource at any time. You can set the value of a tag to an empty string, but you can't set the value of a tag to null. If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the old value. If you delete a resource, any tags associated with the resource are also deleted.
Tag Restrictions

The following basic restrictions apply to tags:

- Maximum number of tags per resource — 50
- Maximum key length — 127 Unicode characters in UTF-8
- Maximum value length — 255 Unicode characters in UTF-8
- Tag keys and values are case-sensitive.
- Do not use the `aws:` prefix in your tag names or values because it is reserved for AWS use. You can’t edit or delete tag names or values with this prefix. Tags with this prefix do not count against your tags per resource limit.
- If your tagging schema is used across multiple services and resources, remember that other services may have restrictions on allowed characters. Generally, allowed characters are: letters, spaces, and numbers representable in UTF-8, plus the following special characters: + - = . _ : / @.
AWS IoT Analytics Commands

BatchPutMessage

Sends messages to a channel.

CLI Synopsis:

```bash
aws iotanalytics batch-put-message
   --channel-name <value>
   --messages <value>
   [--cli-input-json <value>]
   [--generate-cli-skeleton]
```

`cli-input-json` format:

```json
{
   "channelName": "string",
   "messages": [
       {
           "messageId": "string",
           "payload": "blob"
       }
   ]
}
```

**fields:**

- `channelName`
  
  *type*: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)
  
  The name of the channel where the messages are sent.

- `messages`
  
  *type*: list member: Message
  
  The list of messages to be sent. Each message has format: '{ "messageId": "string", "payload": "string"}'.

- `messageId`
  
  *type*: string; (length- max:128 min:1)
  
  The ID you wish to assign to the message. Each "messageId" must be unique within each batch sent.

- `payload`
  
  *type*: blob
  
  The payload of the message. This may be a JSON string or a Base64-encoded string representing binary data (in which case you must decode it by means of a pipeline activity).

Output:

```json
{
}
```
```json
"batchPutMessageErrorEntries": [  
  {  
    "messageId": "string",
    "errorCode": "string",
    "errorMessage": "string"
  }
]
```

**fields:**

- **batchPutMessageErrorEntries**
  
  *type:* list member: BatchPutMessageErrorEntry

  A list of any errors encountered when sending the messages to the channel.

- **messageId**
  
  *type:* string; (length- max:128 min:1)

  The ID of the message that caused the error. (See the value corresponding to the "messageId" key in the message object.)

- **errorCode**
  
  *type:* string

  The code associated with the error.

- **errorMessage**
  
  *type:* string

  The message associated with the error.

**Errors:**

- **ResourceNotFoundException**

  A resource with the specified name could not be found.

  HTTP response code: 404

- **InvalidRequestException**

  The request was not valid.

  HTTP response code: 400

- **InternalFailureException**

  There was an internal failure.

  HTTP response code: 500

- **ServiceUnavailableException**

  The service is temporarily unavailable.

  HTTP response code: 503

- **ThrottlingException**

  The request was denied due to request throttling.
CancelPipelineReprocessing

Cancels the reprocessing of data through the pipeline.

**CLI Synopsis:**

```bash
aws iotanalytics cancel-pipeline-reprocessing
   --pipeline-name <value>
   --reprocessing-id <value>
   [--cli-input-json <value>]
   [--generate-cli-skeleton]
```

**cli-input-json format:**

```json
{
   "pipelineName": "string",
   "reprocessingId": "string"
}
```

**fields:**

- **pipelineName**
  - *type:* string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9\-_.]+$)
  - The name of pipeline for which data reprocessing is canceled.

- **reprocessingId**
  - *type:* string
  - The ID of the reprocessing task (returned by "StartPipelineReprocessing").

**Output:**

None

**Errors:**

- **ResourceNotFoundException**
  - A resource with the specified name could not be found.
  - HTTP response code: 404

- **InvalidRequestException**
  - The request was not valid.
  - HTTP response code: 400

- **InternalFailureException**
  - There was an internal failure.
  - HTTP response code: 500

- **ServiceUnavailableException**
CreateChannel

Creates a channel. A channel collects data from an MQTT topic and archives the raw, unprocessed messages before publishing the data to a pipeline.

CLI Synopsis:

aws iotanalytics create-channel
  --channel-name <value>
  [--retention-period <value>]
  [--tags <value>]
  [--cli-input-json <value>]
  [--generate-cli-skeleton]

cli-input-json format:

```json
{
  "channelName": "string",
  "retentionPeriod": {
    "unlimited": "boolean",
    "numberOfDays": "integer"
  },
  "tags": [
    {
      "key": "string",
      "value": "string"
    }
  ]
}
```

fields:

- **channelName**
  
  *type:* string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the channel.

- **retentionPeriod**
  
  *type:* RetentionPeriod

  How long, in days, message data is kept for the channel.

- **unlimited**
  
  *type:* boolean

  If true, message data is kept indefinitely.

- **numberOfDays**
  
  *type:* integer

  The number of days message data is kept for the channel.
CreateChannel

- **type**: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The "unlimited" parameter must be false.

- **tags**
  - **type**: list member: Tag

    Metadata which can be used to manage the channel.

- **key**
  - **type**: string; (length- max:256 min:1)

    The tag's key.

- **value**
  - **type**: string; (length- max:256 min:1)

    The tag's value.

**Output**:

```json
{
  "channelName": "string",
  "channelArn": "string",
  "retentionPeriod": {
    "unlimited": "boolean",
    "numberOfDays": "integer"
  }
}
```

**fields**:

- **channelName**
  - **type**: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9\-_]+$)

    The name of the channel.

- **channelArn**
  - **type**: string

    The ARN of the channel.

- **retentionPeriod**
  - **type**: RetentionPeriod

    How long, in days, message data is kept for the channel.

- **unlimited**
  - **type**: boolean

    If true, message data is kept indefinitely.

- **numberOfDays**
  - **type**: integer java class: java.lang.Integer range- min:1

    The number of days that message data is kept. The "unlimited" parameter must be false.
CreateDataset

CreateDataset

CreateDataset

Errors:

• InvalidRequestException

  The request was not valid.

  HTTP response code: 400

• ResourceAlreadyExistsException

  A resource with the same name already exists.

  HTTP response code: 409

• InternalFailureException

  There was an internal failure.

  HTTP response code: 500

• ServiceUnavailableException

  The service is temporarily unavailable.

  HTTP response code: 503

• ThrottlingException

  The request was denied due to request throttling.

  HTTP response code: 429

• LimitExceededException

  The command caused an internal limit to be exceeded.

  HTTP response code: 410

CLI Synopsis:

aws iotanalytics create-dataset
  --dataset-name <value>
  --actions <value>
  [--triggers <value>]
  [--tags <value>]
  [--cli-input-json <value>]
  [--generate-cli-skeleton]

cli-input-json format:

{
  "datasetName": "string",
  "actions": [
    {
      "actionName": "string",
      "queryAction": {
       ...
    
}
fields:

- **datasetName**
  
  *type*: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set.

- **actions**

  *type*: list member: DatasetAction

  A list of actions that create the data set. Only one action is supported at this time.

- **actionName**

  *type*: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data set action.

- **queryAction**

  *type*: SqlQueryDatasetAction

  An "SqlQueryDatasetAction" object that contains the SQL query to modify the message.

- **sqlQuery**

  *type*: string

  An SQL query string.

- **triggers**

  *type*: list member: DatasetTrigger

  A list of triggers. A trigger causes data set content to be populated at a specified time or time interval. The list of triggers can be empty or contain up to five DataSetTrigger objects.

- **schedule**

  *type*: Schedule

  The "Schedule" when the trigger is initiated.

- **expression**

  *type*: string
The expression that defines when to trigger an update. For more information, see Schedule Expressions for Rules in the Amazon CloudWatch documentation.

- tags
  type: list member: Tag
  Metadata which can be used to manage the data set.
  - key
    type: string; (length- max:256 min:1)
    The tag's key.
  - value
    type: string; (length- max:256 min:1)
    The tag's value.

Output:

```
{
  "datasetName": "string",
  "datasetArn": "string"
}
```

**fields:**

- datasetName
  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_.]+$)
  The name of the data set.
- datasetArn
  type: string
  The ARN of the data set.

Errors:

- InvalidRequestException
  The request was not valid.
  HTTP response code: 400
- ResourceAlreadyExistsException
  A resource with the same name already exists.
  HTTP response code: 409
- InternalFailureException
  There was an internal failure.
  HTTP response code: 500
- ServiceUnavailableException
CreateDatasetContent

Creates the content of a data set by applying an SQL action.

CLI Synopsis:

aws iotanalytics create-dataset-content
  --dataset-name <value>
  [--cli-input-json <value>]
  [--generate-cli-skeleton]

cli-input-json format:

{
  "datasetName": "string"
}

fields:

• datasetName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]$)

  The name of the data set.

Output:

None

Errors:

• InvalidRequestException

  The request was not valid.

  HTTP response code: 400

• ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
CreateDatastore

Creates a data store, which is a repository for messages.

CLI Synopsis:

```
aws iotanalytics create-datastore
  --datastore-name <value>
  [--retention-period <value>]
  [--tags <value>]
  [--cli-input-json <value>]
  [--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datastoreName": "string",
  "retentionPeriod": {
    "unlimited": "boolean",
    "numberOfDays": "integer"
  },
  "tags": [
    {  
      "key": "string",
      "value": "string"
    }
  ]
}
```

fields:

- datastoreName
  
  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_.]+$)

  The name of the data store.

