Amazon Personalize
Developer Guide
# Table of Contents

What Is Amazon Personalize? ........................................................................................................... 1
Are You a First-Time Amazon Personalize User? .................................................................................. 1
Are You an Experienced Amazon Personalize User? ............................................................................. 2

How It Works .................................................................................................................................... 3
Amazon Personalize Workflow ............................................................................................................. 3
Datasets and Dataset Groups ............................................................................................................ 4
User Events ..................................................................................................................................... 4
Recipes and Solutions .................................................................................................................... 4
Metrics .......................................................................................................................................... 5
Campaigns ................................................................................................................................. 5
Recommendations .......................................................................................................................... 5

Setting Up Amazon Personalize ........................................................................................................ 7
Sign Up for AWS .................................................................................................................................. 7
Regions and Endpoints ..................................................................................................................... 7
Setting Up Permissions .................................................................................................................... 8
Creating a New IAM Policy ................................................................................................................ 8
Creating an IAM Role for Amazon Personalize .................................................................................. 9

Getting Started ............................................................................................................................... 12
Getting Started Prerequisites ........................................................................................................... 12
Creating the Training Data .................................................................................................................. 12
Getting Started (Console) .................................................................................................................. 13
Getting Started (AWS CLI) ............................................................................................................... 21
Getting Started (AWS SDK for Python) ............................................................................................ 29
Prerequisites ..................................................................................................................................... 29
Getting Started Using Amazon Personalize APIs with Jupyter (iPython) Notebooks ..................... 30

Cleaning Up Resources ..................................................................................................................... 30

Datasets and Schemas ....................................................................................................................... 32
Dataset and Schema Requirements ..................................................................................................... 32
Metadata Fields .................................................................................................................................. 33
Reserved Keywords ........................................................................................................................... 33

Interactions Dataset ........................................................................................................................ 34
Required Interaction Data .................................................................................................................... 34
Impressions Data ............................................................................................................................... 35
Contextual Metadata ......................................................................................................................... 35
Interactions Schema Example ........................................................................................................... 35

Users Dataset .................................................................................................................................. 36
Required User Data ............................................................................................................................ 37
Users Schema Example ...................................................................................................................... 37

Items Dataset ................................................................................................................................... 37
Required Item Data ............................................................................................................................ 38
Creation Timestamp Data .................................................................................................................... 38
Items Schema Example ...................................................................................................................... 38

Schema Examples .............................................................................................................................. 39
Creating a schema using Python ......................................................................................................... 39

Preparing and Importing Data .......................................................................................................... 40
Step 1: Creating a Dataset Group .................................................................................................... 40
Creating a Dataset Group (Console) .................................................................................................. 40
Creating a Dataset Group (AWS CLI) ............................................................................................... 41
Creating a Dataset Group (AWS Python SDK) ................................................................................ 41

Step 2: Creating a Dataset and a Schema ......................................................................................... 42
Creating a Dataset and a Schema (Console) .................................................................................... 42
Creating a Dataset and a Schema (AWS CLI) .................................................................................. 43
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Troubleshooting</td>
<td>136</td>
</tr>
<tr>
<td>Logging and Monitoring</td>
<td>138</td>
</tr>
<tr>
<td>Monitoring</td>
<td>138</td>
</tr>
<tr>
<td>CloudWatch Metrics for Amazon Personalize</td>
<td>141</td>
</tr>
<tr>
<td>Logging Amazon Personalize API Calls with AWS CloudTrail</td>
<td>143</td>
</tr>
<tr>
<td>Compliance Validation</td>
<td>145</td>
</tr>
<tr>
<td>Resilience</td>
<td>145</td>
</tr>
<tr>
<td>Infrastructure Security</td>
<td>145</td>
</tr>
<tr>
<td>Quotas</td>
<td>147</td>
</tr>
<tr>
<td>Supported AWS Regions</td>
<td>147</td>
</tr>
<tr>
<td>Compliance</td>
<td>147</td>
</tr>
<tr>
<td>Service Quotas</td>
<td>147</td>
</tr>
<tr>
<td>Requesting a Quota Increase</td>
<td>149</td>
</tr>
<tr>
<td>API Reference</td>
<td>150</td>
</tr>
<tr>
<td>Actions</td>
<td>150</td>
</tr>
<tr>
<td>Amazon Personalize</td>
<td>151</td>
</tr>
<tr>
<td>Amazon Personalize Events</td>
<td>258</td>
</tr>
<tr>
<td>Amazon Personalize Runtime</td>
<td>264</td>
</tr>
<tr>
<td>Data Types</td>
<td>272</td>
</tr>
<tr>
<td>Amazon Personalize</td>
<td>273</td>
</tr>
<tr>
<td>Amazon Personalize Events</td>
<td>349</td>
</tr>
<tr>
<td>Amazon Personalize Runtime</td>
<td>353</td>
</tr>
<tr>
<td>Common Errors</td>
<td>354</td>
</tr>
<tr>
<td>Common Parameters</td>
<td>356</td>
</tr>
<tr>
<td>Document History</td>
<td>359</td>
</tr>
<tr>
<td>AWS glossary</td>
<td>361</td>
</tr>
</tbody>
</table>
What Is Amazon Personalize?

Amazon Personalize is a machine learning service that makes it easy for developers to add individualized recommendations to customers who use their applications. It reflects the vast experience that Amazon has in building personalization systems.

You can use Amazon Personalize in a variety of scenarios, such as giving users recommendations based on their preferences and behavior, personalized re-ranking of results, and personalizing content for emails and notifications.

Amazon Personalize does not require extensive machine learning experience. You can build, train, and deploy a solution version (a trained Amazon Personalize recommendation model) with the AWS console or programmatically by using the AWS SDK. As the developer, you only need to do the following:

- Format input data and upload the data into an Amazon S3 bucket, or send real-time event data.
- Select a training recipe (algorithm) to use on the data.
- Train a solution version using the recipe.
- Deploy the solution version.

Amazon Personalize can capture live events from your users to achieve real-time personalization. Amazon Personalize can blend real-time user activity data with existing user profile and item information to recommend the most relevant items, according to the user's current session and activity. You can also use Amazon Personalize to collect data for new properties, such as a brand new website, and after enough data has been collected, Amazon Personalize can start to make recommendations.

To give recommendations to your users, call one of the recommendation APIs, and then create personalized experiences for them.

Amazon Personalize can improve its recommendations over time as new user activity data is collected. For example, a new movie rental event by a user can result in better movie recommendations.

Amazon Personalize can provide recommendations based on a user's browsing context. For example, Amazon Personalize can provide different recommendations when a user is browsing on a mobile device than when that same user is browsing on a desktop.

With Amazon Personalize you can train a solution for different use cases. For example, user personalization, items related to an item, and re-ranking of items. You choose a recipe based on your use case and provide the input data. A recipe performs featurization of your data, and applies a choice of learning algorithms, along with default hyperparameters, and hyperparameter optimization job configuration.

Recipes in Amazon Personalize allow you to create custom personalization models without needing machine learning expertise. You can choose which recipe to use to train a solution version, or let Amazon Personalize decide on the best recipe to use for your data. To help you decide which recipe to use, Amazon Personalize provides extensive metrics on the performance of a trained solution version.

Are You a First-Time Amazon Personalize User?

If you are a first-time user of Amazon Personalize, we recommend that you read the following sections in order:
Are You an Experienced Amazon Personalize User?

If you are an experienced Amazon Personalize user, you can find in-depth tutorials and code samples in the amazon-personalize-samples GitHub repository.
How It Works

To make recommendations, Amazon Personalize uses a machine learning model that is trained with your data. The data used to train the model is stored in related datasets in a dataset group. Each model is trained by using a recipe that contains an algorithm for a specific use case. In Amazon Personalize, a trained model is known as a solution version. A solution version is deployed for use in a campaign. Users of your applications can receive recommendations through the campaign. For example, a campaign can show movie recommendations on a website or application where the title shown is based on viewing habits that were part of the dataset.

A dataset can grow over time and your models can be retrained on the new data. The data can come from new metadata and the consumption of real-time user event data. In the previous movie recommendations example, you could add new movies as they are released, and add a movie that is chosen by the signed-in user.

Amazon Personalize has an AWS console that you can use to create, manage, and deploy solution versions. Alternatively, you can use the AWS Command Line Interface (AWS CLI) or one of the Amazon Personalize SDKs.

Amazon Personalize consists of three related components:

- Amazon Personalize – Use this to create, manage, and deploy solution versions.
- Amazon Personalize Events – Use this to record events (p. 350) to add to your training data. For more information, see Recording Events (p. 58).
- Amazon Personalize Runtime – Use this to get recommendations from a campaign (deployed solution version). For more information, see Getting Recommendations (p. 105).

Topics

- Amazon Personalize Workflow (p. 3)
- Datasets and Dataset Groups (p. 4)
- User Events (p. 4)
- Recipes and Solutions (p. 4)
- Metrics (p. 5)
- Campaigns (p. 5)
- Recommendations (p. 5)

Amazon Personalize Workflow

The workflow for training, deploying, and getting recommendations from a campaign is:

1. Create related datasets and a dataset group.
2. Get training data.
   - Import historical data to the dataset group.
   - Record user events to the dataset group.
3. Create a solution version (trained model) using a recipe.
4. Evaluate the solution version using metrics.
5. Create a campaign (deploy the solution version).
6. Provide recommendations for users.
The following sections provide a brief overview of the above workflow. Each section includes a link to the main topic that describes the step in depth and which provides a Python example.

The Getting Started (p. 12) guides provide step-by-step procedures using the Amazon Personalize console, the AWS CLI, and a Jupyter (IPython) notebook.

## Datasets and Dataset Groups

Amazon Personalize requires data, stored in Amazon Personalize datasets, in order to train a model.

There are two ways to provide the training data. You can import historical data from an Amazon S3 bucket, and you can record event data as it is created.

A dataset group contains related datasets. You can create three types of historical datasets: Users, Items, and Interactions. A dataset group can contain only one of each kind of dataset and at minimum must contain an Interactions dataset.

You can create dataset groups to serve different purposes. For example, you might have an application that provides recommendations for purchasing shoes and another that provides recommendations for places to visit in Europe. In Amazon Personalize, each application would have its own dataset group.

Historical data must be provided in a CSV file. Each dataset type has a unique schema that specifies the contents of the CSV file.

The minimum data requirements to train a model are as follows:

- 1000 records of combined interaction data (after filtering by `eventType` and `eventValueThreshold`, if provided).
- 25 unique users with at least 2 interactions each.

**Note**

Using existing data allows you to immediately start training a model. If you rely on recorded data as it is created, and there is no historical data, it can take a while before training can begin.

For more information, see Preparing and Importing Data (p. 40).

## User Events

Amazon Personalize can consume real time user events (p. 350) to be used for model training either alone or combined with historical data.

For more information, see Recording Events (p. 58).

## Recipes and Solutions

After enough data is available in the interactions datasets (historical and live events), the data can be used to train a model. A trained model is known as a solution version. The model is trained using a recipe. The recipes available in Amazon Personalize are made of an algorithm and the data processing steps that optimize a solution for a certain type of recommendation based on your input data.

Amazon Personalize supports a number of predefined recipes. Amazon Personalize can automatically choose the most appropriate recipe based on its analysis of the training data. Alternatively, you can
choose which recipe to train the model with. Each recipe has its own use case and you should choose the recipe that best fits your needs.

Each time you train a model, it is assigned a new solution version. Use the solution version ARN to identify which version of the solution you want to use for your campaign.

For more information, see Creating a Solution (p. 64).

Metrics

After you have created your solution version, you evaluate the metrics that were created during training. The metrics give an indication of the solution version's performance. The console shows you the metrics and allows you to create a new solution version, as necessary. Alternatively, you can call the GetSolutionMetrics (p. 226) API. Typically, you train your model with multiple recipes and use the recipe that results in the metrics that show the best performance. After you have created a solution version based on your chosen recipe, the solution version is ready for deployment as a campaign.

For more information, see Step 4: Evaluating the Solution Version (p. 101).

Campaigns

A deployed solution version is known as a campaign. A campaign allows Amazon Personalize to make recommendations for your users. To deploy a solution version, you create a campaign in the console or by calling the CreateCampaign (p. 157) API. You choose which version of the solution to use.

For more information, see Creating a Campaign (p. 104).

Recommendations

After you create a campaign, you are able to get recommendations in real-time or as part of a batch workflow with purely historical data. For more information, see Getting Recommendations (p. 105).

Real-Time Recommendations

Get real-time recommendations in situations where you want to update recommendations as customers use your application. For example, say you provide movie recommendations to users signed into your application, and you want recommendations to update as they choose different movies.

You are able to get two different types of real-time recommendations:

- For user-personalization and related-items recipes, use the GetRecommendations (p. 269) API to get a list of recommended items. For example, movies can be recommended for users who are signed-in to a website.
- For personalized-ranking recipes, use the GetPersonalizedRanking (p. 265) API to have Amazon Personalize re-rank a list of recommended items based on a specified query.

For more information see Getting Real-Time Recommendations (p. 105).

Batch Recommendations

Get batch recommendations in situations where you have large datasets that do not require real-time updates. For instance, you might create a batch inference job to get product recommendations for all users on an email list, or to get item-to-item similarities (SIMS) (p. 89) across an inventory.
For more information see Getting Batch Recommendations (p. 108).
Setting Up Amazon Personalize

Before using Amazon Personalize, you must have an Amazon Web Services (AWS) account. After you have an AWS account, you can access Amazon Personalize through the Amazon Personalize console, the AWS Command Line Interface (AWS CLI), or the AWS SDKs.

This guide includes examples for AWS CLI, Python, and JavaScript with AWS Amplify.

Topics
• Sign Up for AWS (p. 7)
• Regions and Endpoints (p. 7)
• Setting Up Permissions (p. 8)
• Setting Up the AWS CLI (p. 10)
• Setting Up the AWS SDKs (p. 11)

Sign Up for AWS

When you sign up for Amazon Web Services (AWS), your account is automatically signed up for all services in AWS, including Amazon Personalize. You are charged only for the services that you use.

If you have an AWS account already, skip to the next task. If you don't have an AWS account, use the following procedure to create one.

To sign up for AWS

1. Open https://aws.amazon.com, and then choose Create an AWS Account.
2. Follow the on-screen instructions to complete the account creation. Note your 12-digit AWS account number. Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

An IAM user with administrator permissions has unrestricted access to the AWS services in your account. For information about restricting access to Amazon Personalize operations, see Amazon Personalize Identity-Based Policies (p. 131)
4. Create an IAM user for use with Amazon Personalize. The account requires certain permissions. For more information, see Setting Up Permissions (p. 8).

Regions and Endpoints

An endpoint is a URL that is the entry point for a web service. Each endpoint is associated with a specific AWS region. Pay attention to the default regions of the Amazon Personalize console, the AWS CLI, and the Amazon Personalize SDKs, as all Amazon Personalize components of a given campaign (dataset, solution, campaign, event tracker) must be created in the same region. For the regions and endpoints supported by Amazon Personalize, see Regions and Endpoints.
Setting Up Permissions

To use Amazon Personalize, you have to set up permissions that allow IAM users to access the Amazon Personalize console and API operations. You also have to set up permissions that allow Amazon Personalize to perform tasks on your behalf and to access resources that you own.

We recommend creating an AWS Identity and Access Management (IAM) user with access restricted to Amazon Personalize operations. You can add other permissions as needed. For more information, see Amazon Personalize Identity-Based Policies (p. 131).

**Note**
We recommend creating a new IAM policy (p. 8) that grants only the permissions necessary to use Amazon Personalize.

**To set up permissions**

1. Attach a policy to your Amazon Personalize IAM user or group that allows full access to Amazon Personalize.
   - Create a new IAM policy and attach it to your IAM user or group (see Creating a New IAM Policy (p. 8)).
   - or
   - Attach the `AmazonPersonalizeFullAccess` AWS managed policy to your IAM user or group (see AWS Managed Policies (p. 134)).
2. Attach the `AmazonS3FullAccess` AWS managed policy to your user or group to grant permissions to access Amazon S3 and create an Amazon S3 bucket. For more information on granting permission to your Amazon S3 resources see Using Bucket Policies and User Policies in the Amazon S3 Developer Guide.
3. Optionally attach the `CloudWatchFullAccess` AWS managed policy to your IAM user or group to grant permissions to monitor Amazon Personalize with CloudWatch. See AWS Managed Policies (p. 134).
4. Create an IAM role for Amazon Personalize and attach the policy from step 1 to the new role. See Creating an IAM Role for Amazon Personalize (p. 9).
5. If you are using AWS Key Management Service (AWS KMS) for encryption, give your IAM user and Amazon Personalize IAM service role permission to use your key using a key policy. For more information see Using key policies in AWS KMS in the AWS Key Management Service Developer Guide.

Creating a New IAM Policy

Create an IAM policy that provides users and Amazon Personalize full access to your Amazon Personalize resources. Then attach the policy to your IAM user or group.

**To create and attach an IAM policy**

2. In the navigation pane, choose Policies.
3. Choose Create policy.
4. Choose the JSON tab.
5. Paste following JSON policy document in the text field.

```json
{
}
```
Creating an IAM Role for Amazon Personalize

In the following procedure, you create an IAM role that allows Amazon Personalize to access your resources and perform tasks on your behalf.

2. In the navigation pane, choose Roles.
3. Choose Create role.
4. For Select type of trusted entity, choose AWS service.
5. For Choose the service that will use this role, choose Amazon Personalize.
6. Choose Next: Permissions.
7. For Attach permissions policies, either choose the policy you created in Creating a New IAM Policy (p. 8) or choose AmazonPersonalizeFullAccess (see AWS Managed Policies (p. 134)).
   a. To display the policy in the list, type part of the policy name in the Filter policies query filter.
   b. Choose the check box next to the policy name.
8. Choose Next: Tags. You don't need to add any tags, so choose Next: Review.
9. In the **Review** section, for **Role name**, enter a name for the role (for example, `PersonalizeRole`). Update the description for the role in **Role description**, then choose **Create role**.

10. Choose the new role to open the role’s summary page.

11. Copy the **Role ARN** value and save it. You need it to import a dataset into Amazon Personalize.

12. If you are using AWS KMS for encryption, give your Amazon Personalize service role permission to use your key. For more information see [Using key policies in AWS KMS](https://docs.aws.amazon.com/kms/latest/developerguide/key-policies.html) in the **AWS Key Management Service Developer Guide**.

Next, if you are completing the getting started exercise, you are ready create your training data and grant Amazon Personalize access to your Amazon S3 bucket. See [Creating the Training Data](p. 12).

If you are not completing the getting started exercise, you are ready to import your data. See [Preparing and Importing Data](p. 40).

---

### Setting Up the AWS CLI

The AWS Command Line Interface (AWS CLI) is a unified developer tool for managing AWS services, including Amazon Personalize. We recommend that you install it.

1. To install the AWS CLI, follow the instructions in [Installing the AWS Command Line Interface](https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-install.html) in the **AWS Command Line Interface User Guide**.

2. To configure the AWS CLI and set up a profile to call the AWS CLI, follow the instructions in [Configuring the AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html) in the **AWS Command Line Interface User Guide**.

3. To confirm that the AWS CLI profile is configured properly, run the following command.

   ```bash
   aws configure --profile default
   ```

   If your profile has been configured correctly, you will see output similar to the following.

   ```
   AWS Access Key ID [****************52FQ]:
   AWS Secret Access Key [****************xgyZ]:
   Default region name [us-west-2]:
   Default output format [json]:
   ```

4. To verify that the AWS CLI is configured for use with Amazon Personalize, run the following commands.

   ```bash
   aws personalize help
   ```

   and

   ```bash
   aws personalize-runtime help
   ```

   and

   ```bash
   aws personalize-events help
   ```

   If the AWS CLI is configured correctly, you will see a list of the supported AWS CLI commands for Amazon Personalize, Amazon Personalize runtime, and Amazon Personalize events.

   If you set up the AWS CLI and it doesn't recognize the commands for Amazon Personalize, update the AWS CLI. To update the AWS CLI, run the following command.

```bash
aws update-profile --profile default
```
Setting Up the AWS SDKs

Download and install the AWS SDKs that you want to use. This guide provides examples for Python, and JavaScript using the AWS Amplify library. For information about other AWS SDKs, see Tools for Amazon Web Services.

- AWS SDK for Python (Boto3)
  
  To confirm that your Python environment is configured correctly for use with Amazon Personalize, see Getting Started (AWS SDK for Python) (p. 29).

- AWS Amplify

For more information, see Installing the AWS CLI Using pip.
Getting Started

This getting started guide shows you how to create a campaign that returns movie recommendations for a user, based on historical data that consists of 100,000 movie ratings on 9,700 movies from 600 users.

To simplify this guide:

- We rely on the fact that a user saw a movie and not on what they rated the movie. This simplifies the preparation of the training data.
- We don’t record live user interaction events. For information on capturing user events, see Recording Events (p. 58).

To begin, download and prepare the training data. Next, create an AWS Identity and Access Management (IAM) role that allows Amazon Personalize to access the data on your behalf. After creating the training data and role, proceed to either Getting Started (Console) (p. 13) or Getting Started (AWS CLI) (p. 21).

When you finish the getting started exercise, to avoid incurring unnecessary charges, follow the steps in Cleaning Up Resources (p. 30) to delete the resources you created.

Topics

- Getting Started Prerequisites (p. 12)
- Getting Started (Console) (p. 13)
- Getting Started (AWS CLI) (p. 21)
- Getting Started (AWS SDK for Python) (p. 29)
- Cleaning Up Resources (p. 30)

Getting Started Prerequisites

The following steps are prerequisites for the getting started exercises.