- retentionPeriod
  
  type: RetentionPeriod

  How long, in days, message data is kept for the data store.

- unlimited

  ```
  InternalFailureException
  
  There was an internal failure.
  
  HTTP response code: 500

  ServiceUnavailableException
  
  The service is temporarily unavailable.
  
  HTTP response code: 503

  ThrottlingException
  
  The request was denied due to request throttling.
  
  HTTP response code: 429
type: boolean

If true, message data is kept indefinitely.

- numberOfDays
  type: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The "unlimited" parameter must be false.

- tags
  type: list member: Tag

  Metadata which can be used to manage the data store.

  - key
    type: string; (length- max:256 min:1)

    The tag's key.

  - value
    type: string; (length- max:256 min:1)

    The tag's value.

Output:

```json
{
  "datastoreName": "string",
  "datastoreArn": "string",
  "retentionPeriod": {
    "unlimited": "boolean",
    "numberOfDays": "integer"
  }
}
```

fields:

- datastoreName
  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data store.

- datastoreArn
  type: string

  The ARN of the data store.

- retentionPeriod
  type: RetentionPeriod

  How long, in days, message data is kept for the data store.

- unlimited
  type: boolean

  If true, message data is kept indefinitely.
CreatePipeline

Creates a pipeline. A pipeline consumes messages from one or more channels and allows you to process the messages before storing them in a data store.

**CLI Synopsis:**

```
aws iotanalytics create-pipeline
   --pipeline-name <value>
   --pipeline-activities <value>
   [--tags <value>]
   [--cli-input-json <value>]
   [--generate-cli-skeleton]
```

*cli-input-json format:*

```
{
    "pipelineName": "string",
```
"pipelineActivities": [
   {
      "channel": {
         "name": "string",
         "channelName": "string",
         "next": "string"
      },
      "lambda": {
         "name": "string",
         "lambdaName": "string",
         "batchSize": "integer",
         "next": "string"
      },
      "datastore": {
         "name": "string",
         "datastoreName": "string"
      },
      "addAttributes": {
         "name": "string",
         "attributes": {
            "string": "string"
         },
         "next": "string"
      },
      "removeAttributes": {
         "name": "string",
         "attributes": [
            "string"
         ],
         "next": "string"
      },
      "selectAttributes": {
         "name": "string",
         "attributes": [
            "string"
         ],
         "next": "string"
      },
      "filter": {
         "name": "string",
         "filter": "string",
         "next": "string"
      },
      "math": {
         "name": "string",
         "attribute": "string",
         "math": "string",
         "next": "string"
      },
      "deviceRegistryEnrich": {
         "name": "string",
         "attribute": "string",
         "thingName": "string",
         "roleArn": "string",
         "next": "string"
      },
      "deviceShadowEnrich": {
         "name": "string",
         "attribute": "string",
         "thingName": "string",
         "roleArn": "string",
         "next": "string"
      }
   }
],
"tags": [
CreatePipeline

```json
{
    "key": "string",
    "value": "string"
}
}
```

**fields:**

- **pipelineName**
  
  *type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9-]+)$*

  The name of the pipeline.

- **pipelineActivities**

  *type: list member: PipelineActivity*

  A list of pipeline activities. The list can be 1-25 PipelineActivity objects. Activities perform transformations on your messages, such as removing, renaming, or adding message attributes; filtering messages based on attribute values; invoking your Lambda functions on messages for advanced processing; or performing mathematical transformations to normalize device data.

- **channel**

  *type: ChannelActivity*

  Determines the source of the messages to be processed.

- **name**

  *type: string; (length- max:128 min:1)*

  The name of the 'channel' activity.

- **channelName**

  *type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9-]+)$*

  The name of the channel from which the messages are processed.

- **next**

  *type: string; (length- max:128 min:1)*

  The next activity in the pipeline.

- **lambda**

  *type: LambdaActivity*

  Runs a Lambda function to modify the message.

- **name**

  *type: string; (length- max:128 min:1)*

  The name of the 'lambda' activity.

- **lambdaName**

  *type: string; (length- max:64 min:1); (pattern: ^[a-zA-Z0-9-]+)$*

  The name of the Lambda function that is run on the message.

- **batchSize**
The number of messages passed to the Lambda function for processing. The AWS Lambda function must be able to process all of these messages within five minutes, which is the maximum timeout duration for Lambda functions.

- next
  type: string; (length- max:128 min:1)
  The next activity in the pipeline.

- datastore
  type: DatastoreActivity
  Specifies where to store the processed message data.
  - name
    type: string; (length- max:128 min:1)
    The name of the 'datastore' activity.
  - datastoreName
    type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]$)
    The name of the data store where processed messages are stored.
  - addAttributes
    type: AddAttributesActivity
    Adds other attributes based on existing attributes in the message.
    - name
      type: string; (length- max:128 min:1)
      The name of the 'addAttributes' activity.
    - attributes
      type: map key: AttributeName value: AttributeName
      A list of 1-50 "AttributeNameMapping" objects that map an existing attribute to a new attribute. The existing attributes remain in the message, so if you want to remove the originals, use "RemoveAttributeActivity".
  - next
    type: string; (length- max:128 min:1)
    The next activity in the pipeline.

- removeAttributes
  type: RemoveAttributesActivity
  Removes attributes from a message.
  - name
    type: string; (length- max:128 min:1)
    The name of the 'removeAttributes' activity.
• attributes
  
  *type: list member: AttributeName*

  A list of 1-50 attributes to remove from the message.

• next

  *type: string; (length- max:128 min:1)*

  The next activity in the pipeline.

• selectAttributes

  *type: SelectAttributesActivity*

  Creates a new message using only the specified attributes from the original message.

• name

  *type: string; (length- max:128 min:1)*

  The name of the 'selectAttributes' activity.

• attributes

  *type: list member: AttributeName*

  A list of the attributes to select from the message.

• next

  *type: string; (length- max:128 min:1)*

  The next activity in the pipeline.

• filter

  *type: FilterActivity*

  Filters a message based on its attributes.

• name

  *type: string; (length- max:128 min:1)*

  The name of the 'filter' activity.

• filter

  *type: string; (length- max:256 min:1)*

  An expression that looks like an SQL WHERE clause that must return a Boolean value.

• next

  *type: string; (length- max:128 min:1)*

  The next activity in the pipeline.

• math

  *type: MathActivity*

  Computes an arithmetic expression using the message's attributes and adds it to the message.

• name

  *type: string; (length- max:128 min:1)*
The name of the 'math' activity.

- **attribute**
  
  *type*: string; (length- max:256 min:1)

  The name of the attribute that will contain the result of the math operation.

- **math**
  
  *type*: string; (length- max:256 min:1)

  An expression that uses one or more existing attributes and must return an integer value.

- **next**
  
  *type*: string; (length- max:128 min:1)

  The next activity in the pipeline.

- **deviceRegistryEnrich**

  *type*: DeviceRegistryEnrichActivity

  Adds data from the AWS IoT device registry to your message.

  - **name**
    
    *type*: string; (length- max:128 min:1)

    The name of the 'deviceRegistryEnrich' activity.

  - **attribute**
    
    *type*: string; (length- max:256 min:1)

    The name of the attribute that is added to the message.

  - **thingName**
    
    *type*: string; (length- max:256 min:1)

    The name of the IoT device whose registry information is added to the message.

  - **roleArn**
    
    *type*: string; (length- max:2048 min:20)

    The ARN of the role that allows access to the device's registry information.

  - **next**
    
    *type*: string; (length- max:128 min:1)

    The next activity in the pipeline.

- **deviceShadowEnrich**

  *type*: DeviceShadowEnrichActivity

  Adds information from the AWS IoT Device Shadows service to a message.

  - **name**
    
    *type*: string; (length- max:128 min:1)

    The name of the 'deviceShadowEnrich' activity.
CreatePipeline

- **attribute**
  - `type`: string; (length- max:256 min:1)
  - The name of the attribute that is added to the message.
- **thingName**
  - `type`: string; (length- max:256 min:1)
  - The name of the IoT device whose shadow information is added to the message.
- **roleArn**
  - `type`: string; (length- max:2048 min:20)
  - The ARN of the role that allows access to the device's shadow.
- **next**
  - `type`: string; (length- max:128 min:1)
  - The next activity in the pipeline.
- **tags**
  - `type`: list member: Tag
  - Metadata which can be used to manage the pipeline.
    - **key**
      - `type`: string; (length- max:256 min:1)
      - The tag's key.
    - **value**
      - `type`: string; (length- max:256 min:1)
      - The tag's value.

Output:

```json
{
  "pipelineName": "string",
  "pipelineArn": "string"
}
```

**fields:**

- **pipelineName**
  - `type`: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9-_.]+$)
  - The name of the pipeline.
- **pipelineArn**
  - `type`: string
  - The ARN of the pipeline.

**Errors:**
DeleteChannel

Deletes the specified channel.

**CLI Synopsis:**

```bash
aws iotanalytics delete-channel
   --channel-name <value>
   [--cli-input-json <value>]
   [--generate-cli-skeleton]
```

**cli-input-json format:**