1. Create an AWS account and an AWS Identity and Access Management user, as specified in Sign Up for AWS (p. 7).
2. Create an IAM policy that provides users and Amazon Personalize full access to your Amazon Personalize resources. Then attach the policy to your Amazon Personalize user or group. See Creating a New IAM Policy (p. 8).
3. Create an AWS Identity and Access Management (IAM) service role, as specified in Creating an IAM Role for Amazon Personalize (p. 9). Use the role ARN when you upload the movie training data.
4. Prepare your training data and upload the data to your Amazon S3 bucket, as specified in Creating the Training Data (p. 12). Use the name of the Amazon S3 bucket when you upload the movie training data.
5. Give your Amazon Personalize service role permission to access your Amazon S3 resources, as specified in Giving Amazon Personalize Access to Amazon S3 Resources (p. 49).

Creating the Training Data

To create training data, download, modify, and save the movie ratings data to an Amazon Simple Storage Service (Amazon S3) bucket. Then give Amazon Personalize permission to read from the bucket.
1. Download the movie ratings zip file, ml-latest-small.zip from MovieLens (under recommended for education and development). Unzip the file. The user-interactions data is in the file named ratings.csv.

2. Open the ratings.csv file.
   a. Delete the rating column.
   b. Replace the header row with the following:

      USER_ID,ITEM_ID,TIMESTAMP

      These headers must be exactly as shown for Amazon Personalize to recognize the data.

      Save the ratings.csv file.

3. Upload ratings.csv to your Amazon S3 bucket. For more information, see Uploading Files and Folders by Using Drag and Drop in the Amazon Simple Storage Service Console User Guide.

4. Give Amazon Personalize permission to read the data in the bucket. For more information, see Giving Amazon Personalize Access to Amazon S3 Resources (p. 49).

Getting Started (Console)

In this exercise, you use the Amazon Personalize console to create a campaign that returns movie recommendations for a given user.

Before you start this exercise, review the Getting Started Getting Started Prerequisites (p. 12).

When you finish the getting started exercise, to avoid incurring unnecessary charges, follow the steps in Cleaning Up Resources (p. 30) to delete the resources you created.

Step 1: Import Training Data

In this procedure, you first create a dataset group. Next, you create an Amazon Personalize user-item interaction dataset in the dataset group and a schema to match your training data. Next, you import your training data into the dataset.

To import training data

1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/ and sign in to your account.

2. Choose Create dataset group.

3. If this is your first time using Amazon Personalize, on the Create dataset group page, in New dataset group, choose Get started.

4. In Dataset group details, for Dataset group name, specify a name for your dataset group. Your screen should look similar to the following:
5. Choose **Next**.

6. On the **Create user-item interaction data** page, in **Dataset details**, for **Dataset name**, specify a name for your dataset.

7. In **Schema details**, for **Schema selection**, choose **Create new schema**. A minimal Interactions schema is displayed in the **Schema definition** field. The schema matches the headers you previously added to the **ratings.csv** file. For more information see **Creating the Training Data (p. 12)**.

8. For **New schema name**, specify a name for the new schema.

Your screen should look similar to the following:
Dataset details

Dataset name
The name you enter here can help you distinguish this dataset import job from others.

ratings-dataset

The dataset name must have 2-256 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and .:+@-%- (hyphen).

Schema details

Schema selection
- Use existing schema
  Choose an existing schema that matches your dataset.
- Create new schema
  Create a new schema to match your dataset

New schema name
The name you enter here appears in the Schema dashboard. It can help you distinguish this schema from others.

ratings-schema

The schema name must have 2-256 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and .:+@-%- (hyphen).

Schema definition
Ensure your dataset’s schema matches the following schema.

```json
{
  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
    {
      "name": "USER_ID",
      "type": "string"
    },
    {
      "name": "ITEM_ID",
      "type": "string"
    },
    {
      "name": "TIMESTAMP",
      "type": "long"
    },
    "version": "1.0"
  ]
}
```
9. Choose Next.
10. On the Import user-item interaction data page, in Dataset import job details, for Dataset import job name, specify a name for your import job.
11. For IAM service role, keep the default selection of Enter a custom IAM role ARN.
12. For Custom IAM role ARN, specify the role that you created in Creating an IAM Role for Amazon Personalize (p. 9).
13. In the informational dialog box named Additional S3 bucket policy required, follow the instructions (p. 48) to add the required Amazon S3 bucket policy.
14. For Data location, specify where your movie data file is stored in Amazon Simple Storage Service (S3). Use the following syntax:

```
s3://<name of your S3 bucket>/<folder path>/<CSV filename>
```

Your screen should look similar to the following:
15. Choose Finish. The data import job starts and the Dashboard Overview page is displayed.

16. Initially, in Upload datasets, the User-item interaction data status is Create pending (followed by Create in progress), and the Create solutions - Start button is disabled.

   **Note**
   The time it takes for the data to be imported depends on the size of the dataset.

When the data import job has finished, the User-item interaction data status changes to Active and the Create solutions - Start button is enabled. Your screen should look similar to the following:

17. After the import job has finished, choose the Create solutions - Start button. The Create solution page is displayed.

**Step 2: Create a Solution**

In this procedure, you use the dataset that you imported in the previous step to train a model. A trained model is referred to as a solution version.

**To create a solution**

1. If the Create solution page is not already displayed, in the navigation pane, under the dataset group that you created, choose the Solution creation Start button.

2. For Solution name, specify a name for your solution.

3. For Recipe, choose aws-user-personalization. Leave the optional Solution configuration fields unchanged.

Your screen should look similar to the following:
4. Choose **Next** to display the **Create solution version** screen.

Your screen should look similar to the following:

5. There's no need to modify the **Solution config**, so choose **Finish**. Model training starts and the **Dashboard Overview** page is displayed.

6. Initially, in **Create solutions**, the **Solution creation** status is **Create pending** (followed by **Create in progress**), the **Launch campaigns - Start** button is disabled, and a banner is displayed on the top of the console showing the progress.

   **Note**
   The time it takes to train a model depends on the size of the dataset and the chosen recipe.

7. After training has finished, in the navigation pane choose **Dashboard** and choose **Create new campaign**.

**Step 3: Create a Campaign**

In this procedure, you create a campaign by deploying the solution version you created in the previous step.
To create a campaign

1. If the Create new campaign page is not already displayed, in the navigation pane, in the dataset group that you created, choose Dashboard, and then choose Create new campaign.

2. In Campaign details, for Campaign name, specify a name for your campaign.

3. For Solution, choose the solution you created in the previous step and for Solution version ID keep the default.

4. For Minimum provisioned transactions per second, keep the default of 1.

Your screen should look similar to the following:

5. Choose Create campaign. Campaign creation starts and the Campaign page appears with the Campaign inference section displayed.

Your screen should look similar to the following:

Note
Creating a campaign takes time.

After the campaign is created, the page is updated to show the Test campaign results section. Your screen should look similar to the following:
Step 4: Get Recommendations

In this procedure, use the campaign that you created in the previous step to get recommendations.

To get recommendations

1. In Test campaign results, for User ID, specify a value from the ratings dataset, for example, 83. For Filter name keep the default selection of None.
2. Choose Get recommendations. The Recommended item ID list displays the recommended item IDs.

Your screen should look similar to the following:
Getting Started (AWS CLI)

In this exercise, you use the AWS Command Line Interface (AWS CLI) to explore Amazon Personalize. You create a campaign that returns movie recommendations for a given user ID.

Before you start this exercise, do the following:

- Review the Getting Started Getting Started Prerequisites (p. 12).
- Set up the AWS CLI, as specified in Setting Up the AWS CLI (p. 10).

When you finish the getting started exercise, to avoid incurring unnecessary charges, follow the steps in Cleaning Up Resources (p. 30) to delete the resources you created.

**Note**

The CLI commands in this exercise were tested on Linux. For information about using the CLI commands on Windows, see Specifying Parameter Values for the AWS Command Line Interface in the AWS Command Line Interface User Guide.

**Step 1: Import Training Data**

Follow the steps to create a dataset group, add a dataset to the group, and then populate the dataset using the movie ratings data.

1. Create a dataset group by running the following command. You can encrypt the dataset group by passing a AWS Key Management Service key ARN and the ARN of an IAM role that has access permissions to that key as input parameters. For more information about the API, see CreateDatasetGroup (p. 163).
aws personalize create-dataset-group --name MovieRatingDatasetGroup  --kms-key-arn arn:aws:kms:us-west-2:01234567890:key/1682a1e7-a94d-4d92-bbdf-837d3b62315e --role-arn arn:aws:iam::01234567890:KMS-key-access

The dataset group ARN is displayed, for example:

```
{
  "datasetGroupArn": "arn:aws:personalize:us-west-2:acct-id:dataset-group/MovieRatingDatasetGroup"
}
```

Use the describe-dataset-group command to display the dataset group you created, specifying the returned dataset group ARN.

```
aws personalize describe-dataset-group  
```

The dataset group and its properties are displayed, for example:

```
{
  "datasetGroup": {
    "name": "MovieRatingDatasetGroup",
    "status": "ACTIVE",
    "creationDateTime": 1542392161.262,
    "lastUpdatedDateTime": 1542396513.377
  }
}
```

**Note**

Wait until the dataset group’s status shows as ACTIVE before creating a dataset in the group. This operation is usually quick.

If you don’t remember the dataset group ARN, use the list-dataset-groups command to display all the dataset groups that you created, along with their ARNs.

```
aws personalize list-dataset-groups
```

**Note**

The describe-object and list-objects commands are available for most Amazon Personalize objects. These commands are not shown in the remainder of this exercise but they are available.

2. Create a schema file in JSON format by saving the following code to a file named MovieRatingSchema.json. The schema matches the headers you previously added to ratings.csv. The schema name is Interactions, which matches one of the three types of datasets recognized by Amazon Personalize. For more information, see Datasets and Dataset Groups (p. 4).

```json
{
  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
```
3. Create a schema by running the following command. Specify the file you saved in the previous step. The example shows the file as belonging to the current folder. For more information about the API, see CreateSchema (p. 174).

```
aws personalize create-schema \
--name MovieRatingSchema \
--schema file://MovieRatingSchema.json
```

The schema Amazon Resource Name (ARN) is displayed, for example:

```
{ 
    "schemaArn": "arn:aws:personalize:us-west-2:acct-id:schema/MovieRatingSchema"
}
```

4. Create an empty dataset by running the following command. Provide the dataset group ARN and schema ARN that were returned in the previous steps. The dataset-type must match the schema name from the previous step. For more information about the API, see CreateDataset (p. 160).

```
aws personalize create-dataset \
--name MovieRatingDataset \
--dataset-group-arn arn:aws:personalize:us-west-2:acct-id:dataset-group/MovieRatingDatasetGroup \
--dataset-type Interactions \
```

The dataset ARN is displayed, for example:

```
{ 
    "datasetArn": "arn:aws:personalize:us-west-2:acct-id:dataset/MovieRatingDatasetGroup/INTERACTIONS"
}
```

5. Add the training data to the dataset.

   a. Create a dataset import job by running the following command. Supply the AWS Identity and Access Management (IAM) role ARN you created in Creating an IAM Role for Amazon Personalize (p. 9). For more information about the API, see CreateDatasetImportJob (p. 166).

```
aws personalize create-dataset-import-job \
--job-name MovieRatingImportJob \
--dataset-arn arn:aws:personalize:us-west-2:acct-id:dataset/MovieRatingDatasetGroup/INTERACTIONS \
--data-source dataLocation=s3://bucketname/ratings.csv \
```
b. Check the status by using the describe-dataset-import-job command. Provide the dataset import job ARN that was returned in the previous step. For more information about the API, see DescribeDatasetImportJob (p. 208).

```bash
```

The properties of the dataset import job, including its status, are displayed. Initially, the status shows as CREATE PENDING, for example:

```json
{
   "datasetImportJob": {
      "jobName": "MovieRatingImportJob",
      "dataSource": { "dataLocation": "s3://<bucketname>/ratings.csv" },
      "roleArn": "role-arn",
      "status": "CREATE PENDING",
      "creationDateTime": 1542392161.837,
      "lastUpdatedDateTime": 1542393013.377
   }
}
```

The dataset import is complete when the status shows as ACTIVE. Then you are ready to train the model using the specified dataset.

**Note**
Importing takes time. Wait until the dataset import is complete before training the model using the dataset.

### Step 2: Create a Solution (Train the Model)

Two steps are required to initially train a model. First, you create the configuration for training the model using the CreateSolution (p. 176) operation. Second, you train the model using the CreateSolutionVersion (p. 181) operation.

You train a model using a recipe and your training data. Amazon Personalize provides a set of predefined recipes. For more information, see Step 1: Choosing a Recipe (p. 64). For this exercise, you use AutoML, which allows Amazon Personalize to pick the best recipe based on the dataset you created in the preceding step.

1. Create the configuration for training a model by running the following command.

```bash
aws personalize create-solution \
--role-arn roleArn
```
The solution ARN is displayed, for example:

```json
{
    "solutionArn": "arn:aws:personalize:us-west-2:acct-id:solution/MovieSolution"
}
```

2. Check the create status using the describe-solution command. Provide the solution ARN that was returned in the previous step. For more information about the API, see DescribeSolution (p. 220).

```bash
aws personalize describe-solution \
```

The properties of the solution and the create status are displayed. Initially, the status shows as CREATE PENDING, for example:

```json
{
    "solution": {
        "name": "MovieSolution",
        "performAutoML": true,
        "performHPO": false,
        "solutionConfig": {
            "autoMLConfig": {
                ".metricName": "precision_at_25",
                "recipeList": [
                    "arn:aws:personalize::recipe/aws-hrnn"
                ]
            }
        },
        "creationDateTime": 1543864685.016,
        "lastUpdatedDateTime": 1543864685.016,
        "status": "CREATE PENDING"
    }
}
```

Note the metricName and recipeList. Because you specified performAutoML, Amazon Personalize chooses the recipe from the list that optimizes that metric. Because we didn't supply any metadata, the only recipe in the list is the HRNN (p. 75) recipe.

When the create status shows as ACTIVE, the solution configuration is complete and the model is ready for training.

3. Now that the configuration is ACTIVE, train the model by running the following command.

```bash
aws personalize create-solution-version \
```

The solution version ARN is displayed, for example:

```json
{
}
```
Check the training status of the solution version by using the describe-solution-version command. Provide the solution version ARN that was returned in the previous step. For more information about the API, see DescribeSolutionVersion (p. 223).

```
aws personalize describe-solution-version \
```

The properties of the solution version and the training status are displayed. Initially, the status shows as CREATE PENDING, for example:

```
{
  "solutionVersion": {
    ...
    "status": "CREATE PENDING"
  }
}
```

4. When the latest solution version training status shows as ACTIVE, the training is complete. The describe-solution-version response now includes recipeArn, which shows the recipe used to train the model as determined by Amazon Personalize, for example:

```
{
  "solutionVersion": {
    "performAutoML": true,
    "recipeArn": "arn:aws:personalize::recipe/aws-hrnn",
    "solutionConfig": {
      "autoMLConfig": {
        "metricName": "precision_at_25",
        "recipeList": [
          "arn:aws:personalize::recipe/aws-hrnn"
        ]
      }
    },
    ...
    "status": "ACTIVE"
  }
}
```

Now you can check the solution version metrics and create a campaign using the solution version.

**Note**

Training takes time. Wait until training is complete (the training status of the solution version shows as ACTIVE) before using this version of the solution in a campaign. For quicker training, instead of using perform-auto-ml, select a specific recipe using the recipe-arn parameter.

5. You can validate the performance of the solution version by reviewing its metrics. Get the metrics for the solution version by running the following command. Provide the solution version ARN that was returned previously. For more information about the API, see GetSolutionMetrics (p. 226).

```
aws personalize get-solution-metrics \
```
Step 3: Create a Campaign (Deploy the Solution)

Before you can get recommendations, you must deploy a version of the solution. Deploying a solution is also known as creating a campaign. Once you've created your campaign, your client application can get recommendations using the GetRecommendations (p. 269) API.

1. Create a campaign by running the following command. Provide the solution version ARN that was returned in the previous step. For more information about the API, see CreateCampaign (p. 157).

```bash
aws personalize create-campaign \
   --name MovieRecommendationCampaign \
   MovieSolution/version-id \
   --min-provisioned-tps 10
```

A sample response is shown:

```json
{
  "campaignArn": "arn:aws:personalize:us-west-2:acct-id:campaign/ 
   MovieRecommendationCampaign"
}
```

2. Check the deployment status by running the following command. Provide the campaign ARN that was returned in the previous step. For more information about the API, see DescribeCampaign (p. 202).

```bash
aws personalize describe-campaign \
   MovieRecommendationCampaign
```

A sample response is shown:

```json
{
  "campaign": {
    "name": "MovieRecommendationCampaign",
```
"campaignArn": "arn:aws:personalize:us-west-2:acct-id:campaign/MovieRecommendationCampaign",
"minProvisionedTPS": "10",
"creationDateTime": 1543864775.923,
"lastUpdatedDateTime": 1543864791.923,
"status": "CREATE_IN_PROGRESS"
}

Note
Wait until the status shows as ACTIVE before getting recommendations from the campaign.

Step 4: Get Recommendations

Get recommendations by running the get-recommendations command. Provide the campaign ARN that was returned in the previous step. In the request, you specify a user ID from the movie ratings dataset. For more information about the API, see GetRecommendations (p. 269).

Note
Not all recipes support the GetRecommendations API. For more information, see Step 1: Choosing a Recipe (p. 64).

The AWS CLI command you call in this step, personalize-runtime, is different than in previous steps.

```
aws personalize-runtime get-recommendations \
  --campaign-arn arn:aws:personalize:us-west-2:acct-id:campaign/MovieRecommendationCampaign \
  --user-id 123
```

In response, the campaign returns a list of item recommendations (movie IDs) the user might like. The list is sorted in descending order of relevance for the user.

```
{
  "itemList": [
    {
      "itemId": "14"
    },
    {
      "itemId": "15"
    },
    {
      "itemId": "275"
    },
    {
      "itemId": "283"
    },
    {
      "itemId": "273"
    },
    ...
  ]
}
```
Getting Started (AWS SDK for Python)

This topic explains how to get started programming Amazon Personalize with the AWS SDK for Python (Boto3).

Prerequisites

The following are prerequisite steps for using the Python examples in this guide:

- Complete the Getting Started Getting Started Prerequisites (p. 12). You can use the same source data that is listed in the Getting Started (Console) (p. 13) or Getting Started (AWS CLI) (p. 21) exercises. If you are using your own source data, make sure your data is formatted like in the prerequisite step Creating the Training Data (p. 12). For information about preparing your own source data, see Preparing and Importing Data (p. 40).
- Set up your AWS SDK for Python (Boto3) environment, as specified in Setting Up the AWS SDKs (p. 11).

When you finish the getting started exercise, to avoid incurring unnecessary charges, follow the steps in Cleaning Up Resources (p. 30) to delete the resources you created.

Step 1: Verify Your Python Environment

After you complete the prerequisites, run the following Python example to confirm that your environment is configured correctly. If your environment is configured correctly, a list of the available recipes is displayed and you can run the other Python examples in this guide.

```python
import boto3
personalize = boto3.client('personalize
response = personalize.list_recipes()
for recipe in response['recipes']:
    print (recipe)
```

Step 2: Import Training Data

After you verify that your Python environment is configured correctly, import your data. To use a dataset for training, you need to do the following:

1. Add a schema. The schema allows Amazon Personalize to parse the training dataset. For a code sample, see Creating a Schema Using the AWS Python SDK (p. 39).
2. Import the data. You create a dataset group which contains one or several datasets that Amazon Personalize can use for training. For a code sample, see Importing Bulk Records (AWS Python SDK) (p. 53).
3. (Optional) Add an event tracker. To record interactions events, you must add a tracking ID to associate the event with your dataset group. For a code sample, see Creating an Event Tracker (p. 59).
4. (Optional) Add an event record. To add more data in training and create a better model, you can use events. Events are recorded user activities such as a search, a view, or a purchase. For a code sample, see PutEvents Operation (p. 60).
Step 3: Create a Solution

After you import your data, create a solution and solution version. The solution contains the configurations to train a model. A solution version is a trained model. For more information, see Creating a Solution (p. 64).

When you create a solution version, evaluate its performance before proceeding. For a code sample, see Step 4: Evaluating the Solution Version (p. 101).

Step 4: Create a Campaign

After you train and evaluate your solution version, you can deploy it using a campaign. A campaign is an endpoint used to host a solution version and make recommendations to users. For a code sample, see Creating a Campaign (p. 104).

Step 5: Get Recommendations

After you create a campaign, you can use it to get recommendations. For a code sample, see Getting Real-Time Recommendations (API) (p. 106).

Getting Started Using Amazon Personalize APIs with Jupyter (iPython) Notebooks

To get started using Amazon Personalize using Jupyter notebooks, clone or download a series of notebooks found in the getting_started folder of the Amazon Personalize Samples repository. The notebooks walk you through importing training data, creating a solution, creating a campaign, and getting recommendations using Amazon Personalize.

Note
Before starting with the notebooks, make sure to build your environment following the steps in the README.md

Cleaning Up Resources

To avoid incurring unnecessary charges, delete the resources you created after you're done with the getting started exercise. To delete the resources, use either the Amazon Personalize console or the Delete APIs from the SDKs or the AWS Command Line Interface (AWS CLI). For example, use the DeleteCampaign (p. 184) API to delete a campaign.

You can't delete a resource whose status is CREATE PENDING or IN PROGRESS. The resource status must be ACTIVE or CREATE FAILED. Check the status using the Describe APIs, for example, DescribeCampaign (p. 202).

Some resources must be deleted before others, as shown in the following table. This process can take some time.

To delete the training data you uploaded, ratings.csv, see How Do I Delete Objects from an S3 Bucket?

<table>
<thead>
<tr>
<th>Resource to be Deleted</th>
<th>Delete This First</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Campaign (p. 288)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DatasetImportJob (p. 303)</td>
<td></td>
<td>Can not be deleted.</td>
</tr>
<tr>
<td>Resource to be Deleted</td>
<td>Delete This First</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EventTracker (p. 317)</td>
<td></td>
<td>The event-interactions dataset that is associated with the event tracker is not deleted and continues to be used by the solution version.</td>
</tr>
<tr>
<td>Dataset (p. 297)</td>
<td></td>
<td>No associated DatasetImportJob can have a status of CREATE PENDING or IN PROGRESS.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No associated SolutionVersion can have a status of CREATE PENDING or IN PROGRESS.</td>
</tr>
<tr>
<td>DatasetSchema (p. 307)</td>
<td>All datasets that reference the schema.</td>
<td></td>
</tr>
<tr>
<td>Solution (p. 337)</td>
<td>All campaigns based on the solution version.</td>
<td>No associated SolutionVersion can have a status of CREATE PENDING or IN PROGRESS.</td>
</tr>
<tr>
<td>SolutionVersion (p. 344)</td>
<td></td>
<td>Deleted when the associated Solution is deleted.</td>
</tr>
<tr>
<td>DatasetGroup (p. 299)</td>
<td>All associated event trackers. All associated solutions. All datasets in the dataset group.</td>
<td></td>
</tr>
</tbody>
</table>
Datasets and Schemas

Amazon Personalize datasets are containers for data. There are three types of datasets:

- **Users** – This dataset stores metadata about your users. This might include information such as age, gender, or loyalty membership, which can be important signals in personalization systems.
- **Items** – This dataset stores metadata about your items. This might include information such as price, SKU type, or availability.
- **Interactions** – This dataset stores historical and real-time data from interactions between users and items. This data can include impressions data and contextual metadata on your user's browsing context, such as their location or device (mobile, tablet, desktop, and so on). You must at minimum create an Interactions dataset.

The Users and Items dataset types are known as metadata types and are used only by certain recipes. For more information, see Step 1: Choosing a Recipe (p. 64).

Datasets are organized within Amazon Personalize dataset groups. A dataset group can only have one of each type of dataset. Each dataset must have an associated schema. A schema tells Amazon Personalize about the structure of your data and allows Amazon Personalize to parse the data. A schema has a name key whose value must match the dataset type.

You create a dataset and a schema when you import your training data into Amazon Personalize. For more information see Preparing and Importing Data (p. 40).

**Topics**
- Dataset and Schema Requirements (p. 32)
- Interactions Dataset (p. 34)
- Users Dataset (p. 36)
- Items Dataset (p. 37)
- Schema Examples (p. 39)
- Creating a Schema Using the AWS Python SDK (p. 39)

## Dataset and Schema Requirements

Each dataset has a set of required fields, reserved keywords, and their required data types, as shown in the following table.

<table>
<thead>
<tr>
<th>Dataset Type</th>
<th>Required Fields</th>
<th>Reserved Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users</td>
<td>USER_ID (string)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 metadata field</td>
<td></td>
</tr>
<tr>
<td>Items</td>
<td>ITEM_ID (string)</td>
<td>CREATION_TIMESTAMP (long)</td>
</tr>
<tr>
<td></td>
<td>1 metadata field</td>
<td></td>
</tr>
<tr>
<td>Interactions</td>
<td>USER_ID (string)</td>
<td>EVENT_TYPE (string)</td>
</tr>
</tbody>
</table>
Before you add a dataset to Amazon Personalize, you must define a schema for that dataset. Once you define the schema and create the dataset, you can't make changes to the schema. Schemas in Amazon Personalize are defined in the Avro format. For more information, see Apache Avro.

When you create a schema, you must follow these guidelines:

- The schema fields can appear in any order, but they must match the order of the corresponding column headers in the data file.
- Each dataset type requires specific non-metadata fields in its schema (see the preceding table). You must define required fields as their required data types.
- EVENT_VALUE data and Interactions, User, and Item metadata can be a null type. Adding a null type to a field in your schema allows you to use imperfect data (for example, metadata with blank values), to generate personalized recommendations.

### Metadata Fields

Metadata includes string or non-string fields that aren't required or don't use a reserved keyword. Metadata schemas have the following restrictions:

- Users and Items schemas require at least one metadata field,
- Each dataset has a limit on the number of metadata fields you can include. For a list of limits, see Service Quotas (p. 147).
- If you add your own metadata field of type string, it must include the categorical attribute. Otherwise, Amazon Personalize won't use the field when training a model.

### Reserved Keywords

Reserved keywords are optional, non-metadata fields. You must define reserved keywords as their required data type. The following are reserved keywords:

- EVENT_TYPE: Use an EVENT_TYPE field for Interactions datasets with one or more event types, such as Click and Download. You must define an EVENT_TYPE field as a string.
- EVENT_VALUE: Use an EVENT_VALUE field for Interactions datasets that include value data for events, such as PERCENT_WATCHED. You must define an EVENT_VALUE field only as a float or null.
- CREATION_TIMESTAMP: Use a CREATION_TIMESTAMP field for Items datasets with a timestamp for each item's creation date. Amazon Personalize uses CREATION_TIMESTAMP data to calculate the age of an item and adjust recommendations accordingly. See Creation Timestamp Data (p. 38).
- IMPRESSION: Use an IMPRESSION field for Interactions datasets with impressions data. Impressions are lists of items that were visible to a user when they interacted with (for example, clicked or watched) a particular item. For more information see Impressions Data (p. 35).
- RECOMMENDATION_ID: Use a RECOMMENDATION_ID field for Interactions datasets that use previous recommendations as implicit impressions data. For more information see Impressions Data (p. 35).
Interactions Dataset

An Interactions dataset stores historical and real-time data from interactions between users and items. To create a recommendation system using Amazon Personalize, you must at minimum create an Interactions dataset.

In Amazon Personalize, an interaction is an event that you record and then import as training data. You can record multiple event types, such as click, watch or like. For example, if a user clicks a particular item and then likes the item, and you want Amazon Personalize to use these events as training data, for each event you would record the user’s ID, the item’s ID, the timestamp (in Unix time epoch format), and the event type (click and like). You would then add both interaction events to an Interactions dataset. Once you have recorded enough events, you can train a model and use Amazon Personalize to generate recommendations for users. For minimum requirements see Service Quotas (p. 147).

When you create an Interactions dataset, you must also create a schema for the dataset. A schema tells Amazon Personalize about the structure of your data and allows Amazon Personalize to parse the data. For an example of a schema for an Interactions dataset see Interactions Schema Example (p. 35). For information on schema requirements see Dataset and Schema Requirements (p. 32).

This section provides information about the kinds of interactions data, including impressions data and contextual metadata, you can upload for training. It also includes an Interactions Schema Example (p. 35). For information about importing historical interactions data, see Preparing and Importing Data (p. 40). For information about recording events in real-time using the PutEvents (p. 259) API, see Recording Events (p. 58).

Once you create an Interactions dataset and import interaction data, you can then filter recommendations to include or exclude items that a user has interacted with. For more information see Filtering Recommendations (p. 113).

Topics
- Required Interaction Data (p. 34)
- Impressions Data (p. 35)
- Contextual Metadata (p. 35)
- Interactions Schema Example (p. 35)

Required Interaction Data

The training data you provide for each interaction must match your schema. Depending on your schema, interaction metadata can include empty/null values. At minimum, you must provide the following for each interaction:

- User ID
- Item ID
- Timestamp (in Unix epoch time format)

The maximum total number of optional metadata fields you can add to an Interactions dataset, combined with total number of distinct event types in your data, is 10. The metadata fields included in this count are EVENT_TYPE, EVENT_VALUE fields along with any custom metadata fields you add to your schema.

For more information on minimum requirements and maximum data limits for an Interactions dataset, see Service Quotas (p. 147).
Impressions Data

Impressions are lists of items that were visible to a user when they interacted with (for example, clicked or watched) a particular item. Amazon Personalize can model two types of impressions:

- **Implicit impressions** are the recommendations, retrieved from Amazon Personalize, that you show the user. You can integrate them into your recommendation workflow by including the `RecommendationId` (returned by the `GetRecommendations` (p. 269) and `GetPersonalizedRanking` (p. 265) operations) as input for future `PutEvents` (p. 259) requests and Amazon Personalize will derive the implicit impressions based on your recommendation data.

- **Explicit impressions** are impressions that you manually record and send to Amazon Personalize. Use explicit impressions to manipulate results from Amazon Personalize. For example, use explicit impressions to filter out unavailable items and change the order of recommendations based on user interactions.

Contextual Metadata

Interactions datasets can store contextual information for use in training. Contextual metadata is interactions data you collect on the user's environment at the time of an event. Including contextual metadata allows you to provide a more personalized experience for existing users. For example, if customers shop differently when accessing your catalog from a phone compared to a computer, include contextual metadata about the user's device. Recommendations will then be more relevant based on how they are browsing.

Additionally, contextual metadata helps decrease the cold-start phase for new or unidentified users. The cold-start phase refers to the period when your recommendation engine provides less relevant recommendations due to the lack of historical information regarding that user.

For more information on contextual information, see the following AWS Machine Learning Blog post: Increasing the relevance of your Amazon Personalize recommendations by leveraging contextual information.

Interactions Schema Example

The following example shows a schema for an Interactions dataset. The `USER_ID`, `ITEM_ID`, and `TIMESTAMP` fields are required. The `EVENT_TYPE`, `EVENT_VALUE`, `IMPRESSION`, `RECOMMENDATION_ID` fields are optional reserved keywords recognized by Amazon Personalize. `LOCATION` and `DEVICE` are optional contextual metadata fields. For information on schema requirements see Dataset and Schema Requirements (p. 32).

```
{
  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
    { "name": "USER_ID", "type": "string" },
    { "name": "ITEM_ID", "type": "string" },
    { "name": "EVENT_TYPE", "type": "string" },
    { "name": "EVENT_VALUE", "type": "string" },
    { "name": "IMPRESSION", "type": "string" },
    { "name": "RECOMMENDATION_ID", "type": "string" },
    { "name": "LOCATION", "type": "string" },
    { "name": "DEVICE", "type": "string" }
]
```
### Users Dataset

A Users dataset stores metadata about your users. This might include information such as age, gender, or loyalty membership.

When you create a Users dataset, you must also create a schema for the dataset. A schema tells Amazon Personalize about the structure of your data and allows Amazon Personalize to parse the data. For an example of a Users schema, see Users Schema Example (p. 37). For information on schema requirements see Dataset and Schema Requirements (p. 32).

This section provides information about required user data and the kinds of user data you can upload for training. It also includes a Users Schema Example (p. 37). For information about importing user data into a Users dataset, see Preparing and Importing Data (p. 40).

Once you create a Users dataset and add user data, you can then filter recommendations to include or exclude items based on specific user conditions. For more information see Filtering Recommendations (p. 113).

**Note**

RELATED_ITEMS recipes, such as item-to-item similarities (SIMS), do not use Users datasets.

**Topics**

- Required User Data (p. 37)
Required User Data

The training data you provide for each user must match your schema. At minimum, you must provide a User ID for each user. Depending on your schema, user metadata can include empty/null values.

For more information on minimum requirements and maximum data limits for a Users dataset, see Service Quotas (p. 147).

Users Schema Example

The following example shows a Users schema in Avro format. The USER_ID field is required and the AGE and GENDER fields are metadata. At least one metadata field is required. For information on schema requirements see Dataset and Schema Requirements (p. 32).

```json
{
    "type": "record",
    "name": "Users",
    "namespace": "com.amazonaws.personalize.schema",
    "fields": [
    {
        "name": "USER_ID",
        "type": "string"
    },
    {
        "name": "AGE",
        "type": "int"
    },
    {
        "name": "GENDER",
        "type": "string",
        "categorical": true
    }
    ],
    "version": "1.0"
}
```

Items Dataset

An Items dataset stores metadata about your items. This might include information such as price, genre, or availability.

When you create an Items dataset, you must also create a schema for the dataset. A schema tells Amazon Personalize about the structure of your data and allows Amazon Personalize to parse the data. For an example of a schema for an Items dataset see Items Schema Example (p. 38). For information on schema requirements see Dataset and Schema Requirements (p. 32).

This section provides information about required item data and the kinds of item data, including Creation Timestamp data, you can upload for training. It also includes an Items Schema Example (p. 38). For information about importing item data into an Items dataset, see Preparing and Importing Data (p. 40).

Once you create an Items dataset and import item data, you can then filter recommendations to include or exclude items based on specific item conditions. For more information see Filtering Recommendations (p. 113).
Required Item Data

The training data you provide for each item must match your schema. At minimum, you must provide an Item ID for each item. Depending on your schema, item metadata can include empty/null values.

During model training, Amazon Personalize considers a maximum of 750 thousand items. If you import more than 750 thousand items, Amazon Personalize decides which items to include in training, with an emphasis on including new items (items you recently created with no interactions) and existing items with recent interactions data.

For more information on minimum requirements and maximum data limits for an Items dataset, see Service Quotas (p. 147).

Creation Timestamp Data

Amazon Personalize uses creation timestamp data (in UNIX epoch time format, in seconds) to calculate the age of an item and adjust recommendations accordingly.

If creation timestamp data is missing for one or more items, Amazon Personalize infers this information from interaction data, if any, and uses the timestamp of the item's oldest interaction data as the item's creation date. If an item has no interaction data, its creation date is set as the timestamp of the latest interaction in the training set and is considered a new item.

Items Schema Example

The following example shows an Items schema. The ITEM_ID field is required. The GENRE field is metadata. At least one metadata field is required. The CREATION_TIMESTAMP field is a reserved keyword. For information on schema requirements see Dataset and Schema Requirements (p. 32).

```json
{
    "type": "record",
    "name": "Items",
    "namespace": "com.amazonaws.personalize.schema",
    "fields": [
        {
            "name": "ITEM_ID",
            "type": "string"
        },
        {
            "name": "GENRES",
            "type": [
                "null",
                "string"
            ],
            "categorical": true
        },
        {
            "name": "CREATION_TIMESTAMP",
            "type": "long"
        }
    ],
    "version": "1.0"
}
```
Schema Examples

For examples of schemas for each dataset type, see the following sections:

- Interactions Schema Example (p. 35)
- Users Schema Example (p. 37)
- Items Schema Example (p. 38)

Creating a Schema Using the AWS Python SDK

1. Define the Avro format schema that you want to use.
2. Save the schema in a JSON file in the default Python folder.
3. Create the schema using the following code.

   ```python
   import boto3
   personalize = boto3.client('personalize')
   with open('schema.json') as f:
       createSchemaResponse = personalize.create_schema(  
           name = 'YourSchema',  
           schema = f.read()  
       )
   schema_arn = createSchemaResponse['schemaArn']
   print( 'Schema ARN:' + schema_arn)
   ```

4. Amazon Personalize returns the ARN of the new schema. Store it for later use.

Amazon Personalize provides operations for managing schemas. For example, you can use the ListSchemas (p. 248) API to get a list of the available schemas.

After you create a schema, use it with datasets that match the schema. For more information, see Formatting Your Input Data (p. 46).
Preparing and Importing Data

Amazon Personalize uses data that you provide to train a model. When you import data, you can choose to import records in bulk or incrementally or both. With incremental imports, you can add individual historical records or data from live events, or both, depending on your business requirements.

This section provides information about importing historical data into Amazon Personalize. For information about recording live interactions data, see Recording Events (p. 58).

To import your historical training data into Amazon Personalize, you do the following:

1. Create an empty dataset group. **Dataset groups** are domain-specific containers for related datasets. For more information, see Step 1: Creating a Dataset Group (p. 40).
2. For each type of dataset you are using, create an empty dataset with an associated schema. **Datasets** are Amazon Personalize containers for data and schemas that specify contents of a dataset. For more information, see Step 2: Creating a Dataset and a Schema (p. 42).
3. Import your data:
   - Import bulk records stored in an Amazon S3 bucket using a dataset import job. See Importing Bulk Records (p. 46).
   - Import records incrementally using the AWS python SDK or AWS Command Line Interface (AWS CLI). See Importing Records Incrementally (p. 54).

Step 1: Creating a Dataset Group

**Dataset groups** are domain-specific containers for datasets. For example, you might have an application that provides recommendations for streaming video and another that provides recommendations for audio books. In Amazon Personalize, each application would have its own dataset group. You can create a dataset group using the Amazon Personalize console, AWS SDK, or AWS Command Line Interface (AWS CLI).

**Topics**
- Creating a Dataset Group (Console) (p. 40)
- Creating a Dataset Group (AWS CLI) (p. 41)
- Creating a Dataset Group (AWS Python SDK) (p. 41)

Creating a Dataset Group (Console)

Create a dataset group by specifying the dataset group name in the Amazon Personalize console.

**To create a dataset group**

1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/ and sign in to your account.
2. Choose Create dataset group.
3. If this is your first time using Amazon Personalize, on the Create dataset group page, in New dataset group, choose Get started.
4. In Dataset group details, for Dataset group name, specify a name for your dataset group.
5. Choose Next. The Create user-item interaction data page displays. You are now ready to add a dataset with an associated schema to your dataset group. See Creating a Dataset and a Schema (Console) (p. 42).

Creating a Dataset Group (AWS CLI)

Create a dataset group by running the following command. For more information about the API, see CreateDatasetGroup (p. 163).

```bash
aws personalize create-dataset-group --name dataset group name
```

The dataset group Amazon Resource Name (ARN) is displayed as shown in the following example.

```
{
"datasetGroupArn": "arn:aws:personalize:us-west-2:acct-id:dataset-group/DatasetGroupName"
}
```

Record this value for future use. To display the dataset group that you created, use the describe-dataset-group command and specify the returned dataset group ARN.

```bash
aws personalize describe-dataset-group \
--dataset-group-arn dataset group arn
```

The dataset group and its properties are displayed, as shown in the following example.

```
{
  "datasetGroup": {  
    "name": "DatasetGroupName",
    "status": "ACTIVE",
    "creationDateTime": 1542392161.262,
    "lastUpdatedDateTime": 1542396513.377
  }
}
```

When the dataset group's status is ACTIVE, proceed to Creating a Dataset and a Schema (AWS CLI) (p. 43).

Creating a Dataset Group (AWS Python SDK)

Create a dataset group using the CreateDatasetGroup (p. 163) operation.

```python
import boto3
personalize = boto3.client('personalize')
response = personalize.create_dataset_group(name = 'YourDatasetGroup')
dsg_arn = response['datasetGroupArn']
description = personalize.describe_dataset_group(datasetGroupArn = dsg_arn)['datasetGroup']
print('Name: ' + description['name'])
print('ARN: ' + description['datasetGroupArn'])
```
The `print('Status: ' + description['status'])` operation returns the datasetGroupArn and the status of the operation.

When the status is ACTIVE, proceed to Creating a Dataset and a Schema (AWS Python SDK) (p. 44).

**Step 2: Creating a Dataset and a Schema**

After you have completed Step 1: Creating a Dataset Group (p. 40), you are ready to create a dataset. **Datasets** are Amazon Personalize containers for data. Datasets are organized within Amazon Personalize dataset groups.

You can create three types of historical datasets: Users, Items, and Interactions. When you create a dataset, you also create a schema for the dataset. A **schema** tells Amazon Personalize about the structure of your data and allows Amazon Personalize to parse the data.

You can create only one of each kind of dataset in a dataset group, and you must at minimum create an Interactions dataset. You create datasets using the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDK.

For more information, including dataset and schema requirements, see Datasets and Schemas (p. 32).

**Topics**
- Creating a Dataset and a Schema (Console) (p. 42)
- Creating a Dataset and a Schema (AWS CLI) (p. 43)
- Creating a Dataset and a Schema (AWS Python SDK) (p. 44)

### Creating a Dataset and a Schema (Console)

If this is your first dataset in your dataset group, your first dataset type will be an Interactions dataset. To create your Interactions dataset in the console, specify the dataset name and then specify a JSON schema in Avro format. If it is not your first dataset in this dataset group, choose the dataset type and then specify a name and a schema.

For information on Amazon Personalize datasets and schema requirements, see Datasets and Dataset Groups (p. 4).

**Note**

If you just completed Step 1: Creating a Dataset Group (p. 40) and you are already on the user-item interaction page, skip to step 4 in this procedure.

**To create a dataset and a schema**

1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/ and sign in to your account.
2. On the **Dataset groups** page, choose the dataset group you created in Step 1: Creating a Dataset Group (p. 40). This displays the dataset group **Dashboard**.
3. In the **Upload datasets** section, for the type of dataset that you want to import (Amazon Personalize datasets include Interactions, Users, or Items), choose **Import**. The **Configure < dataset type >** page is displayed.
4. In **Dataset details**, for **Dataset name**, specify a name for your dataset.
5. In **Schema details**, for **Schema selection**, either choose an existing schema or choose **Create new schema**.
6. If you are creating a new schema, for **Schema definition**, paste in the schema JSON that matches your data. Use the examples found in **Datasets and Schemas (p. 32)** as a guide.

7. For **New schema name**, specify a name for the new schema.

8. Choose **Next** and follow the instructions in **Step 3: Importing Your Data (p. 45)** to import your data.

---

**Creating a Dataset and a Schema (AWS CLI)**

To create a dataset and a schema using the AWS CLI, you first define a schema in **Avro format** and add it to Amazon Personalize using the **CreateSchema (p. 174)** operation. Then create a dataset using the **CreateDataset (p. 160)** operation. For information on Amazon Personalize datasets and schema requirements, see **Datasets and Dataset Groups (p. 4)**.