```json
{
   "channelName": "string"
}
```

**fields:**

- **channelName**

  *type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)*

  The name of the channel to delete.

---

- **InvalidRequestException**
  
  The request was not valid.
  
  HTTP response code: 400

- **ResourceAlreadyExistsException**
  
  A resource with the same name already exists.
  
  HTTP response code: 409

- **InternalFailureException**
  
  There was an internal failure.
  
  HTTP response code: 500

- **ServiceUnavailableException**
  
  The service is temporarily unavailable.
  
  HTTP response code: 503

- **ThrottlingException**
  
  The request was denied due to request throttling.
  
  HTTP response code: 429

- **LimitExceededException**
  
  The command caused an internal limit to be exceeded.
  
  HTTP response code: 410
Output:
None

Errors:
- InvalidRequestException
  The request was not valid.
  HTTP response code: 400
- ResourceNotFoundException
  A resource with the specified name could not be found.
  HTTP response code: 404
- InternalFailureException
  There was an internal failure.
  HTTP response code: 500
- ServiceUnavailableException
  The service is temporarily unavailable.
  HTTP response code: 503
- ThrottlingException
  The request was denied due to request throttling.
  HTTP response code: 429

DeleteDataset

Deletes the specified data set. You do not have to delete the content of the data set before you perform this operation.

CLI Synopsis:

```
aws iotanalytics delete-dataset
  --dataset-name <value>
  [--cli-input-json <value>]
  [--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "datasetName": "string"
}
```

fields:
- datasetName
  type: string; (length: max:128 min:1); (pattern: ^[a-zA-Z0-9_]$)
  The name of the data set to delete.
Output:
None
Errors:
• InvalidRequestException
  The request was not valid.
  HTTP response code: 400
• ResourceNotFoundException
  A resource with the specified name could not be found.
  HTTP response code: 404
• InternalFailureException
  There was an internal failure.
  HTTP response code: 500
• ServiceUnavailableException
  The service is temporarily unavailable.
  HTTP response code: 503
• ThrottlingException
  The request was denied due to request throttling.
  HTTP response code: 429

`DeleteDatasetContent`

Deletes the content of the specified data set.

*CLI Synopsis:*

```
aws iotanalytics delete-dataset-content
   --dataset-name <value>
   [--version-id <value>]
   [--cli-input-json <value>]
   [--generate-cli-skeleton]
```

`cli-input-json` format:

```
{
   "datasetName": "string",
   "versionId": "string"
}
```

*fields:*

• datasetName
  
  `type: string; (length: max:128 min:1); (pattern: ^[a-zA-Z0-9-_]+)$`
  
  The name of the data set whose content is deleted.
DeleteDatastore

Deletes the specified data store.

**CLI Synopsis:**

```
aws iotanalytics delete-datastore
  --datastore-name <value>
  [--cli-input-json <value>]
  [--generate-cli-skeleton]
```

**cli-input-json format:**

```json
{
  "datastoreName": "string"
}
```
fields:

- datastoreName
  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)
  The name of the data store to delete.

Output:
None

Errors:

- InvalidRequestException
  The request was not valid.
  HTTP response code: 400
- ResourceNotFoundException
  A resource with the specified name could not be found.
  HTTP response code: 404
- InternalFailureException
  There was an internal failure.
  HTTP response code: 500
- ServiceUnavailableException
  The service is temporarily unavailable.
  HTTP response code: 503
- ThrottlingException
  The request was denied due to request throttling.
  HTTP response code: 429

DeletePipeline

Deletes the specified pipeline.

CLI Synopsis:

```
aws iotanalytics delete-pipeline
  --pipeline-name <value>
  [--cli-input-json <value>]
  [--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "pipelineName": "string"
}
```
fields:

- pipelineName
  
  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)
  
  The name of the pipeline to delete.

Output:

None

Errors:

- InvalidRequestException
  
  The request was not valid.
  
  HTTP response code: 400

- ResourceNotFoundException
  
  A resource with the specified name could not be found.
  
  HTTP response code: 404

- InternalFailureException
  
  There was an internal failure.
  
  HTTP response code: 500

- ServiceUnavailableException
  
  The service is temporarily unavailable.
  
  HTTP response code: 503

- ThrottlingException
  
  The request was denied due to request throttling.
  
  HTTP response code: 429

DescribeChannel

Retrieves information about a channel.

CLI Synopsis:

```
aws iotanalytics describe-channel
  --channel-name <value>
  [--include-statistics | --no-include-statistics]
  [--cli-input-json <value>]
  [--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "channelName": "string",
  "includeStatistics": "boolean"
}
```
DescribeChannel

fields:

- **channelName**
  
  *type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)*
  
  The name of the channel whose information is retrieved.

- **includeStatistics**
  
  *type: boolean*
  
  If true, include statistics about the channel in the response.

Output:

```json
{
  "channel": {
    "name": "string",
    "arn": "string",
    "status": "string",
    "retentionPeriod": {
      "unlimited": "boolean",
      "numberOfDays": "integer"
    },
    "creationTime": "timestamp",
    "lastUpdateTime": "timestamp"
  },
  "statistics": {
    "size": {
      "estimatedOn": "timestamp"
    }
  }
}
```

fields:

- **channel**
  
  *type: Channel*
  
  An object that contains information about the channel.

- **name**
  
  *type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)*
  
  The name of the channel.

- **arn**
  
  *type: string*
  
  The ARN of the channel.

- **status**
  
  *type: string*
  
  The status of the channel. enum: CREATING | ACTIVE | DELETING

- **retentionPeriod**
  
  *type: string*
**type**: RetentionPeriod

How long, in days, message data is kept for the channel.

- unlimited

  **type**: boolean

  If true, message data is kept indefinitely.

- numberOfDays

  **type**: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The "unlimited" parameter must be false.

- creationTime

  **type**: timestamp

  When the channel was created.

- lastUpdateTime

  **type**: timestamp

  When the channel was last updated.

- statistics

  **type**: ChannelStatistics

  Statistics about the channel. Included if the 'includeStatistics' parameter is set to true in the request.

- size

  **type**: EstimatedResourceSize

  The estimated size of the channel.

- estimatedOn

  **type**: timestamp

  The time when the estimate of the size of the resource was made.

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400

- ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404

- InternalFailureException

  There was an internal failure.

  HTTP response code: 500

- ServiceUnavailableException
DescribeDataset

Retrieves information about a data set.

**CLI Synopsis:**

```
aws iotanalytics describe-dataset
  --dataset-name <value>
  [--cli-input-json <value>]
  [--generate-cli-skeleton]
```

**cli-input-json format:**

```json
{
  "datasetName": "string"
}
```

**fields:**

- **datasetName**
  
  **type:** string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)
  
  The name of the data set whose information is retrieved.

**Output:**

```json
{
  "dataset": {
    "name": "string",
    "arn": "string",
    "actions": [
      {
        "actionName": "string",
        "queryAction": {
          "sqlQuery": "string"
        }
      }
    ],
    "triggers": [
      {
        "schedule": {
          "expression": "string"
        }
      },
      {
        "status": "string",
        "creationTime": "timestamp",
        
      }  
    ]
  }
}
```
fields:

- dataset
  type: Dataset
  An object that contains information about the data set.
- name
  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)
  The name of the data set.
- arn
  type: string
  The ARN of the data set.
- actions
  type: list member: DatasetAction
  The "DatasetAction" objects that create the data set.
- actionName
  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)
  The name of the data set action.
- queryAction
  type: SqlQueryDatasetAction
  An "SqlQueryDatasetAction" object that contains the SQL query to modify the message.
- sqlQuery
  type: string
  An SQL query string.
- triggers
  type: list member: DatasetTrigger
  The "DatasetTrigger" objects that specify when the data set is automatically updated.
- schedule
  type: Schedule
  The "Schedule" when the trigger is initiated.
- expression
  type: string
  The expression that defines when to trigger an update. For more information, see Schedule Expressions for Rules in the Amazon CloudWatch documentation.
- status
DescribeDatastore

Retrieves information about a data store.

CLI Synopsis:

```bash
aws iotanalytics describe-datastore
   --datastore-name <value>
   [--include-statistics | --no-include-statistics]
   [--cli-input-json <value>]
   [--generate-cli-skeleton]
```

cli-input-json format:

```
type: string
   The status of the data set. enum: CREATING | ACTIVE | DELETING
   • creationTime
type: timestamp
   When the data set was created.
   • lastUpdateTime
type: timestamp
   The last time the data set was updated.