**To create a schema and dataset**

1. Create a schema file in Avro format and save it as a JSON file. This file should be based on the type of dataset, such as Interactions, you are creating.

   The schema must match the columns in your data and the schema name must match one of the three types of datasets recognized by Amazon Personalize. The following is an example of a minimal Interactions dataset schema. For more examples, see **Datasets and Dataset Groups (p. 4)**.

   ```json
   {
   "type": "record",
   "name": "Interactions",
   "namespace": "com.amazonaws.personalize.schema",
   "fields": [
   {
   "name": "USER_ID",
   "type": "string"
   },
   {
   "name": "ITEM_ID",
   "type": "string"
   },
   {
   "name": "TIMESTAMP",
   "type": "long"
   }
   ],
   "version": "1.0"
   }
   ```

2. Create a schema in Amazon Personalize by running the following command. Replace `schemaName` with the name of the schema, and replace `file://SchemaName.json` with the location of the JSON file you created in the previous step. The example shows the file as belonging to the current folder. For more information about the API, see **CreateSchema (p. 174)**.

   ```bash
   aws personalize create-schema \
   --name SchemaName \
   --schema file://SchemaName.json
   ```

   The schema Amazon Resource Name (ARN) is displayed, as shown in the following example:

   ```json
   {
   "schemaArn": "arn:aws:personalize:us-west-2:acct-id:schema/SchemaName"
   }
   ```
3. Create an empty dataset by running the following command. Provide the dataset group Amazon Resource Name (ARN) from Creating a Dataset Group (AWS CLI) (p. 41) and schema ARN from the previous step. The dataset-type must match the schema name from the previous step. For more information about the API, see CreateDataset (p. 160).

```bash
aws personalize create-dataset
--name Dataset Name
--dataset-group-arn Dataset Group ARN
--dataset-type Dataset Type
--schema-arn Schema Arn
```

The dataset ARN is displayed, as shown in the following example.

```
{
  "datasetArn": "arn:aws:personalize:us-west-2:acnt-id:dataset/DatasetName/INTERACTIONS"
}
```

4. Record the dataset ARN for later use. After you have created a dataset, you are ready to import your training data. See Step 3: Importing Your Data (p. 45).

Creating a Dataset and a Schema (AWS Python SDK)

To create a dataset and a schema using the AWS Python SDK, you first define a schema in Avro format and add it to Amazon Personalize using the CreateSchema (p. 174) operation. Then create a dataset using the CreateDataset (p. 160) operation. For information on Amazon Personalize datasets and schema requirements, see Datasets and Dataset Groups (p. 4).

To create a schema and a dataset

1. Create a schema file in Avro format and save it as a JSON file in your working directory.

The schema must match the columns in your data and the schema name must match one of the three types of datasets recognized by Amazon Personalize. The following is an example of a minimal Interactions dataset schema. For more examples, see Datasets and Dataset Groups (p. 4).

```json
{
  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
    {
      "name": "USER_ID",
      "type": "string"
    },
    {
      "name": "ITEM_ID",
      "type": "string"
    },
    {
      "name": "TIMESTAMP",
      "type": "long"
    }
  ],
  "version": "1.0"
}
```

2. Create the schema using the CreateSchema (p. 174) API. Replace Schema Name with the name of your schema.
import boto3
personalize = boto3.client('personalize')

with open('schemaFile.json') as f:
    createSchemaResponse = personalize.create_schema(
        name = 'Schema Name',
        schema = f.read()
    )
schema_arn = createSchemaResponse['schemaArn']
print('Schema ARN: ' + schema_arn)

Amazon Personalize returns the ARN of the new schema. Record it because you'll need it in the next step.

3. Create a dataset using the CreateDataset (p. 160) operation. Specify the datasetGroupArn returned in Creating a Dataset Group (AWS Python SDK) (p. 41). Use the schemaArn created in the previous step. Replace dataset type with the type of dataset you are uploading. For types of datasets, see Datasets and Schemas (p. 32).

import boto3
personalize = boto3.client('personalize')

response = personalize.create_dataset(
    name = 'dataset_name',
    schemaArn = 'schema_arn',
    datasetGroupArn = 'dataset_group_arn',
    datasetType = 'dataset_type'
)
print ('Dataset Arn: ' + response['datasetArn'])

After you have created a dataset, you are ready to import your training data. See Step 3: Importing Your Data (p. 45).

Step 3: Importing Your Data

When you have completed Step 1: Creating a Dataset Group (p. 40) and Step 2: Creating a Dataset and a Schema (p. 42), you are ready to import your training data into Amazon Personalize. When you import data, you can choose to import records in bulk, import records individually, or both, depending on your business requirements and the amount of historical data you have collected. If you have a large amount of historical records, we recommend you first import data in bulk and then add data incrementally as necessary.

Important
Bulk imports in Amazon Personalize are a full refresh of bulk data. Existing bulk data in the dataset is replaced. This does not include records imported incrementally.

To import real-time event data, record user activity (events) in real-time using the AWS SDK, AWS Amplify, or AWS CLI. For more information see Recording Events (p. 58)

Topics
- Importing Bulk Records (p. 46)
- Importing Records Incrementally (p. 54)
Importing Bulk Records

**Important**

Bulk imports in Amazon Personalize are a full refresh of bulk data. Existing bulk data in the dataset is replaced. This does not include records imported incrementally.

After you have completed Step 1: Creating a Dataset Group (p. 40) and Step 2: Creating a Dataset and a Schema (p. 42), you can import bulk records, such as a large CSV file, into an Amazon Personalize dataset.

Amazon Personalize updates any filters you created in the dataset group with your new item and user data within 15 minutes from the last bulk import. This update allows your campaigns to use your most recent data when filtering recommendations for your users.

To import bulk records, you do the following:

1. Format your input data as a comma-separated values (CSV).
2. Upload your CSV files to an Amazon Simple Storage Service (Amazon S3) bucket and give Amazon Personalize access to your Amazon S3 resources.
3. Create a dataset import job that populates the dataset with data from your S3 bucket.

**Topics**

- Formatting Your Input Data (p. 46)
- Uploading to an Amazon S3 Bucket (p. 48)
- Importing Bulk Records with a Dataset Import Job (p. 51)

### Formatting Your Input Data

The files that you use to import data into Amazon Personalize must map to the schema that you are using.

Amazon Personalize imports data only from files that are in the comma-separated values (CSV) format. Amazon Personalize requires the first row of your CSV file to contain column headers. The column headers in your CSV file need to map to the schema to create the dataset. Don't enclose headers in quotation marks (").

TIMESTAMP and CREATION_TIMESTAMP data must be in UNIX epoch time format. For more information see Timestamp Data (p. 47).

**Important**

If your data includes any non-ASCII encoded characters, your CSV file must be encoded in UTF-8 format.

The following interactions data represents historical user activity from a website that sells movie tickets. You can use the data to train a model that gives a user movie recommendations based on the activities of other users.

```
USER_ID,ITEM_ID,EVENT_TYPE,EVENT_VALUE,TIMESTAMP
196,242,click,15,881250949
186,302,click,13,891717742
22,377,click,10,878887116
244,51,click,20,880606923
166,346,click,10,886397596
298,474,click,40,884182806
115,265,click,20,881171488
253,465,click,50,891628467
305,451,click,30,886324817
```
The associated Interactions schema is repeated below.

```json
{
  "type": "record",
  "name": "Interactions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
    { "name": "USER_ID", "type": "string" },
    { "name": "ITEM_ID", "type": "string" },
    { "name": "EVENT_TYPE", "type": "string" },
    { "name": "EVENT_VALUE", "type": "float" },
    { "name": "TIMESTAMP", "type": "long" }
  ],
  "version": "1.0"
}
```

Amazon Personalize requires the USER_ID, ITEM_ID, and TIMESTAMP fields. USER_ID is the identifier for a user of your application. ITEM_ID is the identifier for a movie. EVENT_TYPE and EVENT_VALUE are the identifiers for user activities. In the sample data, a click might represent a movie purchase event and 15 might be the purchase price of the movie. TIMESTAMP represents the Unix epoch time that the movie purchase took place.

**Timestamp Data**

Timestamp data, such as TIMESTAMP (for Interactions datasets) or CREATION_TIMESTAMP (for Items datasets) data, must be in Unix epoch time format in seconds. For example, the Epoch timestamp in seconds for date July 31, 2020 is 1596238243. To convert dates to Unix epoch timestamps use an Epoch Converter - Unix Timestamp Converter.

**Formatting Impressions Data**

Impressions are lists of items that were visible to a user when they interacted with (for example, clicked or watched) a particular item.

Amazon Personalize can model two types of impressions:

- **Implicit impressions** are the recommendations, retrieved from Amazon Personalize, that you show the user. You can integrate them into your recommendation workflow by including the RecommendationId (returned by the GetRecommendations (p. 269) and GetPersonalizedRanking (p. 265) operations) as input for future PutEvents (p. 259) requests and Amazon Personalize will derive the implicit impressions based on your recommendation data.

- **Explicit impressions** are impressions that you manually record and send to Amazon Personalize. Use explicit impressions to manipulate results from Amazon Personalize. For example, use explicit
impressions to filter out unavailable items and change the order of recommendations based on user interactions.

**Formatting Explicit Impressions**

Bulk impressions data is recorded in Interactions datasets, with each impression separated by a `|` character.

The following is a short excerpt from an Interactions dataset that includes explicit impressions in the **IMPRESSION** column.

<table>
<thead>
<tr>
<th>EVENT_TYPE</th>
<th>IMPRESSION</th>
<th>ITEM_ID</th>
<th>TIMESTAMP</th>
<th>USER_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>click</td>
<td>73</td>
<td>70</td>
<td>17</td>
<td>95</td>
</tr>
<tr>
<td>click</td>
<td>35</td>
<td>82</td>
<td>78</td>
<td>57</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

The application showed user **USER_1** items 73, 70, 17, 95, and 96 and the user ultimately chose item 73. When you create a new solution version based on this data, items 70, 17, 95, and 96 will be less frequently recommended to user **USER_1**.

**Categorical Data**

To include multiple categories for a single item when you use categorical string data, separate the values using the vertical bar, `|`, character. For example, to match the Items schema from the previous section using two categories, a data row would resemble the following:

```
ITEM_ID,GENRE
item_123,horror|comedy
```

After you format your data, upload it to an Amazon S3 bucket so you can import it into Amazon Personalize. For more information, see Uploading to an Amazon S3 Bucket (p. 48).

**Uploading to an Amazon S3 Bucket**

After you format your historical input data (see Formatting Your Input Data (p. 46)), you must upload the CSV file to an Amazon S3 bucket and give Amazon Personalize permission to access to your Amazon S3 resources:

1. If you haven't already, follow the steps in Setting Up Permissions (p. 8) to set up permissions so your IAM users can access Amazon Personalize and Amazon Personalize can access your resources.

2. Upload your CSV files to an Amazon Simple Storage Service (Amazon S3) bucket. This is the location that Amazon Personalize imports your data from. For more information, see Uploading Files and Folders by Using Drag and Drop in the Amazon Simple Storage Service Console User Guide.

3. Give Amazon Personalize access to your Amazon S3 resources by attaching access policies to your Amazon S3 bucket and Amazon Personalize service role. See Giving Amazon Personalize Access to Amazon S3 Resources (p. 49).

**Note**

Amazon S3 buckets and objects must be either encryption free or, if you are using AWS Key Management Service (AWS KMS) for encryption, you must give your IAM user and Amazon
Personalize IAM service role permission to use your key. For more information see Using key policies in AWS KMS in the AWS Key Management Service Developer Guide.

After you upload your data to an Amazon S3 bucket and give Amazon Personalize access to Amazon S3, import your data into Amazon Personalize. See Step 3: Importing Your Data (p. 45).

Giving Amazon Personalize Access to Amazon S3 Resources

To give Amazon Personalize access to your Amazon S3 bucket, do the following:

1. If you haven’t already, follow the steps in Setting Up Permissions (p. 8) to set up permissions so your IAM users can access Amazon Personalize and Amazon Personalize can access your resources.
2. Attach a policy to the Amazon Personalize service role (see Creating an IAM Role for Amazon Personalize (p. 9)) that allows access to your Amazon S3 bucket. For more information, see Attaching an Amazon S3 Policy to the Amazon Personalize Service Role (p. 49).
3. Attach a bucket policy to the Amazon S3 bucket containing your data files so Amazon Personalize can access them. For more information, see Attaching an Amazon Personalize Access Policy to Your S3 Bucket (p. 50).
4. If you are using AWS KMS for encryption, give your Amazon Personalize service role permission to use your key. For more information see Using key policies in AWS KMS in the AWS Key Management Service Developer Guide.

Note
Because Amazon Personalize doesn’t communicate with AWS VPCs, Amazon Personalize can’t interact with Amazon S3 buckets that allow only VPC access.

Attaching an Amazon S3 Policy to the Amazon Personalize Service Role

To attach an Amazon S3 policy to your Amazon Personalize role do the following:

2. In the navigation pane, choose Policies, and choose Create policy.
3. Choose the JSON tab, and update the policy as follows. Replace bucket-name with the name of your bucket. If you are using a batch workflow, Amazon Personalize needs additional permissions. See Amazon S3 Policy for Batch Workflows (p. 50).

```
{
   "Version": "2012-10-17",
   "Id": "PersonalizeS3BucketAccessPolicy",
   "Statement": [
      {
         "Sid": "PersonalizeS3BucketAccessPolicy",
         "Effect": "Allow",
         "Action": [
            "s3:GetObject",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::bucket-name",
            "arn:aws:s3:::bucket-name/*"
         ]
      }
   ]
}
```

5. For Name, enter PersonalizeS3BucketAccessPolicy.
6. (Optional) For **Description**, enter a short sentence describing this policy, for example, **Allow Amazon Personalize to access its S3 bucket.**

7. Choose **Create policy**.

8. In the navigation pane, choose **Roles**, and choose the role you created for Amazon Personalize. See **Creating an IAM Role for Amazon Personalize (p. 9)**.

9. For **Permissions**, choose **Attach policies**.

10. To display the policy in the list, type part of the policy name in the **Filter policies** filter box.

11. Choose the check box next to the policy you created earlier in this procedure.

12. Choose **Attach policy**.

Before your role is ready for use with Amazon Personalize you must also attach a bucket policy to the Amazon S3 bucket containing your data. See **Attaching an Amazon Personalize Access Policy to Your S3 Bucket (p. 50)**.

**Amazon S3 Policy for Batch Workflows**

For batch workflows, Amazon Personalize needs permission to access and add files to your Amazon S3 bucket. Follow the steps above to attach the following policy to your Amazon Personalize role. Replace **bucket-name** with the name of your bucket. For more information on batch workflows, see **Getting Batch Recommendations (p. 108)**.

```
{
   "Version": "2012-10-17",
   "Id": "PersonalizeS3BucketAccessPolicy",
   "Statement": [
      {
         "Sid": "PersonalizeS3BucketAccessPolicy",
         "Effect": "Allow",
         "Action": [
            "s3:GetObject",
            "s3:ListBucket",
            "s3:PutObject"
         ],
         "Resource": [
            "arn:aws:s3:::bucket-name",
            "arn:aws:s3:::bucket-name/*"
         ]
      }
   ]
}
```

**Attaching an Amazon Personalize Access Policy to Your S3 Bucket**

Amazon Personalize needs permission to access the S3 bucket. For non-batch workflows, attach the following policy to your bucket. Replace **bucket-name** with the name of your bucket. For batch workflows, see **S3 Bucket Policy for Batch Workflows (p. 51)**.

For more information on Amazon S3 bucket policies, see **How Do I Add an S3 Bucket Policy?**.

```
{
   "Version": "2012-10-17",
   "Id": "PersonalizeS3BucketAccessPolicy",
   "Statement": [
      {
         "Sid": "PersonalizeS3BucketAccessPolicy",
         "Effect": "Allow",
         "Principal": {
```

50
"Service": "personalize.amazonaws.com"
},
"Action": [
  "s3:GetObject",
  "s3:ListBucket"
],
"Resource": [
  "arn:aws:s3:::bucket-name",
  "arn:aws:s3:::bucket-name/*"
]
}
}

S3 Bucket Policy for Batch Workflows

For batch workflows, Amazon Personalize needs permission to access and add files to your Amazon S3 bucket. Attach the following policy to your bucket. Replace bucket-name with the name of your bucket.

For more information on adding an Amazon S3 bucket policy to a bucket, see How Do I Add an S3 Bucket Policy?. For more information on batch workflows, see Getting Batch Recommendations (p. 108).

{
  "Version": "2012-10-17",
  "Id": "PersonalizeS3BucketAccessPolicy",
  "Statement": [
    {
      "Sid": "PersonalizeS3BucketAccessPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "personalize.amazonaws.com"
      },
      "Action": [
        "s3:GetObject",
        "s3:ListBucket",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name",
        "arn:aws:s3:::bucket-name/*"
      ]
    }
  ]
}

Importing Bulk Records with a Dataset Import Job

After you have formatted your input data (see Formatting Your Input Data (p. 46)) and uploaded it to an Amazon Simple Storage Service (Amazon S3) bucket (see Uploading to an Amazon S3 Bucket (p. 48)), import the bulk records by creating a dataset import job.

A dataset import job is a bulk import tool that populates your dataset with data from your S3 bucket. You create a dataset import job and import bulk records using the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDK.

Topics
- Importing Bulk Records (Console) (p. 52)
- Importing Bulk Records (AWS CLI) (p. 52)
- Importing Bulk Records (AWS Python SDK) (p. 53)
Importing Bulk Records (Console)

To import bulk records into a dataset in Amazon Personalize using the console, create a dataset import job with a name, the IAM service role, and the location of your data.

To import bulk records (console)

Note
If you just created your dataset in Step 2: Creating a Dataset and a Schema (p. 42), skip to step 5.

1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/ and sign in to your account.
2. On the Dataset groups page, choose your dataset group. The dataset group Dashboard is displayed.
3. In the Upload datasets section, for the type of dataset you want to import, choose Import. The Configure < dataset type > page is displayed.
4. If you already created a dataset of this type, all Dataset details and Schema details fields are disabled. Choose Next.

If haven't created a dataset of this type, complete the Dataset details and Schema details fields to create a dataset.

5. In Dataset import job details, for Dataset import job name, specify a name for your import job.
6. For IAM service role, keep the default selection of Enter a custom IAM role ARN.
7. For Custom IAM role ARN, specify the role that you created in Creating an IAM Role for Amazon Personalize (p. 9).
8. For Data location, specify where your data file is stored in Amazon S3. Use the following syntax:

   s3://<name of your S3 bucket>/<folder path>/<CSV filename>

   Note
   If your CSV files are in a folder in your S3 bucket and you want to upload multiple CSV files to a dataset with one dataset import job, use this syntax without the CSV file name.

9. Choose Finish. The data import job starts and the Dashboard Overview page is displayed.

The dataset import is complete when the status shows as ACTIVE. You can now train the model using the specified dataset.

After you import your data into the relevant datasets in the dataset group, create a solution version by training a model. For more information, see Creating a Solution (p. 64).

Importing Bulk Records (AWS CLI)

To import bulk records using the AWS CLI, create a dataset import job using the CreateDatasetImportJob (p. 166) command.

To import bulk records (AWS CLI)

1. Create a dataset import job by running the following command. Provide the dataset Amazon Resource Name (ARN) from Step 2: Creating a Dataset and a Schema (p. 42) and your S3 bucket name. Supply the AWS Identity and Access Management (IAM) role Amazon Resource Name (ARN) that you created in Creating an IAM Role for Amazon Personalize (p. 9). For more information about the operation, see CreateDatasetImportJob (p. 166).

   ```bash
   aws personalize create-dataset-import-job \
   --job-name dataset import job name \
   --dataset-arn dataset arn \
   ```
The dataset import job ARN is displayed, as shown in the following example.

```
{
}
```

2. Check the status by using the `describe-dataset-import-job` command. Provide the dataset import job ARN that was returned in the previous step. For more information about the operation, see `DescribeDatasetImportJob` (p. 208).

```
aws personalize describe-dataset-import-job
  --dataset-import-job-arn dataset import job arn
```

The properties of the dataset import job, including its status, are displayed. Initially, the status shows as `CREATE PENDING`.

```
{
  "datasetImportJob": {
    "jobName": "Dataset Import job name",
    "dataSource": {
      "dataLocation": "s3://<bucketname>/ratings.csv"
    },
    "roleArn": "role-arn",
    "status": "CREATE PENDING",
    "creationDateTime": 1542392161.837,
    "lastUpdatedDateTime": 1542393013.377
  }
}
```

The dataset import is complete when the status shows as `ACTIVE`. You can now train the model using the specified dataset.

After you import your data into the relevant datasets in the dataset group, create a solution version by training a model. For more information, see `Creating a Solution` (p. 64).

**Importing Bulk Records (AWS Python SDK)**

To add data to your dataset, create and run a dataset import job using the `CreateDatasetImportJob` (p. 166) operation. Specify the `datasetGroupArn` and set the `dataLocation` to the `bucket-name/file.csv` where you stored the training data.

If your CSV files are in a folder in an S3 bucket, you can upload multiple CSV files to a dataset in one dataset import job. For `dataLocation`, specify the `bucket-name/folder-name/` instead of the file name.

For the `roleArn`, see `Creating an IAM Role for Amazon Personalize` (p. 9). The `roleArn` parameter specifies the AWS Identity and Access Management (IAM) role that gives Amazon Personalize permissions to access your S3 bucket.

```
import boto3
```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_dataset_import_job(
    jobName = 'YourImportJob',
    datasetArn = 'dataset_arn',
    dataSource = {'dataLocation': 's3://bucket/file.csv'},
    roleArn = 'role_arn'
)

dsij_arn = response['datasetImportJobArn']

print('Dataset Import Job arn: ' + dsij_arn)

description = personalize.describe_dataset_import_job(
    datasetImportJobArn = dsij_arn)['datasetImportJob']

print('Name: ' + description['jobName'])
print('ARN: ' + description['datasetImportJobArn'])
print('Status: ' + description['status'])

The response returns the datasetImportJobArn. The response from the DescribeDatasetImportJob (p. 208) operation includes the status of the operation.

You must wait until the status changes to ACTIVE before you can use the data to train a model.

Amazon Personalize provides operations for managing datasets, dataset groups, and dataset import jobs. For example, you can use ListDatasets (p. 239) to list the datasets in a dataset group and DeleteDataset (p. 186) to delete a dataset.

After you import your data into the relevant datasets in the dataset group, create a solution version by training a model. For more information, see Creating a Solution (p. 64).

## Importing Records Incrementally

Once you have completed Step 1: Creating a Dataset Group (p. 40) and Step 2: Creating a Dataset and a Schema (p. 42), you can incrementally add one or more new items or users to an existing dataset.

### Filter Updates for Incremental Record Imports

Amazon Personalize updates any filters you created in the dataset group with your new item and user data within 20 minutes from the last incremental import. This update allows your campaigns to use your most recent data when filtering recommendations for your users.

### How New Records Influence Recommendations

If you have already created a solution version (trained a model), new items and users influence recommendations as follows:

- For **new items**, if you trained the model using the User-Personalization recipe, Amazon Personalize automatically updates the model every two hours, and after each update the new items influence recommendations. See User-Personalization Recipe (p. 66).

  For any other recipe, you must re-train the model for the new items to influence recommendations.

- For **new users**, recommendations will initially be for popular items only. As you record events for the user, recommendations will be more relevant. For more information, see Recording Events (p. 58).

### Topics

- Importing Interactions Incrementally (p. 55)
Importing Interactions Incrementally

You can upload new individual interactions records to your Interactions dataset in real-time. To import real-time event data, record user activity (events) in real-time using the AWS SDK, AWS Amplify or AWS CLI. For more information see Recording Events (p. 58).

Importing Users Incrementally

You can incrementally add one or more new users to your Users dataset using the AWS Command Line Interface (AWS CLI) or AWS SDK. If a user with the same userId is already in your Users dataset, Amazon Personalize replaces the existing user with the new one.

Topics

- Importing Users Incrementally (AWS CLI) (p. 55)
- Importing Users Incrementally (AWS Python SDK) (p. 55)

Importing Users Incrementally (AWS CLI)

Add one or more users to your Users dataset using the PutUsers (p. 263) API. You can add up to 10 users with a single PutUsers call.

Replace the dataset arn with the Amazon Resource Name (ARN) of your dataset and replace userId with the ID of the user. If a user with the same userId is already in your Users dataset, Amazon Personalize replaces the existing user with the new one.

For properties, for each metadata field in your Users dataset, replace the propertyName with the field name from your schema in camel case. For example, NUMBER_OF_VIDEOS_WATCHED would be numberOfVideosWatched. Replace userData with the data for the user.

```bash
aws personalize-events put-users \
  --dataset-arn dataset arn \
  --users '[{
    "userId": "user Id",
    "properties": "\"property data\"
  }]' 
```

Importing Users Incrementally (AWS Python SDK)

Add one or more users to your Users dataset using the PutUsers (p. 263) operation. You can add up to 10 users with a single PutUsers call.

Replace the dataset arn with the ARN of your dataset and replace userId with the ID of the user. If a user with the same userId is already in your Users dataset, Amazon Personalize replaces the existing user with the new one.

For properties, for each metadata field in your Users dataset, replace the propertyName with the field name from your schema in camel case. For example, NUMBER_OF_VIDEOS_WATCHED would be numberOfVideosWatched. Replace userData with the data for the user.

```python
import boto3

personalize_events = boto3.client(service_name='personalize-events')
```
Importing Items Incrementally

You can incrementally add one or more new items, such as a movie or book, to your Items dataset using the AWS Command Line Interface (AWS CLI) or AWS SDK. If an item with the same itemId is already in your Items dataset, Amazon Personalize replaces the existing item with the new one.

Topics

- Importing Items Incrementally (AWS CLI) (p. 56)
- Importing Items Incrementally (AWS Python SDK) (p. 56)

Importing Items Incrementally (AWS CLI)

Add one or more items to your Items dataset using the PutItems (p. 261) API. You can add up to 10 items with a single PutItems call.

Replace the dataset arn with the Amazon Resource Name (ARN) of your dataset and replace itemId with the ID of the item. If an item with the same itemId is already in your Items dataset, Amazon Personalize replaces the existing item with the new one.

For properties, for each field in your Items dataset, replace the propertyName with the field name from your schema in camel case. For example, CREATION_TIMESTAMP would be creationTimestamp. Replace item data with the data for the item.

```bash
aws personalize-events put-items \
   --dataset-arn dataset arn \
   --items '[
   "itemId": "item Id",
   "properties": "{""propertyName"": ""user data""}"
   ]'
```

Importing Items Incrementally (AWS Python SDK)

Add one or more items to your Items dataset using the PutItems (p. 261) operation. You can add up to 10 items with a single PutItems call.

Replace the dataset arn with the ARN of your dataset and replace itemId with the ID of the item. If an item with the same itemId is already in your Items dataset, Amazon Personalize replaces the existing item with the new one.

For properties, for each metadata field in your Items dataset, replace the propertyName with the field name from your schema in camel case. For example, CREATION_TIMESTAMP would be creationTimestamp. Replace item data with the data for the item.

```python
import boto3

personalize_events = boto3.client(service_name='personalize-events')

personalize_events.put_items(
    datasetArn = 'dataset arn',
    users = [{
        'userId': 'user ID',
        'properties': "{""propertyName"": "user data""}"
    }]
)"
```python
datasetArn = 'dataset arn',
items = [{
    'itemId': 'item ID',
    'properties': '{"propertyName": "item data"}
}
}
```
Recording Events

Amazon Personalize can make recommendations based on real-time event data only, historical imported data (as demonstrated in the Getting Started (p. 12) guides) only, or a mixture of both. To import real-time event data, record user activity (events) in real-time using the AWS python SDK, AWS Amplify or AWS CLI.

**Note**
AWS Amplify includes a JavaScript library for recording events from web client applications, and a library for recording events in server code. For more information, see Amplify - Analytics

Requirements for Recording Events

To record events, you need the following:

- A dataset group that includes an Interactions dataset, which can be empty. For information on creating a dataset group and a dataset, see Preparing and Importing Data (p. 40).
- An event tracker.
- A call to the PutEvents (p. 259) operation.

You can start out with an empty Interactions dataset and, when you have recorded enough data, train the model using only new recorded events. The minimum data requirements to train a model are:

- 1000 records of combined interaction data (after filtering by eventType and eventValueThreshold, if provided)
- 25 unique users with at least 2 interactions each

How Recorded Events Influence Recommendations

Once you create a campaign, Amazon Personalize automatically uses new recorded event data for existing items (items you included in the data you used to train the latest model) when generating recommendations for the user. This does not require retraining the model (unless you are using the SIMS or popularity-count recipes).

Instead, the new recorded event data is added to the user's history. Amazon Personalize then uses the modified data when generating recommendations for the user (and this user only).

- For recorded events for *new items* (items you did not include in the data you used to train the model), if you trained your model (solution version) with the User-Personalization recipe, Amazon Personalize automatically updates the model every two hours, and after each update the new items influence recommendations. See User-Personalization Recipe (p. 66).

For any other recipe, you must re-train the model for the new records to influence recommendations. Amazon Personalize stores recorded events for new items and, once you create a new solution version (train a new model), this new data will influence Amazon Personalize recommendations for the user.

- For recorded events for *new users* (users that were not included in the data you used to train the model), recommendations will initially be for popular items only. Recommendations will be more relevant as you record more events for the user. Amazon Personalize stores the new user data, so you can also retrain the model for more relevant recommendations.

Topics

- Creating a Dataset Group and a Dataset (p. 59)
Creating a Dataset Group and a Dataset

If you went through the Getting Started (p. 12) guide, you can use the same dataset group and dataset that you created. For information on creating dataset groups and datasets, see Preparing and Importing Data (p. 40).

Creating an Event Tracker

An event tracker specifies a destination dataset group for new event data. You create an event tracker by calling the CreateEventTracker (p. 169) API and supplying the dataset group Amazon Resource Name(ARN). When you create an event tracker, the response includes a tracking ID, which you pass as a parameter when you use the PutEvents operation. Amazon Personalize then appends the new event data to the Interactions dataset of the dataset group you specify in your event tracker.

Note
Only one event tracker can be associated with a dataset group. You will get an error if you call CreateEventTracker using the same dataset group as an existing event tracker.

Python

```python
import boto3
personalize = boto3.client('personalize')
response = personalize.create_event_tracker(
    name='MovieClickTracker',
    datasetGroupArn='arn:aws:personalize:us-west-2:acct-id:dataset-group/MovieClickGroup'
)
print(response['eventTrackerArn'])
print(response['trackingId'])
```

AWS CLI

```bash
aws personalize create-event-tracker \
    --name MovieClickTracker \
    --dataset-group-arn arn:aws:personalize:us-west-2:acct-id:dataset-group/MovieClickGroup
```

The event tracker ARN and tracking ID display, for example:

```json
{
    "eventTrackerArn": "arn:aws:personalize:us-west-2:acct-id:event-tracker/MovieClickTracker",
    "trackingId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
}
```
PutEvents Operation

To record events, you call the `PutEvents` operation. The following example shows a `PutEvents` call that passes one event that contains the minimum required information. The corresponding Interactions schema is shown, along with an example row from the Interactions dataset.

Your application generates the `sessionId` when a user first visits your website or uses your application. Amazon Personalize uses the `sessionId` to associate events with the user before they log in (is anonymous). After the user logs in and you send an event including the `userId`, Amazon Personalize associates the previously anonymous historical event data with their `userId` by matching the `sessionId`. This creates a continuous event history that includes events that occurred when the user was anonymous.

The event list is an array of `Event` objects. An `eventType` is required for each event, but in this example, `eventType` data is not used in training because it is not included in the schema. You can provide a placeholder value to satisfy the requirement. The `properties` key is a string map (key-value pairs) of event-specific data. In this case, just the item ID is specified.

The `userId`, `itemId`, and `sentAt` parameters map to the USER_ID, ITEM_ID, and TIMESTAMP fields of a corresponding historical Interactions dataset. For more information, see Datasets and Schemas.

**Corresponding Interactions Schema**

| Interactions schema: USER_ID, ITEM_ID, TIMESTAMP |
| Interactions dataset: user123, item-xyz, 1543631760 |

**Code Example**

**Python**

```python
import boto3

personalize_events = boto3.client(service_name='personalize-events')

personalize_events.put_events(
    trackingId = 'tracking_id',
    userId= 'USER_ID',
    sessionId = 'session_id',
    eventList = [{
        'sentAt': TIMESTAMP,
        'eventType': 'EVENT_TYPE',
        'properties': '{"itemId": "ITEM_ID"}'}
    ])
```

**AWS CLI**

```bash
aws personalize-events put-events \
    --tracking-id tracking_id \
    --user-id USER_ID \
    --session-id session_id \
    --event-list '[{"\n        "sentAt": "TIMESTAMP",\n        "eventType": "EVENT_TYPE",\n        "properties": "{"itemId": "ITEM_ID"}"\n    }]'``
After this example, you would proceed to train a model using only the required properties.

The next example shows how to submit data that would train on the event value. It also demonstrates the passing of multiple events of different types (‘like’ and ‘rating’). In this case, you must specify the event type to train on in the CreateSolution (p. 176) operation (see Events and Solutions below). The example also shows the recording of an extra property, numRatings, that is used as metadata by certain recipes.

Interactions schema: USER_ID, ITEM_ID, TIMESTAMP, EVENT_TYPE, EVENT_VALUE, NUM_RATINGS
Interactions dataset: user123, movie_xyz, 1543531139, rating, 5, 12
user321, choc-ghana, 1543531760, like, 4
user111, choc-fake, 1543557118, like, 3

Python

```python
import boto3
import json

personalize_events = boto3.client(service_name='personalize-events')

personalize_events.put_events(
    trackingId = 'tracking_id',
    userId= 'user555',
    sessionId = 'session1',
    eventList = [{
        'eventId': 'event1',
        'sentAt': '1553631760',
        'eventType': 'like',
        'properties': json.dumps({
            'itemId': 'choc-panama',
            'eventValue': 4,
            'numRatings': 0
        })
    }, {
        'eventId': 'event2',
        'sentAt': '1553631782',
        'eventType': 'rating',
        'properties': json.dumps({
            'itemId': 'movie_ten',
            'eventValue': 3,
            'numRatings': 13
        })
    }])
```

AWS CLI

```
aws personalize-events put-events \
    --tracking-id tracking_id \
    --user-id user555 \
    --session-id session1 \
    --event-list '{
        "eventId": "event1",
        "sentAt": "1553631760",
        "eventType": "like",
        "properties": "{\"itemId\": \"choc-panama\", \"eventValue\": \"true\"}"
    }, {
        "eventId": "event2",
        "sentAt": "1553631782",
        "eventType": "rating",
        "properties": "{\"itemId\": \"movie_ten\", \"eventValue\": \"4\", \"numRatings\": \"13\"}"
    }
```
Recording Impressions Data

Impressions are lists of items that were visible to a user when they interacted with (for example, clicked or watched) a particular item. For information on the implicit and explicit impressions Amazon Personalize can model, see Impressions Data (p. 35).

Important
If you provide conflicting implicit and explicit impression data in your PutEvents requests, Amazon Personalize uses the explicit impressions by default.

To record the Amazon Personalize recommendations you show your user as impressions data, include the recommendationId in your PutEvents (p. 259) request and Amazon Personalize will derive the implicit impressions based on your recommendation data.

To manually record impressions data for an event, list the impressions in the PutEvents (p. 259) command’s impression input parameter. The impressions are added to the recommendation that you specify in the recommendationId parameter:

```python
import boto3

personalize_events = boto3.client(service_name='personalize-events')

personalize_events.put_events(
    trackingId = 'tracking_id',
    userId= 'userId',
    sessionId = 'sessionId',
    eventList = [{
        'eventId': 'event1',
        'eventType': 'rating',
        'sentAt': 1553631760,
        'itemId': 'itemId',
        'recommendationId': 'recommendation id',
        'impression': ['itemId1', 'itemId2', 'itemId3']
    }]
)
```

Event Metrics

To monitor the type and number of events sent to Amazon Personalize, use Amazon CloudWatch metrics. For more information, see Monitoring Amazon Personalize (p. 138).

Events and Solutions

When training a model that uses event data, two parameters of the CreateSolution (p. 176) operation are relevant. The eventType parameter must be specified when multiple event types are recorded. The eventType indicates which type of event Amazon Personalize uses as the label for model training.
The `eventValueThreshold` parameter of the `SolutionConfig` object creates an event filter. When this parameter is specified, only events with a value greater than or equal to the threshold are used for training the model. You must specify the event type when using `eventValueThreshold`.

Sample Jupyter Notebook

For a sample Jupyter notebook that shows how to use Amazon Personalize to react to real-time behavior of users using an event tracker and the `PutEvents (p. 259)` operation, see `2.View_Campaign_And_Interactions.ipynb` in the `getting_started` folder of the `amazon-personalize-samples` GitHub repository.
Creating a Solution

Once you have finished Preparing and Importing Data (p. 40), you are ready to create a Solution. A Solution refers to the combination of an Amazon Personalize recipe, customized parameters, and one or more solution versions (trained models). Once you create a solution with a solution version, you can create a campaign (p. 104) to deploy the solution version and get recommendations.

To create a solution in Amazon Personalize, you do the following:

1. **Choose a recipe** – A recipe is an Amazon Personalize term specifying an appropriate algorithm to train for a given use case. See Step 1: Choosing a Recipe (p. 64).
2. **Configure a solution** – Customize solution parameters and recipe-specific hyperparameters so the model meets your specific business needs. See Step 2: Configuring a Solution (p. 92).
3. **Create a solution version (train a model)** – Train the machine learning model Amazon Personalize will use to generate recommendations for your customers. See Step 3: Creating a Solution Version (p. 99).
4. **Evaluate the solution version** – Use the metrics Amazon Personalize generates from the new solution version to evaluate the performance of the model. See Step 4: Evaluating the Solution Version (p. 101).

**Step 1: Choosing a Recipe**

Amazon Personalize provides recipes, based on common use cases, for training models. A recipe is an Amazon Personalize term specifying an appropriate algorithm to train for a given use case. With recipes, you can create a personalization system without prior machine learning experience.

Amazon Personalize recipes use the following during training:

- Predefined attributes of your data
- Predefined feature transformations
- Predefined algorithms
- Initial parameter settings for the algorithms

To optimize your model, you can override many of these parameters when you create a solution. For more information, see Hyperparameters and HPO (p. 94).

Choose a specific recipe based on what you want to accomplish and how familiar you are with the recipes. Each recipe is designed for a specific use case. When creating a solution, choose the recipe that best fits your needs.

**Amazon Personalize Recipes**

Amazon Personalize provides three types of recipes. Besides behavioral differences, each type has different requirements for getting recommendations, as shown in the following table.

<table>
<thead>
<tr>
<th>Recipe type</th>
<th>API</th>
<th>Requirements</th>
<th>Recipes</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_PERSONALIZATION</td>
<td>GetRecommendations (p. 269)</td>
<td>userId: Required</td>
<td>User-Personalization (p. 66)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>itemId: Optional</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>inputList: NA</td>
<td>Popularity-Count (p. 74)</td>
</tr>
</tbody>
</table>
### Viewing Available Amazon Personalize Recipes

To see a list of available recipes:

- In the Amazon Personalize console, choose a dataset group. From the navigation pane, choose **Solutions and recipes**, and choose the **Recipes** tab.
- With the AWS SDK for Python (Boto3), call the `ListRecipes (p. 246)` API.
- With the AWS CLI, use the following command.

```bash
aws personalize list-recipes
```

To get information about a recipe using the SDK for Python (Boto3), call the `DescribeRecipe (p. 216)` API. To get information about a recipe using the AWS CLI, use the following command.

```bash
aws personalize describe-recipe --recipe-arn recipe_arn
```

### USER_PERSONALIZATION Recipes

USER_PERSONALIZATION recipes predict the items that a user will interact with based on Interactions, Items, and Users datasets. If you are building a recommendation system that provides personalized recommendations for each of your users, you should train your model with a USER_PERSONALIZATION recipe.

USER_PERSONALIZATION recipes are as follows:

- User-Personalization Recipe (p. 66)
User-Personalization Recipe

The User-Personalization (aws-user-personalization) recipe is optimized for all personalized recommendation scenarios. It predicts the items that a user will interact with based on Interactions, Items, and Users datasets. When recommending items, it uses automatic item exploration.

With automatic exploration, Amazon Personalize automatically tests different item recommendations, learns from how users interact with these recommended items, and boosts recommendations for items that drive better engagement and conversion. This improves item discovery and engagement when you have a fast-changing catalog, or when new items, such as news articles or promotions, are more relevant to users when fresh.

You can balance how much to explore (where items with less interactions data or relevance are recommended more frequently) against how much to exploit (where recommendations are based on what we know or relevance). Amazon Personalize automatically adjusts future recommendations based on implicit user feedback.

Automatic Updates

With User-Personalization, Amazon Personalize automatically updates the latest model (solution version) every two hours to include new data. With each update, Amazon Personalize updates the solution version with the latest item information and adjusts the exploration according to implicit feedback from users. This allows Amazon Personalize to gauge item quality based on new interactions for already explored items and continually update item exploration.

Note
There is no cost for automatic updates.

Update Requirements

Amazon Personalize automatically updates only the latest solution version trained with `trainingMode` set to `FULL` and only if you provide new item or interactions data since the last automatic update. If you have trained a new solution version, Amazon Personalize will not automatically update older solution versions that you have deployed in a campaign. Updates also do not occur if you have deleted your dataset.

Note
Amazon Personalize automatically updates only solution version created on or after November 17, 2020.

Working with Impressions Data

Unlike other recipes, which solely use positive interactions (clicking, watching, or purchasing), the User-Personalization recipe can also use impressions data. Impressions are lists of items that were visible to the user when they interacted with (clicked, watched, purchased, and so on) a particular item.

Using this information, a solution trained with the User-Personalization recipe can quickly infer the suitability of new items based on how frequently an item has been ignored, and change recommendation accordingly. For more information see Formatting Impressions Data (p. 47).

Properties and Hyperparameters

The User-Personalization recipe has the following properties:

- **Name** – `aws-user-personalization`
- **Recipe Amazon Resource Name (ARN)** – `arn:aws:personalize:::recipe/aws-user-personalization`
- **Algorithm ARN** – `arn:aws:personalize:::algorithm/aws-user-personalization`

For more information, see Step 1: Choosing a Recipe (p. 64).

The following table describes the hyperparameters for the User-Personalization recipe. A hyperparameter is an algorithm parameter that you can adjust to improve model performance. Algorithm hyperparameters control how the model performs. Featurization hyperparameters control how to filter the data to use in training. The process of choosing the best value for a hyperparameter is called hyperparameter optimization (HPO). For more information, see Hyperparameters and HPO (p. 94).

The table also provides the following information for each hyperparameter:

- **Range**: [lower bound, upper bound]
- **Value type**: Integer, Continuous (float), Categorical (Boolean, list, string)
- **HPO tunable**: Can the parameter participate in HPO?