Errors:
   • InvalidRequestException
      The request was not valid.
      HTTP response code: 400
   • ResourceNotFoundException
      A resource with the specified name could not be found.
      HTTP response code: 404
   • InternalFailureException
      There was an internal failure.
      HTTP response code: 500
   • ServiceUnavailableException
      The service is temporarily unavailable.
      HTTP response code: 503
   • ThrottlingException
      The request was denied due to request throttling.
      HTTP response code: 429
```
DescribeDatastore

```json
{
    "datastoreName": "string",
    "includeStatistics": "boolean"
}
```

**fields:**

- **datastoreName**
  
  *type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)*

  The name of the data store

- **includeStatistics**
  
  *type: boolean*

  If true, include statistics about the data store in the response.

**Output:**

```json
{
    "datastore": {
        "name": "string",
        "arn": "string",
        "status": "string",
        "retentionPeriod": {
            "unlimited": "boolean",
            "numberOfDays": "integer"
        },
        "creationTime": "timestamp",
        "lastUpdateTime": "timestamp"
    },
    "statistics": {
        "size": {
            "estimatedOn": "timestamp"
        }
    }
}
```

**fields:**

- **datastore**
  
  *type: Datastore*

  Information about the data store.

- **name**
  
  *type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)*

  The name of the data store.

- **arn**
  
  *type: string*

  The ARN of the data store.

- **status**
  
  *type: string*
The status of a data store: CREATING The data store is being created. ACTIVE The data store has been created and can be used. DELETING The data store is being deleted. enum: CREATING | ACTIVE | DELETING

• retentionPeriod
  
  type: RetentionPeriod

  How long, in days, message data is kept for the data store.

• unlimited
  
  type: boolean

  If true, message data is kept indefinitely.

• numberOfDays
  
  type: integer java class: java.lang.Integer range- min:1

  The number of days that message data is kept. The "unlimited" parameter must be false.

• creationTime
  
  type: timestamp

  When the data store was created.

• lastUpdateTime
  
  type: timestamp

  The last time the data store was updated.

• statistics
  
  type: DatastoreStatistics

  Statistics about the data store. Included if the 'includeStatistics' parameter is set to true in the request.

• size
  
  type: EstimatedResourceSize

  The estimated size of the data store.

• estimatedOn
  
  type: timestamp

  The time when the estimate of the size of the resource was made.

Errors:

• InvalidRequestException

  The request was not valid.

  HTTP response code: 400

• ResourceNotFoundException

  A resource with the specified name could not be found.

  HTTP response code: 404
• InternalFailureException
  There was an internal failure.
  HTTP response code: 500
• ServiceUnavailableException
  The service is temporarily unavailable.
  HTTP response code: 503
• ThrottlingException
  The request was denied due to request throttling.
  HTTP response code: 429

DescribeLoggingOptions

Retrieves the current settings of the AWS IoT Analytics logging options.

CLI Synopsis:

```
aws iotanalytics describe-logging-options
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
}
```

fields:

Output:

```
{
  "loggingOptions": {
    "roleArn": "string",
    "level": "string",
    "enabled": "boolean"
  }
}
```

fields:

• loggingOptions
  type: LoggingOptions
  The current settings of the AWS IoT Analytics logging options.
• roleArn
  type: string; (length- max:2048 min:20)
  The ARN of the role that grants permission to AWS IoT Analytics to perform logging.
• level
DescribePipeline

Retrieves information about a pipeline.

**CLI Synopsis:**

```
aws iotanalytics describe-pipeline
   --pipeline-name <value>
   [--cli-input-json <value>]
   [--generate-cli-skeleton]
```

**cli-input-json format:**

```
{
   "pipelineName": "string"
}
```

**fields:**

- **type:** string
  
  The logging level. Currently, only "ERROR" is supported. enum: ERROR

- **enabled:**
  
  type: boolean

  If true, logging is enabled for AWS IoT Analytics.

**Errors:**

- **InvalidRequestException**
  
  The request was not valid.

  HTTP response code: 400

- **ResourceNotFoundException**
  
  A resource with the specified name could not be found.

  HTTP response code: 404

- **InternalFailureException**
  
  There was an internal failure.

  HTTP response code: 500

- **ServiceUnavailableException**
  
  The service is temporarily unavailable.

  HTTP response code: 503

- **ThrottlingException**
  
  The request was denied due to request throttling.

  HTTP response code: 429
### DescribePipeline

- **pipelineName**

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9-]_+$)

  The name of the pipeline whose information is retrieved.

**Output:**

```json
{
  "pipeline": {
    "name": "string",
    "arn": "string",
    "activities": [
      {
        "channel": {
          "name": "string",
          "channelName": "string",
          "next": "string"
        },
        "lambda": {
          "name": "string",
          "lambdaName": "string",
          "batchSize": "integer",
          "next": "string"
        },
        "datastore": {
          "name": "string",
          "datastoreName": "string"
        },
        "addAttributes": {
          "name": "string",
          "attributes": {
            "string": "string"
          },
          "next": "string"
        },
        "removeAttributes": {
          "name": "string",
          "attributes": [
            "string"
          ],
          "next": "string"
        },
        "selectAttributes": {
          "name": "string",
          "attributes": [
            "string"
          ],
          "next": "string"
        },
        "filter": {
          "name": "string",
          "filter": "string",
          "next": "string"
        },
        "math": {
          "name": "string",
          "attribute": "string",
          "math": "string",
          "next": "string"
        },
        "deviceRegistryEnrich": {
          "name": "string",
          "attribute": "string"
        }
      }
    ]
  }
}
```
"thingName": "string",
"roleArn": "string",
"next": "string"
},
"deviceShadowEnrich": {
  "name": "string",
  "attribute": "string",
  "thingName": "string",
  "roleArn": "string",
  "next": "string"
}
],
"reprocessingSummaries": [  {
    "id": "string",
    "status": "string",
    "creationTime": "timestamp"
  }
],
"creationTime": "timestamp",
"lastUpdateTime": "timestamp"
}

fields:

• pipeline

  type: Pipeline

A "Pipeline" object that contains information about the pipeline.

• name

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the pipeline.

• arn

  type: string

  The ARN of the pipeline.

• activities

  type: list member: PipelineActivity

  The activities that perform transformations on the messages.

• channel

  type: ChannelActivity

  Determines the source of the messages to be processed.

• name

  type: string; (length- max:128 min:1)

  The name of the 'channel' activity.

• channelName

  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)
The name of the channel from which the messages are processed.

- **next**
  
  *type: string; (length- max:128 min:1)*

  The next activity in the pipeline.

- **lambda**
  
  *type: LambdaActivity*

  Runs a Lambda function to modify the message.

- **name**
  
  *type: string; (length- max:128 min:1)*

  The name of the 'lambda' activity.

- **lambdaName**
  
  *type: string; (length- max:64 min:1); (pattern: ^[a-zA-Z0-9_-]+$)*

  The name of the Lambda function that is run on the message.

- **batchSize**
  
  *type: integer java class: java.lang.Integer range- max:1000 min:1*

  The number of messages passed to the Lambda function for processing. The AWS Lambda function must be able to process all of these messages within five minutes, which is the maximum timeout duration for Lambda functions.

- **next**
  
  *type: string; (length- max:128 min:1)*

  The next activity in the pipeline.

- **datastore**
  
  *type: DatastoreActivity*

  Specifies where to store the processed message data.

- **name**
  
  *type: string; (length- max:128 min:1)*

  The name of the 'datastore' activity.

- **datastoreName**
  
  *type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_-]+$)*

  The name of the data store where processed messages are stored.

- **addAttributes**
  
  *type: AddAttributesActivity*

  Adds other attributes based on existing attributes in the message.

- **name**
  
  *type: string; (length- max:128 min:1)*
The name of the 'addAttributes' activity.

- attributes

  type: map
  key: AttributeName
  value: AttributeName

  A list of 1-50 "AttributeNameMapping" objects that map an existing attribute to a new attribute. The existing attributes remain in the message, so if you want to remove the originals, use "RemoveAttributeActivity".

- next

  type: string
  (length- max:128 min:1)

  The next activity in the pipeline.

- removeAttributes

  type: RemoveAttributesActivity

  Removes attributes from a message.

- name

  type: string
  (length- max:128 min:1)

  The name of the 'removeAttributes' activity.

- attributes

  type: list
  member: AttributeName

  A list of 1-50 attributes to remove from the message.

- next

  type: string
  (length- max:128 min:1)

  The next activity in the pipeline.

- selectAttributes

  type: SelectAttributesActivity

  Creates a new message using only the specified attributes from the original message.

- name

  type: string
  (length- max:128 min:1)

  The name of the 'selectAttributes' activity.

- attributes

  type: list
  member: AttributeName

  A list of the attributes to select from the message.

- next

  type: string
  (length- max:128 min:1)

  The next activity in the pipeline.

- filter

  type: FilterActivity
Filters a message based on its attributes.

- **name**
  
  *type:* string; (length- max:128 min:1)

  The name of the ‘filter’ activity.

- **filter**
  
  *type:* string; (length- max:256 min:1)

  An expression that looks like an SQL WHERE clause that must return a Boolean value.

- **next**
  
  *type:* string; (length- max:128 min:1)

  The next activity in the pipeline.

- **math**
  
  *type:* MathActivity

  Computes an arithmetic expression using the message's attributes and adds it to the message.

  - **name**
    
    *type:* string; (length- max:128 min:1)

    The name of the ‘math’ activity.

  - **attribute**
    
    *type:* string; (length- max:256 min:1)

    The name of the attribute that will contain the result of the math operation.

  - **math**
    
    *type:* string; (length- max:256 min:1)

    An expression that uses one or more existing attributes and must return an integer value.

  - **next**
    
    *type:* string; (length- max:128 min:1)

    The next activity in the pipeline.

- **deviceRegistryEnrich**
  
  *type:* DeviceRegistryEnrichActivity

  Adds data from the AWS IoT device registry to your message.

  - **name**
    
    *type:* string; (length- max:128 min:1)

    The name of the ‘deviceRegistryEnrich’ activity.

  - **attribute**
    
    *type:* string; (length- max:256 min:1)

    The name of the attribute that is added to the message.

  - **thingName**
**DescribePipeline**

- **type**: string; (length: max: 256 min: 1)
  
  The name of the IoT device whose registry information is added to the message.

  - **roleArn**
    - **type**: string; (length: max: 2048 min: 20)
    
    The ARN of the role that allows access to the device's registry information.

  - **next**
    - **type**: string; (length: max: 128 min: 1)
    
    The next activity in the pipeline.

- **deviceShadowEnrich**

  - **type**: DeviceShadowEnrichActivity
    
    Adds information from the AWS IoT Device Shadows service to a message.

  - **name**
    - **type**: string; (length: max: 128 min: 1)
    
    The name of the 'deviceShadowEnrich' activity.

  - **attribute**
    - **type**: string; (length: max: 256 min: 1)
    
    The name of the attribute that is added to the message.

  - **thingName**
    - **type**: string; (length: max: 256 min: 1)
    
    The name of the IoT device whose shadow information is added to the message.

  - **roleArn**
    - **type**: string; (length: max: 2048 min: 20)
    
    The ARN of the role that allows access to the device's shadow.

  - **next**
    - **type**: string; (length: max: 128 min: 1)
    
    The next activity in the pipeline.

- **reprocessingSummaries**

  - **type**: list member: ReprocessingSummary
    
    A summary of information about the pipeline reprocessing.

  - **id**
    - **type**: string
    
    The 'reprocessingId' returned by "StartPipelineReprocessing".

  - **status**
    - **type**: string
The status of the pipeline reprocessing. enum: RUNNING | SUCCEEDED | CANCELLED | FAILED

- **creationTime**
  
  *type*: timestamp

  The time the pipeline reprocessing was created.

- **lastUpdateTime**
  
  *type*: timestamp

  The last time the pipeline was updated.

Errors:

- **InvalidRequestException**
  
  The request was not valid.

  HTTP response code: 400

- **ResourceNotFoundException**
  
  A resource with the specified name could not be found.

  HTTP response code: 404

- **InternalFailureException**
  
  There was an internal failure.

  HTTP response code: 500

- **ServiceUnavailableException**
  
  The service is temporarily unavailable.

  HTTP response code: 503

- **ThrottlingException**
  
  The request was denied due to request throttling.

  HTTP response code: 429

---

**GetDatasetContent**

Retrieves the contents of a data set as pre-signed URIs.

**CLI Synopsis:**