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithm Hyperparameters</strong></td>
<td></td>
</tr>
<tr>
<td>hidden_dimension</td>
<td>The number of hidden variables used in the model. Hidden variables recreate users' purchase history and item statistics to generate ranking scores. Specify a greater number of hidden dimensions when your Interactions dataset includes more complicated patterns. Using more hidden dimensions requires a larger dataset and more time to process. To decide on the optimal value, use HPO. To use HPO, set performHPO to true when you call CreateSolution (p. 176) and CreateSolutionVersion (p. 181) operations.</td>
</tr>
<tr>
<td></td>
<td>Default value: 149</td>
</tr>
<tr>
<td></td>
<td>Range: [32, 256]</td>
</tr>
<tr>
<td></td>
<td>Value type: Integer</td>
</tr>
<tr>
<td></td>
<td>HPO tunable: Yes</td>
</tr>
<tr>
<td>bptt</td>
<td>Determines whether to use the back-propagation through time technique. Back-propagation through time is a technique that updates weights in recurrent neural network-based algorithms. Use bptt for long-term credits to connect delayed rewards to early events. For example, a delayed reward can be a purchase made after several clicks. An early event can be an initial click. Even within the same event types, such as a click, it's a good idea to consider long-term effects and maximize the total rewards. To consider long-term effects, use larger bptt values. Using a larger bptt value requires larger datasets and more time to process.</td>
</tr>
<tr>
<td></td>
<td>Default value: 32</td>
</tr>
<tr>
<td></td>
<td>Range: [2, 32]</td>
</tr>
<tr>
<td></td>
<td>Value type: Integer</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>recency_mask</td>
<td>Determines whether the model should consider the latest popularity trends in the Interactions dataset. Latest popularity trends might include sudden changes in the underlying patterns of interaction events. To train a model that places more weight on recent events, set recency_mask to true. To train a model that equally weighs all past interactions, set recency_mask to false. To get good recommendations using an equal weight, you might need a larger training dataset.</td>
</tr>
<tr>
<td></td>
<td>Default value: True</td>
</tr>
<tr>
<td></td>
<td>Range: True or False</td>
</tr>
<tr>
<td></td>
<td>Value type: Boolean</td>
</tr>
</tbody>
</table>

**Featurization Hyperparameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_user_history_length_percentile</td>
<td>The minimum percentile of user history lengths to include in model training. History length is the total amount of data about a user. Use min_user_history_length_percentile to exclude a percentage of users with short history lengths. Users with a short history often show patterns based on item popularity instead of the user's personal needs or wants. Removing them can train models with more focus on underlying patterns in your data. Choose an appropriate value after you review user history lengths, using a histogram or similar tool. We recommend setting a value that retains the majority of users, but removes the edge cases.</td>
</tr>
<tr>
<td></td>
<td>For example, setting min_user_history_length_percentile to 0.05 and max_user_history_length_percentile to 0.95 includes all users except those with history lengths at the bottom or top 5%.</td>
</tr>
<tr>
<td></td>
<td>Default value: 0.0</td>
</tr>
<tr>
<td></td>
<td>Range: [0.0, 1.0]</td>
</tr>
<tr>
<td></td>
<td>Value type: Float</td>
</tr>
<tr>
<td></td>
<td>HPO tunable: No</td>
</tr>
</tbody>
</table>
### Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>max_user_history_length_percentile</strong></td>
<td>The maximum percentile of user history lengths to include in model training. <em>History length</em> is the total amount of data about a user. Use <code>max_user_history_length_percentile</code> to exclude a percentage of users with long history lengths because data for these users tend to contain noise. For example, a robot might have a long list of automated interactions. Removing these users limits noise in training. Choose an appropriate value after you review user history lengths using a histogram or similar tool. We recommend setting a value that retains the majority of users but removes the edge cases. For example, setting <code>min_user_history_length_percentile</code> to 0.05 and <code>max_user_history_length_percentile</code> to 0.95 includes all users except those with history lengths at the bottom or top 5%. Default value: 0.99 Range: [0.0, 1.0] Value type: Float HPO tunable: No</td>
</tr>
<tr>
<td><strong>Item Exploration Campaign Configuration Hyperparameters</strong></td>
<td><strong>exploration_weight</strong> Determines how frequently recommendations include items with less interactions data or relevance. The closer the value is to 1.0, the more exploration. At zero, no exploration occurs and recommendations are based on current data (relevance). For more information see the section called “CampaignConfig” (p. 290). Default value: 0.3 Range: [0.0, 1.0] Value type: Float HPO tunable: No</td>
</tr>
</tbody>
</table>
## Exploration Item Age Cut Off

Determines items to be explored based on time frame since latest interaction. Provide the maximum item age, in days since the latest interaction, to define the scope of item exploration. The larger the value, the more items are considered during exploration. For more information see the section called “CampaignConfig” (p. 290).

Default value: 30.0

Range: Positive floats

Value type: Float

HPO tunable: No

### Training with the User-Personalization recipe (Console)

To use the User-Personalization recipe to generate recommendations in the console, first train a new solution version using the recipe. Then deploy a campaign using the solution version and use the campaign to get recommendations.

#### Training a new solution version with the User-Personalization recipe (console)

1. Open the Amazon Personalize console at [https://console.aws.amazon.com/personalize/](https://console.aws.amazon.com/personalize/) and sign into your account.
2. Create a dataset group with a new schema and upload your dataset with impressions data. Optionally include `CREATION_TIMESTAMP` (p. 38) data in your Items dataset so Amazon Personalize can more accurately calculate the age of an item and identify cold items.

   For more information on creating dataset groups and uploading training data, see Step 1: Import Training Data (p. 13) in the Getting Started tutorial.

3. On the **Dataset groups** page, choose the new dataset group that contains the dataset or datasets with impressions data.
4. In the navigation pane, choose **Solutions and recipes** and choose **Create solution**.
5. On the **Create solution** page, for the **Solution name**, enter the name of your new solution.
6. For **Recipe**, choose **aws-user-personalization**. The **Solution configuration** section appears providing several configuration options.
7. Specify the following:
   - **Event type**: If your data has multiple event types, enter the event type, such as click or download, to use for training.
   - **Event value threshold**: Enter a value to define which events to use for training. Only events with a value greater than or equal to this value will be used for training.

   For example, suppose that your application rates user interest in videos based on the percentage of a video watched by the user, with values greater than 70 indicating high interest. If you want to use only high-interest events for training, for the **Event type** enter `watched` and for **Event value threshold** enter 70. Only videos with a `watched` value greater than 70 will be used for training.
8. Optionally configure hyperparameters for your solution. For a list of User-Personalization recipe properties and hyperparameters, see Properties and Hyperparameters (p. 66).
9. Choose **Next**. You can review your settings on the **Create solution version** page.
10. Choose **Finish** to create the solution version.
On the solution details page, you can track training progress in the Solution versions section. When training is complete, the status is Active.

Creating a campaign and getting recommendations (console)

When your solution version status is Active you are ready to create your campaign and get recommendations as follows:

1. On either the solution details page or the Campaigns page, choose Create new campaign.
2. On the Create new campaign page, for Campaign details, provide the following information:
   - **Campaign name**: Enter the name of the campaign. The text you enter here appears on the Campaign dashboard and details page.
   - **Solution**: Choose the solution that you just created.
   - **Solution version ID**: Choose the ID of the solution version that you just created.
   - **Minimum provisioned transactions per second**: Set the minimum provisioned transactions per second that Amazon Personalize supports. For more information, see the CreateCampaign (p. 157) operation.
3. For Campaign configuration, provide the following information:
   - **Exploration weight**: Configure how much to explore, where recommendations include items with less interactions data or relevance more frequently the more exploration you specify. The closer the value is to 1, the more exploration. At zero, no exploration occurs and recommendations are based on current data (relevance).
   - **Exploration item age cut off**: Enter the maximum item age, in days since the latest interaction, to define the scope of item exploration. To increase the number of items Amazon Personalize considers during exploration, enter a greater value. For example, if you enter 10, only items with interactions data from the 10 days since the latest interaction in the dataset are considered during exploration.

   **Note**
   Recommendations might include items without interactions data from outside this time frame. This is because these items are relevant to the user's interests, and exploration wasn't required to identify them.
4. Choose Create campaign.
5. On the campaign details page, when the campaign status is Active, you can use the campaign to get recommendations and record impressions. For more information, see Step 4: Get Recommendations (p. 20) in "Getting Started."

Amazon Personalize automatically updates your latest solution version every two hours to include new data. Your campaign automatically uses the updated solution version. For more information see Automatic Updates (p. 66).

To manually update the campaign, you first create and train a new solution version using the console or the CreateSolutionVersion (p. 181) operation, with trainingMode set to update. You then manually update the campaign on the Campaign page of the console or by using the UpdateCampaign (p. 256) operation.

   **Note**
   Amazon Personalize doesn't automatically update solution versions you created before November 17, 2020.
Training with the User-Personalization Recipe (Python SDK)

When you have created a dataset group and uploaded your dataset(s) with impressions data, you can train a solution with the User-Personalization recipe. Optionally include CREATION_TIMESTAMP (p. 38) data in your dataset so Amazon Personalize can more accurately calculate the age of an item and identify cold start items. For more information on creating dataset groups and uploading training data see Datasets and Schemas (p. 32).

To train a solution with the User-Personalization recipe using the AWS SDK

1. Create a new solution using the create_solution method.

   Replace solution name with your solution name and dataset group arn with the Amazon Resource Name (ARN) of your dataset group.

   ```python
   import boto3
   personalize = boto3.client('personalize')
   print('Creating solution')
   create_solution_response = personalize.create_solution(name='solution name',
   recipeArn= 'arn:aws:personalize:::recipe/aws-user-
   personalization',
   datasetGroupArn = 'dataset group arn',
   )
   solution_arn = create_solution_response['solutionArn']
   print('solution_arn: ', solution_arn)
   ```

   For a list of aws-user-personalization recipe properties and hyperparameters, see Properties and Hyperparameters (p. 66).

2. Create a new solution version with the updated training data and set trainingMode to FULL using the following code snippet. Replace the solution arn with the ARN of your solution.

   ```python
   import boto3
   personalize = boto3.client('personalize')
   create_solution_version_response = personalize.create_solution_version(solutionArn =
   'solution arn',
   trainingMode='FULL')
   new_solution_version_arn = create_solution_version_response['solutionVersionArn']
   print('solution_version_arn:', new_solution_version_arn)
   ```

3. When Amazon Personalize is finished creating your solution version, create your campaign with the following parameters:

   - Provide a new campaign name and the solution version arn generated in step 2.
   - Modify the explorationWeight item exploration configuration hyperparameter to configure how much to explore. Items with less interactions data or relevance are recommended more frequently the closer the value is to 1.0. The default value is 0.3.
   - Modify the explorationItemAgeCutOff item exploration configuration hyperparameter parameter to provide the maximum duration, in days relative to the latest interaction, for which items should be explored. The larger the value, the more items are considered during exploration.

   Use the following Python snippet to create a new campaign with an emphasis on exploration with exploration cut-off at 30 days. Creating a campaign usually takes a few minutes but can take over an hour.
import boto3
personalize = boto3.client('personalize')

campaign_arn = create_campaign_response['campaignArn']
print('campaign_arn:', campaign_arn)

With User-Personalization, Amazon Personalize automatically updates your solution version every two hours to include new data. Your campaign automatically uses the updated solution version. For more information see Automatic Updates (p. 66).

To manually update the campaign, you first create and train a new solution version using the console or the CreateSolutionVersion (p. 181) operation, with trainingMode set to update. You then manually update the campaign on the Campaign page of the console or by using the UpdateCampaign (p. 256) operation.

Note
Amazon Personalize doesn't automatically update solution versions you created before November 17, 2020.

Getting Recommendations and Recording Impressions (Python SDK)

When your campaign is created, you can use it to get recommendations for a user and record impressions. For information on getting batch recommendations using the AWS SDK see Getting Batch Recommendations (AWS Python SDK) (p. 112).

To get recommendations and record impressions

1. Call the get_recommendations method. Change the campaign arn to the ARN of your new campaign and user id to the userId of the user.

```python
import boto3

rec_response = personalize_runtime.get_recommendations(campaignArn = 'campaign arn',
    userId = 'user id')
print(rec_response['recommendationId'])
```

2. Create a new event tracker for sending PutEvents requests. Replace event tracker name with the name of your event tracker and dataset group arn with the ARN of your dataset group.

```python
import boto3

personalize = boto3.client('personalize')

event_tracker_arn = event_tracker_response['eventTrackerArn']
event_tracking_id = event_tracker_response['trackingId']
```
print('eventTrackerArn:{},
  eventTrackingId:{}'.format(event_tracker_arn, event_tracking_id))

3. Use the recommendationId from step 1 and the event tracking id from step 2 to create a new PutEvents request. This request logs the new impression data from the user’s session. Change the user id to the ID of the user.

```python
import boto3

personalize_events.put_events(
    trackingId = 'event tracking id',
    userId= 'user id',
    sessionId = '1',
    eventList = [{
        'sentAt': datetime.now().timestamp(),
        'eventType' : 'click',
        'itemId' : rec_response['itemList'][0]['itemId'],
        'recommendationId': rec_response['recommendationId'],
        'impression': [item['itemId'] for item in rec_response['itemList']],
    }]
)
```

Sample Jupyter Notebook

For a sample Jupyter notebook that shows how to use the User-Personalization recipe, see User Personalization with Exploration.

Popularity-Count Recipe

Popularity-count returns the top popular items from a dataset. A popular item is defined by the number of times it occurs in the dataset. The recipe returns the same popular items for all users. Popularity-count is a good starting point to compare against other recipes.

This predefined recipe has the following properties:

- **Name** – aws-popularity-count
- **Recipe ARN** – arn:aws:personalize:::recipe/aws-popularity-count
- **Algorithm ARN** – arn:aws:personalize:::algorithm/aws-popularity-count
- **Feature Transformation ARN** – arn:aws:personalize:::feature-transformation/sims
- **Recipe type** – USER_PERSONALIZATION

Popularity-count has no exposed hyperparameters.

Legacy User Personalization Recipes

**Note**

We recommend using the aws-user-personalization (User-Personalization) recipe over the legacy HRNN recipes. User-Personalization improves upon and unifies the functionality offered by the HRNN recipes. For more information, see User-Personalization Recipe (p. 66).

The following are legacy USER_PERSONALIZATION recipes.

- **HRNN Recipe (Legacy)** (p. 75)
- **HRNN-Coldstart Recipe (Legacy)** (p. 81)
- **HRNN-Metadata Recipe (Legacy)** (p. 78)
HRNN Recipe (Legacy)

Note
We recommend using the aws-user-personalization (User-Personalization) recipe over the legacy HRNN recipes. User-Personalization improves upon and unifies the functionality offered by the HRNN recipes. For more information, see User-Personalization Recipe (p. 66).

The Amazon Personalize hierarchical recurrent neural network (HRNN) recipe models changes in user behavior to provide recommendations during a session. A session is a set of user interactions within a given timeframe with a goal of finding a specific item to fill a need, for example. By weighing a user's recent interactions higher, you can provide more relevant recommendations during a session.

HRNN accommodates user intent and interests, which can change over time. It takes ordered user histories and automatically weights them to make better inferences. HRNN uses a gating mechanism to model the discount weights as a learnable function of the items and timestamps.

Amazon Personalize derives the features for each user from your dataset. If you have done real-time data integration, these features are updated in real time according to user activity. To get a recommendation, you provide only the USER_ID. If you also provide an ITEM_ID, Amazon Personalize ignores it.

The HRNN recipe has the following properties:

- **Name** – aws-hrnn
- **Recipe Amazon Resource Name (ARN)** – arn:aws:personalize:::recipe/aws-hrnn
- **Algorithm ARN** – arn:aws:personalize:::algorithm/aws-hrnn
- **Feature transformation ARN** – arn:aws:personalize:::feature-transformation/JSON-percentile-filtering
- **Recipe type** – USER_PERSONALIZATION

The following table describes the hyperparameters for the HRNN recipe. A hyperparameter is an algorithm parameter that you can adjust to improve model performance. Algorithm hyperparameters control how the model performs. Featurization hyperparameters control how to filter the data to use in training. The process of choosing the best value for a hyperparameter is called hyperparameter optimization (HPO). For more information, see Hyperparameters and HPO (p. 94).

The table also provides the following information for each hyperparameter:

- **Range**: [lower bound, upper bound]
- **Value type**: Integer, Continuous (float), Categorical (Boolean, list, string)
- **HPO tunable**: Can the parameter participate in HPO?

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithm Hyperparameters</strong></td>
<td></td>
</tr>
<tr>
<td>hidden_dimension</td>
<td>The number of hidden variables used in the model. <em>Hidden variables</em> recreate users' purchase history and item statistics to generate ranking scores. Specify a greater number of hidden dimensions when your Interactions dataset includes more complicated patterns. Using more hidden dimensions requires a larger dataset and more time to process. To decide on the optimal value, use HPO. To use HPO, set performHPO to true when you call CreateSolution (p. 176) and CreateSolutionVersion (p. 181) operations.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td><strong>Default value</strong>: 43</td>
</tr>
<tr>
<td></td>
<td><strong>Range</strong>: [32, 256]</td>
</tr>
<tr>
<td></td>
<td><strong>Value type</strong>: Integer</td>
</tr>
<tr>
<td></td>
<td><strong>HPO tunable</strong>: Yes</td>
</tr>
<tr>
<td>bptt</td>
<td>Determines whether to use the back-propagation through time technique. <em>Back-propagation through time</em> is a technique that updates weights in recurrent neural network-based algorithms. Use bptt for long-term credits to connect delayed rewards to early events. For example, a delayed reward can be a purchase made after several clicks. An early event can be an initial click. Even within the same event types, such as a click, it's a good idea to consider long-term effects and maximize the total rewards. To consider long-term effects, use larger bptt values. Using a larger bptt value requires larger datasets and more time to process.</td>
</tr>
<tr>
<td></td>
<td><strong>Default value</strong>: 32</td>
</tr>
<tr>
<td></td>
<td><strong>Range</strong>: [2, 32]</td>
</tr>
<tr>
<td></td>
<td><strong>Value type</strong>: Integer</td>
</tr>
<tr>
<td></td>
<td><strong>HPO tunable</strong>: Yes</td>
</tr>
<tr>
<td>recency_mask</td>
<td>Determines whether the model should consider the latest popularity trends in the Interactions dataset. Latest popularity trends might include sudden changes in the underlying patterns of interaction events. To train a model that places more weight on recent events, set recency_mask to true. To train a model that equally weighs all past interactions, set recency_mask to false. To get good recommendations using an equal weight, you might need a larger training dataset.</td>
</tr>
<tr>
<td></td>
<td><strong>Default value</strong>: True</td>
</tr>
<tr>
<td></td>
<td><strong>Range</strong>: True or False</td>
</tr>
<tr>
<td></td>
<td><strong>Value type</strong>: Boolean</td>
</tr>
<tr>
<td></td>
<td><strong>HPO tunable</strong>: Yes</td>
</tr>
</tbody>
</table>

**Featurization Hyperparameters**
### Name: min_user_history_length_percentile

- **Description:** The minimum percentile of user history lengths to include in model training. *History length* is the total amount of data about a user. Use `min_user_history_length_percentile` to exclude a percentage of users with short history lengths. Users with a short history often show patterns based on item popularity instead of the user's personal needs or wants. Removing them can train models with more focus on underlying patterns in your data. Choose an appropriate value after you review user history lengths, using a histogram or similar tool. We recommend setting a value that retains the majority of users, but removes the edge cases.

  For example, setting `min_user_history_length_percentile` to 0.05 and `max_user_history_length_percentile` to 0.95 includes all users except those with history lengths at the bottom or top 5%.

  - **Default value:** 0.0
  - **Range:** [0.0, 1.0]
  - **Value type:** Float
  - **HPO tunable:** No

---

### Name: max_user_history_length_percentile

- **Description:** The maximum percentile of user history lengths to include in model training. *History length* is the total amount of data about a user. Use `max_user_history_length_percentile` to exclude a percentage of users with long history lengths because data for these users tend to contain noise. For example, a robot might have a long list of automated interactions. Removing these users limits noise in training. Choose an appropriate value after you review user history lengths using a histogram or similar tool. We recommend setting a value that retains the majority of users but removes the edge cases.

  For example, setting `min_user_history_length_percentile` to 0.05 and `max_user_history_length_percentile` to 0.95 includes all users except those with history lengths at the bottom or top 5%.

  - **Default value:** 0.99
  - **Range:** [0.0, 1.0]
  - **Value type:** Float
  - **HPO tunable:** No
HRNN Sample Notebooks

The following Jupyter notebooks show how to use the HRNN recipe:

- For a sample notebook that shows how to use HRNN with a hold-out set, see Amazon Personalize with temporal evaluation on hold-out set.
- For a sample notebook that shows how to use HRNN with the PutEvents API, see Using the PutEvents API with Amazon Personalize.

HRNN-Metadata Recipe (Legacy)

Note
We recommend using the aws-user-personalizaton (User-Personalization) recipe over the legacy HRNN recipes. User-Personalization improves upon and unifies the functionality offered by the HRNN recipes. For more information, see User-Personalization Recipe (p. 66).

The HRNN-Metadata recipe predicts the items that a user will interact with. It is similar to the HRNN (p. 75) recipe, with additional features derived from contextual, user, and item metadata (from Interactions, Users, and Items datasets, respectively). HRNN-Metadata provides accuracy benefits over non-metadata models when high quality metadata is available. Using this recipe might require longer training times.