```
aws iotanalytics get-dataset-content
  --dataset-name <value>
  [--version-id <value>]
```
GetDatasetContent

Usage:
[--cli-input-json <value>]
[--generate-cli-skeleton]

cli-input-json format:

```json
{
  "datasetName": "string",
  "versionId": "string"
}
```

fields:
- datasetName
  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)
  The name of the data set whose contents are retrieved.
- versionId
  type: string
  The version of the data set whose contents are retrieved. You can also use the strings "$LATEST" or "$LATEST_SUCCEEDED" to retrieve the contents of the latest or latest successfully completed data set.
  If not specified, "$LATEST_SUCCEEDED" is the default.

Output:

```json
{
  "entries": [
    {
      "entryName": "string",
      "dataURI": "string"
    }
  ],
  "timestamp": "timestamp",
  "status": {
    "state": "string",
    "reason": "string"
  }
}
```

fields:
- entries
  type: list member: DatasetEntry
  A list of "DatasetEntry" objects.
- entryName
  type: string
  The name of the data set item.
- dataURI
  type: string
  The pre-signed URI of the data set item.
ListChannels

Retrieves a list of channels.

CLI Synopsis:
aws iotanalytics list-channels
[--next-token <value>]
[--max-results <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]

cli-input-json format:

```json
{
  "nextToken": "string",
  "maxResults": "integer"
}
```

fields:
- nextToken
  type: string
  The token for the next set of results.
- maxResults
  type: integer java class: java.lang.Integer range: max:250 min:1
  The maximum number of results to return in this request. The default value is 100.

Output:

```json
{
  "channelSummaries": [
  {
    "channelName": "string",
    "status": "string",
    "creationTime": "timestamp",
    "lastUpdateTime": "timestamp"
  }
  ],
  "nextToken": "string"
}
```

fields:
- channelSummaries
  type: list member: ChannelSummary
  A list of "ChannelSummary" objects.
- channelName
  type: string (length: max:128 min:1) (pattern: ^[a-zA-Z0-9_]+$)
  The name of the channel.
- status
  type: string
  The status of the channel. enum: CREATING | ACTIVE | DELETING
- creationTime
**ListDatasets**

Retrieves information about data sets.

**CLI Synopsis:**

```
aws iotanalytics list-datasets
[--next-token <value>]
[--max-results <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

**cli-input-json format:**

```json
{
   "nextToken": "string",
   "maxResults": "integer"
}
```
fields:

- nextToken
  type: string
  The token for the next set of results.

- maxResults
  type: integer java class: java.lang.Integer range- max:250 min:1
  The maximum number of results to return in this request. The default value is 100.

Output:

```
{
  "datasetSummaries": [
    {
      "datasetName": "string",
      "status": "string",
      "creationTime": "timestamp",
      "lastUpdateTime": "timestamp"
    }
  ],
  "nextToken": "string"
}
```

fields:

- datasetSummaries
  type: list member: DatasetSummary
  A list of "DatasetSummary" objects.

- datasetName
  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)
  The name of the data set.

- status
  type: string
  The status of the data set. enum: CREATING | ACTIVE | DELETING

- creationTime
  type: timestamp
  The time the data set was created.

- lastUpdateTime
  type: timestamp
  The last time the data set was updated.

- nextToken
  type: string
  The token to retrieve the next set of results, or null if there are no more results.
Errors:

- **InvalidRequestException**
  - The request was not valid.
  - HTTP response code: 400
- **InternalFailureException**
  - There was an internal failure.
  - HTTP response code: 500
- **ServiceUnavailableException**
  - The service is temporarily unavailable.
  - HTTP response code: 503
- **ThrottlingException**
  - The request was denied due to request throttling.
  - HTTP response code: 429

ListDatastores

Retrieves a list of data stores.

**CLI Synopsis:**

```bash
aws iotanalytics list-datastores
[--next-token <value>]
[--max-results <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

**cli-input-json format:**

```json
{
  "nextToken": "string",
  "maxResults": "integer"
}
```

**fields:**

- **nextToken**
  - *type: string*
  - The token for the next set of results.
- **maxResults**
  - *type: integer java class: java.lang.Integer range- max:250 min:1*
  - The maximum number of results to return in this request. The default value is 100.

Output:
ListDatastores

```json
{
    "datastoreSummaries": [
        {
            "datastoreName": "string",
            "status": "string",
            "creationTime": "timestamp",
            "lastUpdateTime": "timestamp"
        }
    ],
    "nextToken": "string"
}
```

**fields:**

- **datastoreSummaries**
  
  *type*: list member: DatastoreSummary

  A list of "DatastoreSummary" objects.

- **datastoreName**
  
  *type*: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the data store.

- **status**
  
  *type*: string

  The status of the data store. enum: CREATING | ACTIVE | DELETING

- **creationTime**
  
  *type*: timestamp

  When the data store was created.

- **lastUpdateTime**
  
  *type*: timestamp

  The last time the data store was updated.

- **nextToken**
  
  *type*: string

  The token to retrieve the next set of results, or null if there are no more results.

**Errors:**

- **InvalidRequestException**

  The request was not valid.

  HTTP response code: 400

- **InternalFailureException**

  There was an internal failure.

  HTTP response code: 500

- **ServiceUnavailableException**
The service is temporarily unavailable.
HTTP response code: 503
• ThrottlingException
The request was denied due to request throttling.
HTTP response code: 429

ListPipelines

Retrieves a list of pipelines.

CLI Synopsis:

```
aws iotanalytics list-pipelines
[--next-token <value>]
[--max-results <value>]
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```
{
    "nextToken": "string",
    "maxResults": "integer"
}
```

fields:
• nextToken
  type: string
  The token for the next set of results.
• maxResults
  type: integer java class: java.lang.Integer range- max:250 min:1
  The maximum number of results to return in this request. The default value is 100.

Output:

```
{
    "pipelineSummaries": [
        {
            "pipelineName": "string",
            "reprocessingSummaries": [
                {
                    "id": "string",
                    "status": "string",
                    "creationTime": "timestamp"
                }
            ],
            "creationTime": "timestamp",
            "lastUpdateTime": "timestamp"
        }
    ]
}
```
### ListPipelines

```json
{
　　"pipelineSummaries": [
　　　　{ "pipelineName": "string" },
　　　　{ "pipelineName": "string" }
　　],
　　"nextToken": "string"
}
```

**fields:**

- **pipelineSummaries**
  
  ```
  type: list member: PipelineSummary
  A list of "PipelineSummary" objects.
  ```

- **pipelineName**
  
  ```
  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)
  The name of the pipeline.
  ```

- **reprocessingSummaries**
  
  ```
  type: list member: ReprocessingSummary
  A summary of information about the pipeline reprocessing.
  ```

- **id**
  
  ```
  type: string
  The 'reprocessingId' returned by "StartPipelineReprocessing".
  ```

- **status**
  
  ```
  type: string
  The status of the pipeline reprocessing. enum: RUNNING | SUCCEEDED | CANCELLED | FAILED
  ```

- **creationTime**
  
  ```
  type: timestamp
  The time the pipeline reprocessing was created.
  ```

- **creationTime**
  
  ```
  type: timestamp
  When the pipeline was created.
  ```

- **lastUpdateTime**
  
  ```
  type: timestamp
  When the pipeline was last updated.
  ```

- **nextToken**
  
  ```
  type: string
  The token to retrieve the next set of results, or null if there are no more results.
  ```

**Errors:**

- **InvalidRequestException**
  
  The request was not valid.
HTTP response code: 400
• InternalFailureException

There was an internal failure.

HTTP response code: 500
• ServiceUnavailableException

The service is temporarily unavailable.

HTTP response code: 503
• ThrottlingException

The request was denied due to request throttling.

ListTagsForResource

Lists the tags (metadata) which you have assigned to the resource.

**CLI Synopsis:**

```
aws iotanalytics list-tags-for-resource
   --resource-arn <value>
   [--cli-input-json <value>]
   [--generate-cli-skeleton]
```

**cli-input-json format:**

```
{
   "resourceArn": "string"
}
```

**fields:**

• resourceArn

  *type: string; (length- max:2048 min:20)*

  The ARN of the resource whose tags you want to list.

**Output:**

```
{
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

**fields:**


• tags
  
  *type*: list member: Tag

  The tags (metadata) which you have assigned to the resource.