The HRNN-Metadata recipe has the following properties:

- Name – aws-hrnn-metadata
- Recipe Amazon Resource Name (ARN) – arn:aws:personalize:::recipe/aws-hrnn-metadata
- Algorithm ARN – arn:aws:personalize:::algorithm/aws-hrnn-metadata
- Feature transformation ARN – arn:aws:personalize:::feature-transformation/featurize_metadata
- Recipe type – USER_PERSONALIZATION

The following table describes the hyperparameters for the HRNN-Metadata recipe. A hyperparameter is an algorithm parameter that you can adjust to improve model performance. Algorithm hyperparameters control how the model performs. Featurization hyperparameters control how to filter the data to use in training. The process of choosing the best value for a hyperparameter is called hyperparameter optimization (HPO). For more information, see Hyperparameters and HPO (p. 94).

The table also provides the following information for each hyperparameter:

- Range: [lower bound, upper bound]
- Value type: Integer, Continuous (float), Categorical (Boolean, list, string)
- HPO tunable: Can the parameter participate in hyperparameter optimization (HPO)?

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hidden_dimension</td>
<td>The number of hidden variables used in the model. Hidden variables recreate users’ purchase history and item statistics to generate ranking scores. Specify a greater number of hidden dimensions when your Interactions dataset includes more complicated patterns. Using more hidden dimensions requires a larger dataset and more time to process. To decide on the optimal value, use HPO. To use HPO, set performHPO to</td>
</tr>
</tbody>
</table>
### Name Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| true      | when you call `CreateSolution` (p. 176) and `CreateSolutionVersion` (p. 181) operations.  
Default value: 43  
Range: [32, 256]  
Value type: Integer  
HPO tunable: Yes |
| bptt      | Determines whether to use the back-propagation through time technique. `Back-propagation through time` is a technique that updates weights in recurrent neural network-based algorithms. Use `bptt` for long-term credits to connect delayed rewards to early events. For example, a delayed reward can be a purchase made after several clicks. An early event can be an initial click. Even within the same event types, such as a click, it's a good idea to consider long-term effects and maximize the total rewards. To consider long-term effects, use larger `bptt` values. Using a larger `bptt` value requires larger datasets and more time to process.  
Default value: 32  
Range: [2, 32]  
Value type: Integer  
HPO tunable: Yes |
| recency_mask | Determines whether the model should consider the latest popularity trends in the Interactions dataset. Latest popularity trends might include sudden changes in the underlying patterns of interaction events. To train a model that places more weight on recent events, set `recency_mask` to `true`. To train a model that equally weighs all past interactions, set `recency_mask` to `false`. To get good recommendations using an equal weight, you might need a larger training dataset.  
Default value: True  
Range: True or False  
Value type: Boolean  
HPO tunable: Yes |

### Featurization Hyperparameters
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_user_history_length_percentile</td>
<td>The minimum percentile of user history lengths to include in model training. History length is the total amount of data about a user. Use min_user_history_length_percentile to exclude a percentage of users with short history lengths. Users with a short history often show patterns based on item popularity instead of the user's personal needs or wants. Removing them can train models with more focus on underlying patterns in your data. Choose an appropriate value after you review user history lengths, using a histogram or similar tool. We recommend setting a value that retains the majority of users, but removes the edge cases. For example, setting min_user_history_length_percentile to 0.05 and max_user_history_length_percentile to 0.95 includes all users except those with history lengths at the bottom or top 5%. Default value: 0.0 Range: [0.0, 1.0] Value type: Float HPO tunable: No</td>
</tr>
<tr>
<td>max_user_history_length_percentile</td>
<td>The maximum percentile of user history lengths to include in model training. History length is the total amount of data about a user. Use max_user_history_length_percentile to exclude a percentage of users with long history lengths because data for these users tend to contain noise. For example, a robot might have a long list of automated interactions. Removing these users limits noise in training. Choose an appropriate value after you review user history lengths using a histogram or similar tool. We recommend setting a value that retains the majority of users but removes the edge cases. For example, setting min_user_history_length_percentile to 0.05 and max_user_history_length_percentile to 0.95 includes all users except those with history lengths at the bottom or top 5%. Default value: 0.99 Range: [0.0, 1.0] Value type: Float HPO tunable: No</td>
</tr>
</tbody>
</table>
HRNN-Coldstart Recipe (Legacy)

Note
We recommend using the aws-user-personalization (User-Personalization) recipe over the legacy HRNN recipes. User-Personalization improves upon and unifies the functionality offered by the HRNN recipes. For more information, see User-Personalization Recipe (p. 66).

Use the HRNN-Coldstart recipe to predict the items that a user will interact with when you frequently add new items and interactions and want to get recommendations for those items immediately. The HRNN-Coldstart recipe is similar to the HRNN-Metadata (p. 78) recipe, but it allows you to get recommendations for new items.

In addition, you can use the HRNN-Coldstart recipe when you want to exclude from training items that have a long list of interactions either because of a recent popularity trend or because the interactions might be highly unusual and introduce noise in training. With HRNN-Coldstart, you can filter out less relevant items to create a subset for training. The subset of items, called cold items, are items that have related interaction events in the Interactions dataset. An item is considered a cold item when it has the following:

- Fewer interactions than a specified number of maximum interactions. You specify this value in the recipe's cold_start_max_interactions hyperparameter.
- A shorter relative duration than the maximum duration. You specify this value in the recipe's cold_start_max_duration hyperparameter.

To reduce the number of cold items, set a lower value for cold_start_max_interactions or cold_start_max_duration. To increase the number of cold items, set a greater value for cold_start_max_interactions or cold_start_max_duration.

HRNN-Coldstart has the following cold item limits:
- Maximum cold start items: 80,000
- Minimum cold start items: 100

If the number of cold items is outside this range, attempts to create a solution will fail.

The HRNN-Coldstart recipe has the following properties:

- **Name** – aws-hrnn-coldstart
- **Recipe Amazon Resource Name (ARN)** – arn:aws:personalize:::recipe/aws-hrnn-coldstart
- **Algorithm ARN** – arn:aws:personalize:::algorithm/aws-hrnn-coldstart
- **Feature transformation ARN** – arn:aws:personalize:::feature-transformation/featurize_coldstart
- **Recipe type** – USER_PERSONALIZATION

For more information, see Step 1: Choosing a Recipe (p. 64).

The following table describes the hyperparameters for the HRNN-Coldstart recipe. A hyperparameter is an algorithm parameter that you can adjust to improve model performance. Algorithm hyperparameters control how the model performs. Featurization hyperparameters control how to filter the data to use in training. The process of choosing the best value for a hyperparameter is called hyperparameter optimization (HPO). For more information, see Hyperparameters and HPO (p. 94).

The table also provides the following information for each hyperparameter:

- **Range**: [lower bound, upper bound]
- **Value type**: Integer, Continuous (float), Categorical (Boolean, list, string)
- **HPO tunable**: Can the parameter participate in HPO?

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hidden_dimension</td>
<td>The number of hidden variables used in the model. <em>Hidden variables</em> recreate users’ purchase history and item statistics to generate ranking scores. Specify a greater number of hidden dimensions when your Interactions dataset includes more complicated patterns. Using more hidden dimensions requires a larger dataset and more time to process. To decide on the optimal value, use HPO. To use HPO, set <code>performHPO</code> to <code>true</code> when you call <code>CreateSolution (p. 176)</code> and <code>CreateSolutionVersion (p. 181)</code> operations. Default value: 149 Range: [32, 256] Value type: Integer HPO tunable: Yes</td>
</tr>
<tr>
<td>bptt</td>
<td>Determines whether to use the back-propagation through time technique. <em>Back-propagation through time</em> is a technique that updates weights in recurrent neural network-based algorithms. Use <code>bptt</code> for long-term credits to connect delayed rewards to early events. For example, a delayed reward can be a purchase made after several clicks. An early event can be an initial click. Even within the same event types, such as a click, it's a good idea to consider long-term effects and maximize the total rewards. To consider long-term effects, use larger <code>bptt</code> values. Using a larger <code>bptt</code> value requires larger datasets and more time to process. Default value: 32 Range: [2, 32] Value type: Integer HPO tunable: Yes</td>
</tr>
<tr>
<td>recency_mask</td>
<td>Determines whether the model should consider the latest popularity trends in the Interactions dataset. Latest popularity trends might include sudden changes in the underlying patterns of interaction events. To train a model that places more weight on recent events, set <code>recency_mask</code> to <code>true</code>. To train a model that equally weighs all past interactions, set <code>recency_mask</code> to <code>false</code>. To get good recommendations using an equal weight, you might need a larger training dataset. Default value: True</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cold_start_max_interactions</td>
<td>The maximum number of user-item interactions an item can have to be considered a cold item.</td>
</tr>
<tr>
<td></td>
<td>Default value: 15</td>
</tr>
<tr>
<td></td>
<td>Range: Positive integers</td>
</tr>
<tr>
<td></td>
<td>Value type: Integer</td>
</tr>
<tr>
<td></td>
<td>HPO tunable: No</td>
</tr>
<tr>
<td></td>
<td>Featurization Hyperparameters</td>
</tr>
<tr>
<td>cold_start_max_duration</td>
<td>The maximum duration in days relative to the starting point for a user-item interaction to be considered a cold start item. To set the starting point of the user-item interaction, set the cold_start_relative_from hyperparameter.</td>
</tr>
<tr>
<td></td>
<td>Default value: 5.0</td>
</tr>
<tr>
<td></td>
<td>Range: Positive floats</td>
</tr>
<tr>
<td></td>
<td>Value type: Float</td>
</tr>
<tr>
<td></td>
<td>HPO tunable: No</td>
</tr>
<tr>
<td>cold_start_relative_from</td>
<td>Determines the starting point for the HRNN-Coldstart recipe to calculate cold_start_max_duration. To calculate from the current time, choose currentTime. To calculate cold_start_max_duration from the timestamp of the latest item in the Interactions dataset, choose latestItem. This setting is useful if you frequently add new items.</td>
</tr>
<tr>
<td></td>
<td>Default value: latestItem</td>
</tr>
<tr>
<td></td>
<td>Range: currentTime, latestItem</td>
</tr>
<tr>
<td></td>
<td>Value type: String</td>
</tr>
<tr>
<td></td>
<td>HPO tunable: No</td>
</tr>
</tbody>
</table>
### USER_PERSONALIZATION Recipes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| **min_user_history_length_percentile** | The minimum percentile of user history lengths to include in model training. History length is the total amount of data about a user. Use **min_user_history_length_percentile** to exclude a percentage of users with short history lengths. Users with a short history often show patterns based on item popularity instead of the user’s personal needs or wants. Removing them can train models with more focus on underlying patterns in your data. Choose an appropriate value after you review user history lengths, using a histogram or similar tool. We recommend setting a value that retains the majority of users, but removes the edge cases. For example, setting **min_user_history_length_percentile** to 0.05 and **max_user_history_length_percentile** to 0.95 includes all users except those with history lengths at the bottom or top 5%. Default value: 0.0  
Range: [0.0, 1.0]  
Value type: Float  
HPO tunable: No |
| **max_user_history_length_percentile** | The maximum percentile of user history lengths to include in model training. History length is the total amount of data about a user. Use **max_user_history_length_percentile** to exclude a percentage of users with long history lengths because data for these users tend to contain noise. For example, a robot might have a long list of automated interactions. Removing these users limits noise in training. Choose an appropriate value after you review user history lengths using a histogram or similar tool. We recommend setting a value that retains the majority of users but removes the edge cases. For example, setting **min_user_history_length_percentile** to 0.05 and **max_user_history_length_percentile** to 0.95 includes all users except those with history lengths at the bottom or top 5%. Default value: 0.99  
Range: [0.0, 1.0]  
Value type: Float  
HPO tunable: No |
HRNN-Coldstart Sample Notebook

For a sample Jupyter notebook that shows how to use the HRNN-Coldstart recipe, see Handling cold or new items with Amazon Personalize.

Using AutoML to Choose an HRNN Recipe (API Only)

Amazon Personalize can automatically choose the most appropriate hierarchical recurrent neural network (HRNN) recipe based on its analysis of the input data. This option is called AutoML. To perform AutoML, set the performAutoML parameter to true when you call the CreateSolution (p. 176) API.

You can also specify the list of recipes that Amazon Personalize examines to determine the optimal recipe, based on a metric you specify. In this case, you call the CreateSolution operation, specify true for the performAutoML parameter, omit the recipeArn parameter, and include the solutionConfig parameter, specifying the metricName and recipeList as part of the autoMLConfig object.

How a recipe is chosen is shown in the following table. Either performAutoML or recipeArn must be specified but not both. AutoML is only performed using the HRNN recipes.

<table>
<thead>
<tr>
<th>performAutoML</th>
<th>recipeArn</th>
<th>solutionConfig</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>omit</td>
<td>omitted</td>
<td>Amazon Personalize chooses the recipe</td>
</tr>
<tr>
<td>true</td>
<td>omit</td>
<td>autoMLConfig:metricName and recipeList specified</td>
<td>Amazon Personalize chooses a recipe from the list that optimizes the metric</td>
</tr>
<tr>
<td>omit</td>
<td>specified</td>
<td>omitted</td>
<td>You specify the recipe</td>
</tr>
<tr>
<td>omit</td>
<td>specified</td>
<td>specified</td>
<td>You specify the recipe and override the default training properties</td>
</tr>
</tbody>
</table>

Note
When performAutoML is true, all parameters of the solutionConfig object are ignored except for autoMLConfig.

PERSONALIZED_RANKING Recipes

The PERSONALIZED_RANKING recipe, Personalized-Ranking, provides recommendations in ranked order based on predicted interest level.

Personalized-Ranking (p. 85)

The Personalized-Ranking recipe is a hierarchical recurrent neural network (HRNN) recipe that also can filter and re-rank results. Personalized-Ranking provides a list of the best recommendations. Use the Personalized-Ranking recipe when you’re personalizing the results for your users, such as personalized re-ranking of search results or curated lists.

To train a model, the Personalized-Ranking recipe uses the Interactions dataset from a dataset group. A dataset group is a set of related datasets, which can include the Users, Items, and Interactions datasets.

Personalized-Ranking Recipe

The Personalized-Ranking recipe generates personalized rankings. A personalized ranking is a list of recommended items that are re-ranked for a specific user. To get a personalized ranking, use the GetPersonalizedRanking (p. 265) API.
This recipe has the following properties:

- **Name** – aws-personalized-ranking
- **Recipe Amazon Resource Name (ARN)** – arn:aws:personalize:::recipe/aws-personalized-ranking
- **Algorithm ARN** – arn:aws:personalize:::algorithm/aws-personalized-ranking
- **Feature transformation ARN** – arn:aws:personalize:::feature-transformation/JSON-percentile-filtering
- **Recipe type** – PERSONALIZED_RANKING

### Hyperparameters

The following table describes the hyperparameters for the Personalize-Ranking recipe. A hyperparameter is an algorithm parameter that you can adjust to improve model performance. Algorithm hyperparameters control how the model performs. Featurization hyperparameters control how to filter the data to use in training. The process of choosing the best value for a hyperparameter is called hyperparameter optimization (HPO). For more information, see Hyperparameters and HPO (p. 94).

The table also provides the following information for each hyperparameter:

- **Range**: [lower bound, upper bound]
- **Value type**: Integer, Continuous (float), Categorical (Boolean, list, string)
- **HPO tunable**: Can the parameter participate in hyperparameter optimization (HPO)?

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithm Hyperparameters</strong></td>
<td></td>
</tr>
<tr>
<td>hidden_dimension</td>
<td>The number of hidden variables used in the model. <em>Hidden variables</em> recreate users' purchase history and item statistics to generate ranking scores. Specify a greater number of hidden dimensions when your Interactions dataset includes more complicated patterns. Using more hidden dimensions requires a larger dataset and more time to process. To decide on the optimal value, use HPO. To use HPO, set <code>performHPO</code> to <code>true</code> when you call CreateSolution (p. 176) and CreateSolutionVersion (p. 181) operations. Default value: 149 Range: [32, 256] Value type: Integer HPO tunable: Yes</td>
</tr>
<tr>
<td>bptt</td>
<td>Determines whether to use the back-propagation through time technique. <em>Back-propagation through time</em> is a technique that updates weights in recurrent neural network-based algorithms. Use bptt for long-term credits to connect delayed rewards to early events. For example, a delayed reward can be a purchase made after several clicks. An early event can be an initial click. Even within the same event types, such as a click, it's a good</td>
</tr>
</tbody>
</table>
### PERSONALIZED_RANKING Recipes

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bptt</strong></td>
<td>Idea to consider long-term effects and maximize the total rewards. To consider long-term effects, use larger <code>bptt</code> values. Using a larger <code>bptt</code> value requires larger datasets and more time to process.</td>
</tr>
<tr>
<td><strong>recency_mask</strong></td>
<td>Determines whether the model should consider the latest popularity trends in the Interactions dataset. Latest popularity trends might include sudden changes in the underlying patterns of interaction events. To train a model that places more weight on recent events, set <code>recency_mask</code> to <code>true</code>. To train a model that equally weighs all past interactions, set <code>recency_mask</code> to <code>false</code>. To get good recommendations using an equal weight, you might need a larger training dataset.</td>
</tr>
</tbody>
</table>

### Featurization Hyperparameters
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_user_history_length_percentile</td>
<td>The minimum percentile of user history lengths to include in model training. History length is the total amount of data about a user. Use min_user_history_length_percentile to exclude a percentage of users with short history lengths. Users with a short history often show patterns based on item popularity instead of the user’s personal needs or wants. Removing them can train models with more focus on underlying patterns in your data. Choose an appropriate value after you review user history lengths, using a histogram or similar tool. We recommend setting a value that retains the majority of users, but removes the edge cases. For example, setting min_user_history_length_percentile to 0.05 and max_user_history_length_percentile to 0.95 includes all users except those with history lengths at the bottom or top 5%. Default value: 0.0 Range: [0.0, 1.0] Value type: Float HPO tunable: No</td>
</tr>
<tr>
<td>max_user_history_length_percentile</td>
<td>The maximum percentile of user history lengths to include in model training. History length is the total amount of data about a user. Use max_user_history_length_percentile to exclude a percentage of users with long history lengths because data for these users tend to contain noise. For example, a robot might have a long list of automated interactions. Removing these users limits noise in training. Choose an appropriate value after you review user history lengths using a histogram or similar tool. We recommend setting a value that retains the majority of users but removes the edge cases. For example, setting min_user_history_length_percentile to 0.05 and max_user_history_length_percentile to 0.95 includes all users except those with history lengths at the bottom or top 5%. Default value: 0.99 Range: [0.0, 1.0] Value type: Float HPO tunable: No</td>
</tr>
</tbody>
</table>
Personalized-Ranking Sample Notebook

For a sample Jupyter notebook that shows how to use the Personalized-Ranking recipe, see Personalize Ranking Example.

RELATED_ITEMS Recipes

The RELATED_ITEMS recipe, SIMS, returns items similar to a given item.

SIMS (p. 89)

The item-to-item similarities (SIMS) recipe generates items similar to a given item based on the co-occurrence of the item in user history in the user-item interaction dataset. If sufficient user behavior data for an item isn't available, or if the specified item ID isn't found, the recipe returns popular items as recommendations. Use the SIMS recipe to improve item discovery. Training is faster with the SIMS recipe compared to other recipes.

To train a model, the SIMS recipe uses the Interactions dataset from a dataset group. A dataset group is a set of related datasets, which can include the Users, Items, and Interactions datasets.

SIMS Recipe

The Item-to-item similarities (SIMS) recipe is based on the concept of collaborative filtering. A SIMS model leverages user-item interaction data to recommend items similar to a given item. In the absence of sufficient user behavior data for an item, this recipe recommends popular items.

This predefined recipe has the following properties:

- **Name** – aws-sims
- **Recipe Amazon Resource Name (ARN)** – arn:aws:personalize:::recipe/aws-sims
- **Algorithm ARN** – arn:aws:personalize:::algorithm/aws-sims
- **Feature transformation ARN** – arn:aws:personalize:::feature-transformation/sims
- **Recipe type** – RELATED_ITEMS

The following table describes the hyperparameters for the SIMS recipe. A hyperparameter is an algorithm parameter that you can adjust to improve model performance. Algorithm hyperparameters control how the model performs. Featurization hyperparameters control how to filter the data to use in training. The process of choosing the best value for a hyperparameter is called hyperparameter optimization (HPO). For more information, see Hyperparameters and HPO (p. 94).

The table also provides the following information for each hyperparameter:

- **Range**: [lower bound, upper bound]
- **Value type**: Integer, Continuous (float), Categorical (Boolean, list, string)
- **HPO tunable**: Can the parameter participate in hyperparameter optimization (HPO)?