• key
  
  *type*: string; (length- max:256 min:1)

  The tag's key.

• value
  
  *type*: string; (length- max:256 min:1)

  The tag's value.

Errors:

• InvalidRequestException
  
  The request was not valid.

  HTTP response code: 400

• InternalFailureException
  
  There was an internal failure.

  HTTP response code: 500

• ServiceUnavailableException
  
  The service is temporarily unavailable.

  HTTP response code: 503

• ThrottlingException
  
  The request was denied due to request throttling.

  HTTP response code: 429

• LimitExceededExcepcion
  
  The command caused an internal limit to be exceeded.

  HTTP response code: 410

• ResourceNotFoundException
  
  A resource with the specified name could not be found.

  HTTP response code: 404

PutLoggingOptions

Sets or updates the AWS IoT Analytics logging options. Note that if you update the value of any loggingOptions field, it takes up to one minute for the change to take effect. Also, if you change the policy attached to the role you specified in the roleArn field (for example, to correct an invalid policy) it takes up to 5 minutes for that change to take effect.
CLI Synopsis:

```
aws iotanalytics put-logging-options
   --logging-options <value>
   [--cli-input-json <value>]
   [--generate-cli-skeleton]
```

cli-input-json format:

```
{
   "loggingOptions": {
      "roleArn": "string",
      "level": "string",
      "enabled": "boolean"
   }
}
```

fields:

- loggingOptions
  
  type: LoggingOptions

  The new values of the AWS IoT Analytics logging options.

- roleArn

  type: string; (length- max:2048 min:20)

  The ARN of the role that grants permission to AWS IoT Analytics to perform logging.

- level

  type: string

  The logging level. Currently, only "ERROR" is supported. enum: ERROR

- enabled

  type: boolean

  If true, logging is enabled for AWS IoT Analytics.

Output:

None

Errors:

- InvalidRequestException

  The request was not valid.

  HTTP response code: 400

- InternalFailureException

  There was an internal failure.

  HTTP response code: 500

- ServiceUnavailableException

  The service is temporarily unavailable.
HTTP response code: 503
• ThrottlingException

The request was denied due to request throttling.

HTTP response code: 429

RunPipelineActivity

Simulates the results of running a pipeline activity on a message payload.

CLI Synopsis:

```bash
aws iotanalytics run-pipeline-activity
--pipeline-activity <value>
--payloads <value>
[--cli-input-json <value>]
[--generate-cli-skeleton]
```

cli-input-json format:

```json
{
    "pipelineActivity": {
        "channel": {
            "name": "string",
            "channelName": "string",
            "next": "string"
        },
        "lambda": {
            "name": "string",
            "lambdaName": "string",
            "batchSize": "integer",
            "next": "string"
        },
        "datastore": {
            "name": "string",
            "datastoreName": "string"
        },
        "addAttributes": {
            "name": "string",
            "attributes": {
                "string": "string"
            },
            "next": "string"
        },
        "removeAttributes": {
            "name": "string",
            "attributes": [
                "string"
            ],
            "next": "string"
        },
        "selectAttributes": {
            "name": "string",
            "attributes": [
                "string"
            ],
            "next": "string"
        },
        "filter": {
```
### fields:

- **pipelineActivity**
  - *type*: `PipelineActivity`
    
    The pipeline activity that is run. This must not be a 'channel' activity or a 'datastore' activity because these activities are used in a pipeline only to load the original message and to store the (possibly) transformed message. If a 'lambda' activity is specified, only short-running Lambda functions (those with a timeout of less than 30 seconds or less) can be used.

- **channel**
  - *type*: `ChannelActivity`
    
    Determines the source of the messages to be processed.

- **name**
  - *type*: `string`; (length- max:128 min:1)
    
    The name of the 'channel' activity.

- **channelName**
  - *type*: `string`; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)
    
    The name of the channel from which the messages are processed.

- **next**
  - *type*: `string`; (length- max:128 min:1)
    
    The next activity in the pipeline.

- **lambda**

```json
"name": "string",
"filter": "string",
"next": "string"
},
"math": {
  "name": "string",
  "attribute": "string",
  "math": "string",
  "next": "string"
},
"deviceRegistryEnrich": {
  "name": "string",
  "attribute": "string",
  "thingName": "string",
  "roleArn": "string",
  "next": "string"
},
"deviceShadowEnrich": {
  "name": "string",
  "attribute": "string",
  "thingName": "string",
  "roleArn": "string",
  "next": "string"
}
"payloads": [
  "blob"
]
```
**LambdaActivity**

Runs a Lambda function to modify the message.

- **name**

  *type: string* (length- max:128 min:1)

  The name of the 'lambda' activity.

- **lambdaName**

  *type: string* (length- max:64 min:1); (pattern: ^[a-zA-Z0-9-_]+$)

  The name of the Lambda function that is run on the message.

- **batchSize**

  *type: integer java class: java.lang.Integer range- max:1000 min:1*

  The number of messages passed to the Lambda function for processing. The AWS Lambda function must be able to process all of these messages within five minutes, which is the maximum timeout duration for Lambda functions.

- **next**

  *type: string* (length- max:128 min:1)

  The next activity in the pipeline.

- **datastore**

  *type: DatastoreActivity*

  Specifies where to store the processed message data.

- **name**

  *type: string* (length- max:128 min:1)

  The name of the 'datastore' activity.

- **datastoreName**

  *type: string* (length- max:128 min:1); (pattern: ^[a-zA-Z0-9-_]+$)

  The name of the data store where processed messages are stored.

- **addAttributes**

  *type: AddAttributesActivity*

  Adds other attributes based on existing attributes in the message.

- **name**

  *type: string* (length- max:128 min:1)

  The name of the 'addAttributes' activity.

- **attributes**

  *type: map key: AttributeName value: AttributeName*

  A list of 1-50 "AttributeNameMapping" objects that map an existing attribute to a new attribute. The existing attributes remain in the message, so if you want to remove the originals, use "RemoveAttributeActivity".
• next
  type: string; (length- max:128 min:1)
  The next activity in the pipeline.
• removeAttributes
  type: RemoveAttributesActivity
  Removes attributes from a message.
  • name
    type: string; (length- max:128 min:1)
    The name of the 'removeAttributes' activity.
  • attributes
    type: list member: AttributeName
    A list of 1-50 attributes to remove from the message.
• next
  type: string; (length- max:128 min:1)
  The next activity in the pipeline.
• selectAttributes
  type: SelectAttributesActivity
  Creates a new message using only the specified attributes from the original message.
  • name
    type: string; (length- max:128 min:1)
    The name of the 'selectAttributes' activity.
  • attributes
    type: list member: AttributeName
    A list of the attributes to select from the message.
• next
  type: string; (length- max:128 min:1)
  The next activity in the pipeline.
• filter
  type: FilterActivity
  Filters a message based on its attributes.
  • name
    type: string; (length- max:128 min:1)
    The name of the 'filter' activity.
  • filter
    type: string; (length- max:256 min:1)
An expression that looks like an SQL WHERE clause that must return a Boolean value.

- **next**
  
  *type*: string; (length- max:128 min:1)
  
  The next activity in the pipeline.

- **math**
  
  *type*: MathActivity
  
  Computes an arithmetic expression using the message's attributes and adds it to the message.

- **name**
  
  *type*: string; (length- max:128 min:1)
  
  The name of the 'math' activity.

- **attribute**
  
  *type*: string; (length- max:256 min:1)
  
  The name of the attribute that will contain the result of the math operation.

- **math**
  
  *type*: string; (length- max:256 min:1)
  
  An expression that uses one or more existing attributes and must return an integer value.

- **next**
  
  *type*: string; (length- max:128 min:1)
  
  The next activity in the pipeline.

- **deviceRegistryEnrich**
  
  *type*: DeviceRegistryEnrichActivity
  
  Adds data from the AWS IoT device registry to your message.

- **name**
  
  *type*: string; (length- max:128 min:1)
  
  The name of the 'deviceRegistryEnrich' activity.

- **attribute**
  
  *type*: string; (length- max:256 min:1)
  
  The name of the attribute that is added to the message.

- **thingName**
  
  *type*: string; (length- max:256 min:1)
  
  The name of the IoT device whose registry information is added to the message.

- **roleArn**
  
  *type*: string; (length- max:2048 min:20)
  
  The ARN of the role that allows access to the device's registry information.

- **next**
RunPipelineActivity

- **type**: string; (length- max:128 min:1)
  The next activity in the pipeline.

- **deviceShadowEnrich**
  - **type**: DeviceShadowEnrichActivity
  - **name**
    - **type**: string; (length- max:128 min:1)
    The name of the 'deviceShadowEnrich' activity.
  - **attribute**
    - **type**: string; (length- max:256 min:1)
    The name of the attribute that is added to the message.
  - **thingName**
    - **type**: string; (length- max:256 min:1)
    The name of the IoT device whose shadow information is added to the message.
  - **roleArn**
    - **type**: string; (length- max:2048 min:20)
    The ARN of the role that allows access to the device's shadow.
  - **next**
    - **type**: string; (length- max:128 min:1)
    The next activity in the pipeline.
  - **payloads**
    - **type**: list member: MessagePayload
    The sample message payloads on which the pipeline activity is run.

**Output:**

```json
{
  "payloads": ["blob"],
  "logResult": "string"
}
```

**fields**:

- **payloads**
  - **type**: list member: MessagePayload
  The enriched or transformed sample message payloads as base64-encoded strings. (The results of running the pipeline activity on each input sample message payload, encoded in base64.)
- **logResult**
**SampleChannelData**

Retrieves a sample of messages from the specified channel ingested during the specified timeframe. Up to 10 messages can be retrieved.

**CLI Synopsis:**

```
aws iotanalytics sample-channel-data
    --channel-name <value>
    [--max-messages <value>]
    [--start-time <value>]
    [--end-time <value>]
    [--cli-input-json <value>]
    [--generate-cli-skeleton]
```

**cli-input-json format:**

```
{
    "channelName": "string",
    "maxMessages": "integer",
    "startTime": "timestamp",
    "endTime": "timestamp"
}
```

**fields:**

- channelName

  type: string; (length: max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)
The name of the channel whose message samples are retrieved.

- **maxMessages**
  
  *type*: integer java class: java.lang.Integer range: max:10 min:1

  The number of sample messages to be retrieved. The limit is 10, the default is also 10.

- **startTime**
  
  *type*: timestamp

  The start of the time window from which sample messages are retrieved.

- **endTime**
  
  *type*: timestamp

  The end of the time window from which sample messages are retrieved.

**Output:**

```json
{
  "payloads": [
    "blob"
  ]
}
```

**fields:**

- **payloads**

  *type*: list member: MessagePayload

  The list of message samples. Each sample message is returned as a base64-encoded string.

**Errors:**

- **InvalidRequestException**

  The request was not valid.

  HTTP response code: 400

- **ResourceNotFoundException**

  A resource with the specified name could not be found.

  HTTP response code: 404

- **InternalFailureException**

  There was an internal failure.

  HTTP response code: 500

- **ServiceUnavailableException**

  The service is temporarily unavailable.

  HTTP response code: 503

- **ThrottlingException**
The request was denied due to request throttling.

HTTP response code: 429

StartPipelineReprocessing

Starts the reprocessing of raw message data through the pipeline.

**CLI Synopsis:**

```
aws iotanalytics start-pipeline-reprocessing
   --pipeline-name <value>
   [--start-time <value>]
   [--end-time <value>]
   [--cli-input-json <value>]
   [--generate-cli-skeleton]
```

**cli-input-json format:**

```json
{
    "pipelineName": "string",
    "startTime": "timestamp",
    "endTime": "timestamp"
}
```

**fields:**

- **pipelineName**
  
  *type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)*

  The name of the pipeline on which to start reprocessing.

- **startTime**
  
  *type: timestamp*

  The start time (inclusive) of raw message data that is reprocessed.

- **endTime**
  
  *type: timestamp*

  The end time (exclusive) of raw message data that is reprocessed.

**Output:**

```json
{
    "reprocessingId": "string"
}
```

**fields:**

- **reprocessingId**
  
  *type: string*
The ID of the pipeline reprocessing activity that was started.

Errors:

- ResourceNotFoundException
  
  A resource with the specified name could not be found.
  
  HTTP response code: 404

- ResourceAlreadyExistsException
  
  A resource with the same name already exists.
  
  HTTP response code: 409

- InvalidRequestException
  
  The request was not valid.
  
  HTTP response code: 400

- InternalFailureException
  
  There was an internal failure.
  
  HTTP response code: 500

- ServiceUnavailableException
  
  The service is temporarily unavailable.
  
  HTTP response code: 503

- ThrottlingException
  
  The request was denied due to request throttling.
  
  HTTP response code: 429

---

**TagResource**

Adds to or modifies the tags of the given resource. Tags are metadata which can be used to manage a resource.

**CLI Synopsis:**

```bash
aws iotanalytics tag-resource
   --resource-arn <value>
   --tags <value>
   [--cli-input-json <value>]
   [--generate-cli-skeleton]
```

**cli-input-json format:**

```json
{
   "resourceArn": "string",
   "tags": [
   
   }
```
TagResource

```json
"key": "string",
"value": "string"
}
]
}
``` 

**fields:**

- `resourceArn`
  
  *type*: string; (length-max:2048 min:20)

  The ARN of the resource whose tags will be modified.

- `tags`
  
  *type*: list member: Tag

  The new or modified tags for the resource.

- `key`
  
  *type*: string; (length-max:256 min:1)

  The tag’s key.

- `value`
  
  *type*: string; (length-max:256 min:1)

  The tag’s value.

**Output:**

None

**Errors:**

- `InvalidRequestException`

  The request was not valid.

  HTTP response code: 400

- `InternalFailureException`

  There was an internal failure.

  HTTP response code: 500

- `ServiceUnavailableException`

  The service is temporarily unavailable.

  HTTP response code: 503

- `ThrottlingException`

  The request was denied due to request throttling.

  HTTP response code: 429

- `LimitExceededException`

  The command caused an internal limit to be exceeded.
HTTP response code: 410
- ResourceNotFoundException
  A resource with the specified name could not be found.

HTTP response code: 404

**UntagResource**

Removes the given tags (metadata) from the resource.

**CLI Synopsis:**

```
aws iotanalytics untag-resource
  --resource-arn <value>
  --tag-keys <value>
  [--cli-input-json <value>]
  [--generate-cli-skeleton]
```

**cli-input-json format:**

```
{
  "resourceArn": "string",
  "tagKeys": [
    "string"
  ]
}
```

**fields:**

- resourceArn
  
  *type: string; (length- max:2048 min:20)*
  
  The ARN of the resource whose tags will be removed.

- tagKeys
  
  *type: list member: TagKey*
  
  The keys of those tags which will be removed.

**Output:**

None

**Errors:**

- InvalidRequestException
  
  The request was not valid.

  HTTP response code: 400

- InternalFailureException
  
  There was an internal failure.
HTTP response code: 500
• ServiceUnavailableException
  The service is temporarily unavailable.

HTTP response code: 503
• ThrottlingException
  The request was denied due to request throttling.

HTTP response code: 429
• LimitExceededException
  The command caused an internal limit to be exceeded.

HTTP response code: 410
• ResourceNotFoundException
  A resource with the specified name could not be found.

HTTP response code: 404

UpdateChannel

Updates the settings of a channel.

CLI Synopsis:

```
aws iotanalytics update-channel
  --channel-name <value>
  [--retention-period <value>]
  [--cli-input-json <value>]
  [--generate-cli-skeleton]
```

cli-input-json format:

```
{
  "channelName": "string",
  "retentionPeriod": {
    "unlimited": "boolean",
    "numberOfDays": "integer"
  }
}
```

fields:

• channelName
  type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]$)
  The name of the channel to be updated.

• retentionPeriod
  type: RetentionPeriod
How long, in days, message data is kept for the channel.
- unlimited
  type: boolean
  If true, message data is kept indefinitely.
- numberOfDays
  type: integer java class: java.lang.Integer range- min:1
  The number of days that message data is kept. The "unlimited" parameter must be false.

Output:
None

Errors:
- InvalidRequestException
  The request was not valid.
  HTTP response code: 400
- ResourceNotFoundException
  A resource with the specified name could not be found.
  HTTP response code: 404
- InternalFailureException
  There was an internal failure.
  HTTP response code: 500
- ServiceUnavailableException
  The service is temporarily unavailable.
  HTTP response code: 503
- ThrottlingException
  The request was denied due to request throttling.
  HTTP response code: 429

UpdateDataset

Updates the settings of a data set.

CLI Synopsis:

aws iotanalytics update-dataset
  --dataset-name <value>
  --actions <value>
  [--triggers <value>]
  [--cli-input-json <value>]
cli-input-json format:

```json
{
  "datasetName": "string",
  "actions": [
    {
      "actionName": "string",
      "queryAction": {
        "sqlQuery": "string"
      }
    }
  ],
  "triggers": [
    {
      "schedule": {
        "expression": "string"
      }
    }
  ]
}
```

fields:

- **datasetName**
  
  *type*: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)
  
  The name of the data set to update.

- **actions**
  
  *type*: list member: DatasetAction
  
  A list of "DatasetAction" objects. Only one action is supported at this time.

- **actionName**
  
  *type*: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)
  
  The name of the data set action.

- **queryAction**
  
  *type*: SqlQueryDatasetAction
  
  An "SqlQueryDatasetAction" object that contains the SQL query to modify the message.

- **sqlQuery**
  
  *type*: string
  
  An SQL query string.

- **triggers**
  
  *type*: list member: DatasetTrigger
  
  A list of "DatasetTrigger" objects. The list can be empty or can contain up to five DataSetTrigger objects.

- **schedule**
  
  *type*: Schedule
The "Schedule" when the trigger is initiated.

- expression
  
  type: string
  
  The expression that defines when to trigger an update. For more information, see Schedule Expressions for Rules in the Amazon CloudWatch documentation.

Output:
None

Errors:
- InvalidRequestException
  
  The request was not valid.
  
  HTTP response code: 400
- ResourceNotFoundException
  
  A resource with the specified name could not be found.
  
  HTTP response code: 404
- InternalFailureException
  
  There was an internal failure.
  
  HTTP response code: 500
- ServiceUnavailableException
  
  The service is temporarily unavailable.
  
  HTTP response code: 503
- ThrottlingException
  
  The request was denied due to request throttling.
  
  HTTP response code: 429

**UpdateDatastore**

Updates the settings of a data store.

**CLI Synopsis:**

```
aws iotanalytics update-datastore
   --datastore-name <value>
   [--retention-period <value>]
   [--cli-input-json <value>]
   [--generate-cli-skeleton]
```

**cli-input-json format:**

```json
{
```

```
```
UpdateDatastore

```
"datastoreName": "string",
"retentionPeriod": {
  "unlimited": "boolean",
  "numberOfDays": "integer"
}
```

**fields:**

- **datastoreName**
  
  *type*: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9-\_]+$)
  
  The name of the data store to be updated.

- **retentionPeriod**
  
  *type*: RetentionPeriod
  
  How long, in days, message data is kept for the data store.

- **unlimited**
  
  *type*: boolean
  
  If true, message data is kept indefinitely.

- **numberOfDays**
  
  *type*: integer java class: java.lang.Integer range- min:1
  
  The number of days that message data is kept. The "unlimited" parameter must be false.

**Output:**

None

**Errors:**

- **InvalidRequestException**
  
  The request was not valid.
  
  HTTP response code: 400

- **ResourceNotFoundException**
  
  A resource with the specified name could not be found.
  
  HTTP response code: 404

- **InternalFailureException**
  
  There was an internal failure.
  
  HTTP response code: 500

- **ServiceUnavailableException**
  
  The service is temporarily unavailable.
  
  HTTP response code: 503

- **ThrottlingException**
  
  The request was denied due to request throttling.
HTTP response code: 429

UpdatePipeline

Updates the settings of a pipeline.

CLI Synopsis:

```bash
aws iotanalytics update-pipeline
  --pipeline-name <value>
  --pipeline-activities <value>
  [--cli-input-json <value>]
  [--generate-cli-skeleton]
```

cli-input-json format:

```json
{
  "pipelineName": "string",
  "pipelineActivities": [
    {
      "channel": {
        "name": "string",
        "channelName": "string",
        "next": "string"
      },
      "lambda": {
        "name": "string",
        "lambdaName": "string",
        "batchSize": "integer",
        "next": "string"
      },
      "datastore": {
        "name": "string",
        "datastoreName": "string"
      },
      "addAttributes": {
        "name": "string",
        "attributes": {
          "string": "string"
        },
        "next": "string"
      },
      "removeAttributes": {
        "name": "string",
        "attributes": [
          "string"
        ],
        "next": "string"
      },
      "selectAttributes": {
        "name": "string",
        "attributes": [
          "string"
        ],
        "next": "string"
      },
      "filter": {
        "name": "string",
        "filter": "string",
        "next": "string"
      }
    }
  ]
}
```
"math": {  
  "name": "string",  
  "attribute": "string",  
  "math": "string",  
  "next": "string"
},
"deviceRegistryEnrich": {  
  "name": "string",  
  "attribute": "string",  
  "thingName": "string",  
  "roleArn": "string",  
  "next": "string"
},
"deviceShadowEnrich": {  
  "name": "string",  
  "attribute": "string",  
  "thingName": "string",  
  "roleArn": "string",  
  "next": "string"
}
]

fields:

- **pipelineName**

  *type:* string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the pipeline to update.

- **pipelineActivities**

  *type:* list member: PipelineActivity

  A list of "PipelineActivity" objects. The list can be 1-25 PipelineActivity objects. Activities perform transformations on your messages, such as removing, renaming or adding message attributes; filtering messages based on attribute values; invoking your Lambda functions on messages for advanced processing; or performing mathematical transformations to normalize device data.

- **channel**

  *type:* ChannelActivity

  Determines the source of the messages to be processed.

- **name**

  *type:* string; (length- max:128 min:1)

  The name of the 'channel' activity.

- **channelName**

  *type:* string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

  The name of the channel from which the messages are processed.

- **next**

  *type:* string; (length- max:128 min:1)

  The next activity in the pipeline.
type: LambdaActivity

Runs a Lambda function to modify the message.

- name

  type: string; (length- max:128 min:1)

  The name of the 'lambda' activity.

- lambdaName

  type: string; (length- max:64 min:1); (pattern: ^[a-zA-Z0-9_-]+$)

  The name of the Lambda function that is run on the message.

- batchSize

  type: integer java class: java.lang.Integer range- max:1000 min:1

  The number of messages passed to the Lambda function for processing. The AWS Lambda function must be able to process all of these messages within five minutes, which is the maximum timeout duration for Lambda functions.

- next

  type: string; (length- max:128 min:1)

  The next activity in the pipeline.

- datastore

  type: DatastoreActivity

  Specifies where to store the processed message data.

  - name

    type: string; (length- max:128 min:1)

    The name of the 'datastore' activity.

  - datastoreName

    type: string; (length- max:128 min:1); (pattern: ^[a-zA-Z0-9_]+$)

    The name of the data store where processed messages are stored.

  - addAttributes

    type: AddAttributesActivity

    Adds other attributes based on existing attributes in the message.

    - name

      type: string; (length- max:128 min:1)

      The name of the 'addAttributes' activity.

    - attributes

      type: map key: AttributeName value: AttributeName

      A list of 1-50 "AttributeNameMapping" objects that map an existing attribute to a new attribute. The existing attributes remain in the message, so if you want to remove the originals, use "RemoveAttributeActivity".
- next
  type: string; (length- max:128 min:1)
  The next activity in the pipeline.
- removeAttributes
  type: RemoveAttributesActivity
  Removes attributes from a message.
  - name
    type: string; (length- max:128 min:1)
    The name of the 'removeAttributes' activity.
  - attributes
    type: list member: AttributeName
    A list of 1-50 attributes to remove from the message.
- next
  type: string; (length- max:128 min:1)
  The next activity in the pipeline.
- selectAttributes
  type: SelectAttributesActivity
  Creates a new message using only the specified attributes from the original message.
  - name
    type: string; (length- max:128 min:1)
    The name of the 'selectAttributes' activity.
  - attributes
    type: list member: AttributeName
    A list of the attributes to select from the message.
- next
  type: string; (length- max:128 min:1)
  The next activity in the pipeline.
- filter
  type: FilterActivity
  Filters a message based on its attributes.
  - name
    type: string; (length- max:128 min:1)
    The name of the 'filter' activity.
  - filter
    type: string; (length- max:256 min:1)
An expression that looks like an SQL WHERE clause that must return a Boolean value.

- next
  
  type: string; (length- max:128 min:1)
  
The next activity in the pipeline.

- math
  
  type: MathActivity
  
  Computes an arithmetic expression using the message's attributes and adds it to the message.

- name
  
  type: string; (length- max:128 min:1)
  
The name of the 'math' activity.

- attribute
  
  type: string; (length- max:256 min:1)
  
The name of the attribute that will contain the result of the math operation.

- math
  
  type: string; (length- max:256 min:1)
  
  An expression that uses one or more existing attributes and must return an integer value.

- next
  
  type: string; (length- max:128 min:1)
  
The next activity in the pipeline.

- deviceRegistryEnrich
  
  type: DeviceRegistryEnrichActivity
  
  Adds data from the AWS IoT device registry to your message.

- name
  
  type: string; (length- max:128 min:1)
  
The name of the 'deviceRegistryEnrich' activity.

- attribute
  
  type: string; (length- max:256 min:1)
  
The name of the attribute that is added to the message.

- thingName
  
  type: string; (length- max:256 min:1)
  
The name of the IoT device whose registry information is added to the message.

- roleArn
  
  type: string; (length- max:2048 min:20)
  
The ARN of the role that allows access to the device's registry information.

- next
type: string; (length- max:128 min:1)
The next activity in the pipeline.

- deviceShadowEnrich
  
  type: DeviceShadowEnrichActivity
  
  Adds information from the AWS IoT Device Shadows service to a message.
  
  - name
    
    type: string; (length- max:128 min:1)
    
    The name of the 'deviceShadowEnrich' activity.
  
  - attribute
    
    type: string; (length- max:256 min:1)
    
    The name of the attribute that is added to the message.
  
  - thingName
    
    type: string; (length- max:256 min:1)
    
    The name of the IoT device whose shadow information is added to the message.
  
  - roleArn
    
    type: string; (length- max:2048 min:20)
    
    The ARN of the role that allows access to the device's shadow.
  
  - next
    
    type: string; (length- max:128 min:1)
    
    The next activity in the pipeline.

Output:
None

Errors:

- InvalidRequestException
  
  The request was not valid.

  HTTP response code: 400

- ResourceNotFoundException
  
  A resource with the specified name could not be found.

  HTTP response code: 404

- InternalFailureException
  
  There was an internal failure.

  HTTP response code: 500

- ServiceUnavailableException
The service is temporarily unavailable.

HTTP response code: 503

• ThrottlingException

The request was denied due to request throttling.

HTTP response code: 429

• LimitExceedededException

The command caused an internal limit to be exceeded.

HTTP response code: 410