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithm Hyperparameters</strong></td>
<td></td>
</tr>
<tr>
<td>popularity_discount_factor</td>
<td>Affects the balance between popularity and correlation when you calculate similarity. If you calculate similarities to a specific item, a value of 0 makes the most popular items appear as recommendations regardless of their</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>correlation</td>
<td>A value of 1 makes most items that have co-interactions (shared interaction) with the specific item appear as recommendations regardless of their popularity. Using either extreme might create an overly long list of recommend items. For most cases, a value around 0.5 works best.</td>
</tr>
<tr>
<td></td>
<td>Default value: 0.5</td>
</tr>
<tr>
<td></td>
<td>Range: [0.0, 1.0]</td>
</tr>
<tr>
<td></td>
<td>Value type: Float</td>
</tr>
<tr>
<td></td>
<td>HPO tunable: Yes</td>
</tr>
<tr>
<td>min_cointeraction_count</td>
<td>The minimum number of co-interactions you need to calculate the similarity between a pair of items. For example, a value of 3 means that you need three or more users who interacted with both items for the algorithm to calculate their similarity.</td>
</tr>
<tr>
<td></td>
<td>Default value: 3</td>
</tr>
<tr>
<td></td>
<td>Range: [0, 10]</td>
</tr>
<tr>
<td></td>
<td>Value type: Integer</td>
</tr>
<tr>
<td></td>
<td>HPO tunable: Yes</td>
</tr>
</tbody>
</table>

**Featurization Hyperparameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_user_history_length_percentile</td>
<td>The minimum percentile of user history lengths to include in model training. <em>History length</em> is the total amount of available data on a user. Use min_user_history_length_percentile to exclude a percentage of users with short history lengths. Users with a short history often show patterns based on item popularity instead of the user's personal needs or wants. Removing them can train models with more focus on underlying patterns in your data. Choose an appropriate value after you review user history lengths, using a histogram or similar tool. We recommend setting a value that retains the majority of users, but removes the edge cases.</td>
</tr>
<tr>
<td></td>
<td>Default value: 0.005</td>
</tr>
<tr>
<td></td>
<td>Range: [0.0, 1.0]</td>
</tr>
<tr>
<td></td>
<td>Value type: Float</td>
</tr>
<tr>
<td></td>
<td>HPO tunable: No</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>max_user_history_length_percentile</strong></td>
<td>The maximum percentile of user history lengths to include in model training. History length is the total amount of available data on a user. Use max_user_history_length_percentile to exclude a percentage of users with long history lengths. Users with a long history tend to contain noise. For example, a robot might have a long list of automated interactions. Removing these users limits noise in training. Choose an appropriate value after you review user history lengths using a histogram or similar tool. We recommend setting a value that retains the majority of users but removes the edge cases. For example, min_hist_length_percentile = 0.05 and max_hist_length_percentile = 0.95 includes all users except ones with history lengths at the bottom or top 5%. Default value: 0.995 Range: [0.0, 1.0] Value type: Float HPO tunable: No</td>
</tr>
<tr>
<td><strong>min_item_interaction_count_percentile</strong></td>
<td>The minimum percentile of item interaction counts to include in model training. Use min_item_interaction_count_percentile to exclude a percentage of items with a short history of interactions. Items with a short history often are new items. Removing them can train models with more focus on items with a known history. Choose an appropriate value after you review user history lengths, using a histogram or similar tool. We recommend setting a value that retains the majority of items, but removes the edge cases. Default value: 0.01 Range: [0.0, 1.0] Value type: Float HPO tunable: No</td>
</tr>
</tbody>
</table>
Step 2: Configuring a Solution

Once you complete Step 1: Choosing a Recipe (p. 64), you are ready to configure a solution for training a model.

Configuring a solution allows you to customize training so the model meets your specific business needs. To configure a solution, you specify the dataset group with the data to be used for training, the recipe to be used for training, and any additional solution parameters and recipe-specific hyperparameters. If your Interactions training data includes EVENT_TYPE and EVENT_VALUE data, when you configure a solution you may filter out Interactions data before training.

You can create and configure a solution using the console, AWS Command Line Interface (AWS CLI), or AWS SDK.

Topics

- Configuring a Solution (Console) (p. 93)
- Configuring a Solution (AWS CLI) (p. 94)
- Configuring a Solution (AWS Python SDK) (p. 94)
- Hyperparameters and HPO (p. 94)
- Filtering Interactions Data Before Training (p. 96)
Configuring a Solution (Console)

To configure a solution in the console, choose the dataset group containing the dataset you'll be using, and then specify a solution name, recipe, and optional recipe specific hyperparameters.

To configure a solution using the console

1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/ and sign in to your account.
2. Choose the dataset group you want to use for training.
3. On the dashboard, in the Create solutions section, choose the **Start** button.
   
   If you have already created a solution, choose the **Create solution** button.
4. For **Solution name**, specify a name for your solution.
5. For **Recipe**, choose a recipe (see Step 1: Choosing a Recipe (p. 64)).
6. Expand the **Solution configuration** section and optionally configure any hyperparameter options based on your recipe and business needs. Different recipes use different hyperparameters. For the available hyperparameters, see the individual recipes in Step 1: Choosing a Recipe (p. 64).
   
   An example of the Solution configuration options for the **aws-user-personalization** recipe is below.

   ![Solution configuration](image)

Configuring a Solution (AWS CLI)

To configure a solution using AWS CLI use the following create-solution operation. Specify the solution name, dataset group arn, and recipe arn using the following code.

```bash
aws personalize create-solution \
--name solution name \
--dataset-group-arn dataset group arn \
--recipe-arn recipe arn
```

The solution Amazon Resource Name (ARN) is displayed, for example:

```json
{
 "solutionArn": "arn:aws:personalize:<region>:solution/<solution name>"
}
```

You can modify the above code to optimize recipe properties and hyperparameters (see Hyperparameters and HPO (p. 94)) or filter the Interactions data used for training (see Filtering Interactions Data Before Training (p. 96)).

Record the solution ARN for future use and proceed to Creating a Solution Version (AWS CLI) (p. 100).

Configuring a Solution (AWS Python SDK)

Create a new solution using the create_solution method. Replace solution name with your solution name, recipe arn with the recipe Amazon Resource Name (ARN) from Step 1: Choosing a Recipe (p. 64), and dataset group arn with the ARN of your dataset group.

```python
import boto3

personalize = boto3.client('personalize')

print('Creating solution')
create_solution_response = personalize.create_solution(
    name='solution name',
    recipeArn='recipe arn',
    datasetGroupArn='dataset group arn'
)

solution_arn = create_solution_response['solutionArn']
print('solution_arn: ', solution_arn)
```

You can modify the above code to optimize recipe properties and hyperparameters (see Hyperparameters and HPO (p. 94)) or filter the Interactions data used for training (see Filtering Interactions Data Before Training (p. 96)).

Record the solution ARN for future use and proceed to Creating a Solution Version (AWS Python SDK) (p. 101).

Hyperparameters and HPO

Hyperparameters are used to optimize the trained model and are set before training begins. This contrasts with model parameters whose values are determined during the training process.

Hyperparameters are specified using the algorithmHyperParameters key that is part of the SolutionConfig (p. 340) object that is passed to the CreateSolution (p. 176) operation.
A condensed version of the CreateSolution request is below. The example includes the solutionConfig object. You use solutionConfig to override the default parameters of a recipe. When performAutoML is true, all parameters of the solutionConfig object are ignored except for autoMLConfig.

```json
{
    "name": "string",
    "performAutoML": boolean,
    "recipeArn": "string",
    "performHPO": boolean,
    "eventType": "string",
    "solutionConfig": {
        "autoMLConfig": {
            "metricName": "string",
            "recipeList": [ "string" ]
        },
        "eventValueThreshold": "string",
        "featureTransformationParameters": {
            "string": "string"
        },
        "algorithmHyperParameters": {
            "string": "string"
        },
        "hpoConfig": {
            "algorithmHyperParameterRanges": {
                "hidden_dimension": "55"
            },
            "hpoResourceConfig": {
                "maxNumberOfTrainingJobs": "string",
                "maxParallelTrainingJobs": "string"
            }
        }
    }
}
```

Different recipes use different hyperparameters. For the available hyperparameters, see the individual recipes in Step 1: Choosing a Recipe (p. 64).

Hyperparameter optimization (HPO), or tuning, is the task of choosing optimal hyperparameters for a specific learning objective. The optimal hyperparameters are determined by running many training jobs using different values from the specified ranges of possibilities. By default, Amazon Personalize does not perform HPO. To use HPO, set performHPO to true, and include the hpoConfig object.

Hyperparameters can be categorical, continuous, or integer-valued. The hpoConfig object has keys that correspond to each of these types, where you specify the hyperparameters and their ranges. Note that not all hyperparameters can be tuned (see the recipe tables).

The following is a partial example of a CreateSolution request using the HRNN (p. 75) recipe.

```json
{
    "performAutoML": false,
    "recipeArn": "arn:aws:personalize:::recipe/aws-hrnn",
    "performHPO": true,
    "solutionConfig": {
        "algorithmHyperParameters": {
            "hidden_dimension": "55"
        },
        "hpoConfig": {
            "algorithmHyperParameterRanges": {
                "categoricalHyperParameterRanges": [
                    {
                        "name": "recency_mask",
                        "values": [ "true", "false" ]
                    }
                ]
            }
        }
    }
}
```
Once training is complete, you can view the hyperparameters of the best performing model by calling the section called “DescribeSolutionVersion” (p. 223) operation. The following sample shows a condensed DescribeSolutionVersion output with the optimized hyperparameters displayed in the tunedHPOParams object.

```json
{
    "solutionVersion":{
        "creationDateTime":1562191944.745,
        "lastUpdatedDateTime":1562194465.075,
        "performAutoML":false,
        "performHPO":true,
        "recipeArn":"arn:aws:personalize:::recipe/aws-hrn",
        "status":"ACTIVE",
        "tunedHPOParams":{
            "algorithmHyperParameters":{
                "hidden_dimension":"58",
                "recency_mask":false
            }
        }
    }
}
```

For more information, see Automatic Model Tuning.

## Filtering Interactions Data Before Training

If your Interactions dataset includes data that you don't want to use for training, you can filter out those records by setting a threshold for a value, such as price, or by specifying a type of event, such as purchase or click. By filtering, you can train using only a relevant subset of your data or remove noise to train a more optimized model.

You can filter records from an Interactions dataset two ways:

- **Set a threshold to exclude records based on a specific value by specifying an event value in your recipe** - If the records include an amount that is associated with a specific event—for example, the price a user paid is associated with the purchase of an item—you can set a specific value in a recipe as a threshold to exclude records from training. The amount is called an *event value*.

- **Exclude records of a certain type by specifying an event type in your recipe** – A dataset often includes specific types of activities, for example, “purchase”, “click”, or "wishlisted". These
are called event types. To include only records for specific event types in training, filter your dataset by event type in your recipe.

Likewise, an Interactions dataset often includes specific types of activities, for example, “purchase”, “click”, or “wishlisted”. These are called event types. To include only records for specific event types in training, filter your dataset by event type in your recipe.

To filter by event value or event type, you create an Interactions schema for the recipe that you use to create your solution. To use a more specific subset in training, you can also filter a dataset by an event value and event type.

**Filtering Records by Event Value and Event Type**

In the following procedure, you use the AWS SDK for Python (Boto3) to create an Interaction schema that filters a training dataset. You can use a Jupyter (iPython) notebook to accomplish the same task. For more information, see Getting Started Using Amazon Personalize APIs with Jupyter (iPython) Notebooks (p. 30).

**Prerequisites:** Complete the prerequisites and verify that your Python environment is set up as described in Getting Started (AWS SDK for Python) (p. 29).

**To filter records used in a training dataset by event value or event type**

1. Create an Interaction schema and include the EVENT_TYPE and EVENT_VALUE fields using "name" and "type" key-value pairs as shown.

   ```python
   import boto3
   import json

   personalize = boto3.client('personalize')

   # Create a name for your schema
   schema_name = 'YourSchemaName'

   # Define the schema for your dataset
   schema = {
       "type": "record",
       "name": "Interactions",
       "namespace": "com.amazonaws.personalize.schema",
       "fields": [
           {
               "name": "USER_ID",
               "type": "string"
           },
           {
               "name": "ITEM_ID",
               "type": "string"
           },
           {
               "name": "EVENT_VALUE",
               "type": "float"
           },
           {
               "name": "EVENT_TYPE",
               "type": "string"
           },
           {
               "name": "TIMESTAMP",
               "type": "long"
           }
   }
   ```
# Create the schema for Amazon Personalize
create_schema_response = personalize.create_schema(
    name = schema_name,
    schema = json.dumps(schema)
)

#To get the schema ARN, use the following lines
schema_arn = create_schema_response['schemaArn']
print('Schema ARN: ' + schema_arn)

2. Format your input data to match your schema. For a code sample, see Formatting Your Input Data (p. 46).

3. Upload your data to an Amazon Simple Storage Service (Amazon S3) bucket. For a code sample, see Uploading to an Amazon S3 Bucket (p. 48).

4. Import your data into Amazon Personalize with the CreateDatasetImportJob (p. 166) API. Be sure to record your dataset group Amazon Resource Name (ARN) because you will need it when you create the solution. For a code sample, see Importing Bulk Records (AWS Python SDK) (p. 53).

5. Get the ARN of the recipe that you want to use when you create your solution. You’ll need it when you create the solution.

```python
import boto3
personalize = boto3.client('personalize')

# Display the ARNs of the recipes
recipe_list = personalize.list_recipes()
for recipe in recipe_list['recipes']:
    print(recipe['recipeArn'])

# Store the ARN of the recipe that you want to use
recipe_arn = "arn:aws:personalize:::recipe/aws-recipe-name"
```

6. Call the CreateSolution (p. 176) API. If you want to specify the event type, for example “purchase”, set it in the eventType parameter. If you want to specify an event value, for example 10, set it in the eventValueThreshold parameter. You can also specify both an event type and an event value.

```python
import boto3
personalize = boto3.client('personalize')

# Create the solution
create_solution_response = personalize.create_solution(
    name = "your-solution-name",
    datasetGroupArn = dataset_group_arn,
    recipeArn = recipe_arn,
    eventType = 'watched',
    solutionConfig = {
        "eventValueThreshold": "0.5"
    }
)

# Store the solution ARN
solution_arn = create_solution_response['solutionArn']

# Use the solution ARN to get the solution status
solution_description = personalize.describe_solution(solutionArn = solution_arn) ['solution']
```
Step 3: Creating a Solution Version

Once you have completed Step 1: Choosing a Recipe (p. 64) and Step 2: Configuring a Solution (p. 92), you are ready to create a Solution Version.

A Solution Version refers to a trained machine learning model you can deploy to get recommendations for customers. You can create a solution version using the console, AWS Command Line Interface (AWS CLI), or AWS SDK.

Topics
- Creating a Solution Version (Console) (p. 99)
- Creating a Solution Version (AWS CLI) (p. 100)
- Creating a Solution Version (AWS Python SDK) (p. 101)

Creating a Solution Version (Console)

If you just completed Step 2: Configuring a Solution (p. 92) and the Create solution version is displayed, choose FINISH to create a solution version.

On the solution details page, you can track training progress in the Solution versions section. When training is complete, the status is Active and you are ready to deploy a campaign and get recommendations. See step 3 in the Getting Started (Console) (p. 13) tutorial.

If you navigated away from the Create solution version page or want to create an additional solution version for an existing solution, create a new solution version from the solution overview page as follows.
Create a new solution version

1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/ and sign into your account.
2. Navigate to the dataset groups page and choose the dataset group with your new solution.
3. In the navigation pane, choose Solutions and recipes.
4. Choose the solution. The solution overview page displays.
5. Choose Create solution version to start training a new model.

On the solution details page, you can track training progress in the Solution versions section. When training is complete, the status is Active and you are ready to deploy a campaign and get recommendations. See step 3 in the Getting Started (Console) (p. 13) tutorial.

Creating a Solution Version (AWS CLI)

When your solution is ACTIVE, train the model by running the following command. Replace solution arn with the solution Amazon Resource Name (ARN) from Step 2: Configuring a Solution (p. 92).

```
aws personalize create-solution-version \
  --solution-arn solution arn
```

The solution version ARN is displayed, for example:

```
{
}
```

Check the training status of the solution version by using the describe-solution-version command. Provide the solution version ARN that was returned in the previous step. For more information about the API, see DescribeSolutionVersion (p. 223).

```
aws personalize describe-solution-version \
  --solution-version-arn solution version arn
```

The properties of the solution version and the training status are displayed. Initially, the status shows as CREATE PENDING, for example:

```
{
  "solutionVersion": {
    ...
    "status": "CREATE PENDING"
  }
}
```

Training is complete when the status is ACTIVE.

Now that you have created a solution version, evaluate it using metrics supplied by Amazon Personalize. For more information, see Step 4: Evaluating the Solution Version (p. 101).
Creating a Solution Version (AWS Python SDK)

When your solution is ACTIVE, Create a solution version using the `create_solution` method. Replace the `solution_arn` with the Amazon Resource Name (ARN) of the solution from Step 2: Configuring a Solution (p. 92).

```python
import boto3

personalize = boto3.client('personalize')

# Store the solution ARN
solution_arn = 'solution arn'

# Use the solution ARN to get the solution status.
solution_description = personalize.describe_solution(solutionArn = 'solution_arn')['solution']
print('Solution status: ' + solution_description['status'])

# Use the solution ARN to create a solution version.
print ('Creating solution version')
response = personalize.create_solution_version(solutionArn = solution_arn)
solution_version_arn = response['solutionVersionArn']
print('Solution version ARN: ' + solution_version_arn)

# Use the solution version ARN to get the solution version status.
solution_version_description = personalize.describe_solution_version(solutionVersionArn = solution_version_arn)['solutionVersion']
print('Solution version status: ' + solution_version_description['status'])
```

To check the current solution version status, call the `DescribeSolutionVersion` (p. 223) operation and pass the ARN of the solution version returned from the `CreateSolutionVersion` operation. Training is complete when the status is ACTIVE.

Now that you have a created solution version, evaluate it using metrics supplied by Amazon Personalize. For more information, see Step 4: Evaluating the Solution Version (p. 101).

Step 4: Evaluating the Solution Version

When you create a solution version, Amazon Personalize generates metrics that you can use to evaluate the performance of the model before you create a campaign and provide recommendations. Metrics allow you to view the effects of modifying a solution's hyperparameters. You can also use metrics to compare the results between solutions that use the same training data but were created with different recipes.

To get performance metrics, Amazon Personalize splits the input interactions data by randomly selecting 90% of users and their related interactions as training data and the other 10% as testing data. The solution version is then created using the training data. Afterwards, the solution version is given the oldest 90% of each user's testing data as input, and the recommendations it generates are compared against the real interactions given by the most recent 10% of testing data.

To generate a baseline for comparison purposes, we recommend using the Popularity-Count (p. 74) recipe, which recommends the top K most popular items.

**Important**

In order for Amazon Personalize to generate solution version metrics, you must have at least 10 datapoints in your input dataset group.
Metrics

You retrieve the metrics for a specific solution version by calling the GetSolutionMetrics (p. 226) operation.

Retrieve metrics using the AWS Python SDK

1. Create a solution version. For more information, see Creating a Solution (p. 64).
2. Use the following code to retrieve metrics.

```python
import boto3

personalize = boto3.client('personalize')

response = personalize.get_solution_metrics(
    solutionVersionArn = 'solution version arn')

print(response['metrics'])
```

The following is an example of the output from a solution version created using the HRNN (p. 75) recipe with the default solution configuration.

```json
{
    "metrics": {
        "coverage": 0.27,
        "mean_recall_rank_at_25": 0.0379,
        "normalized_discounted_cumulative_gain_at_5": 0.0405,
        "normalized_discounted_cumulative_gain_at_10": 0.0513,
        "normalized_discounted_cumulative_gain_at_25": 0.0828,
        "precision_at_5": 0.0136,
        "precision_at_10": 0.0102,
        "precision_at_25": 0.0091
    }
}
```

The above metrics are described below using the following terms:

- **Relevant recommendation** refers to a recommendation that matches a value in the testing data for the particular user.
- **Rank** refers to the position of a recommended item in the list of recommendations. Position 1 (the top of the list) is presumed to be the most relevant to the user.
- **Query** refers to the internal equivalent of a GetRecommendations (p. 269) call.

For each metric, higher numbers are better.

**coverage**

The proportion of unique recommended items from all queries out of the total number of unique items in the interactions and items datasets.

**mean_recall_rank_at_25**

The mean of the reciprocal ranks of the first relevant recommendation out of the top 25 recommendations over all queries.

This metric is appropriate if you're interested in the single highest ranked recommendation.
normalized_discounted_cumulative_gain_at_K

Discounted gain assumes that recommendations lower on a list of recommendations are less relevant than higher recommendations. Therefore, each recommendation is discounted (given a lower weight) by a factor dependent on its position. To produce the cumulative discounted gain (DCG) at K, each relevant discounted recommendation in the top K recommendations is summed together. The normalized discounted cumulative gain (NDCG) is the DCG divided by the ideal DCG such that NDCG is between 0 - 1. (The ideal DCG is where the top K recommendations are sorted by relevance.)

Amazon Personalize uses a weighting factor of $1/\log(1 + \text{position})$, where the top of the list is position 1.

This metric rewards relevant items that appear near the top of the list, because the top of a list usually draws more attention.

precision_at_K

The number of relevant recommendations out of the top K recommendations divided by K.

This metric rewards precise recommendation of the relevant items.

Example

The following is a simple example where, to generate metrics, a solution version produces a list of recommendations for a specific user. The second and fifth recommendations match records in the testing data for this user. These are the relevant recommendations. If K is set at 5, the following metrics are generated for the user.

reciprocal_rank

Calculation: 1/2

Result: 0.5000

normalized_discounted_cumulative_gain_at_5

Calculation: $(1/\log(1 + 2) + 1/\log(1 + 5)) / (1/\log(1 + 1) + 1/\log(1 + 2))$

Result: 0.6241

precision_at_5

Calculation: 2/5

Result: 0.4000

Now that you have evaluated your solution version, create a campaign by deploying the optimum solution version. For more information, see Creating a Campaign (p. 104).

More Info

For a sample Jupyter notebook that shows you how to retrieve metrics based on hold-out data, see Personalize with temporal evaluation on hold-out set.
Amazon Personalize Developer Guide

Creating a Campaign
A campaign is used to make recommendations for your users. You create a campaign by deploying
a solution version. For an example using the AWS CLI, see Step 3: Create a Campaign (Deploy the
Solution) (p. 27).
To create a campaign with the SDK, call the CreateCampaign (p. 157) API and pass the following:
• A name for the campaign.
• The Amazon Resource Name (ARN) of the solution version to deploy.
• The minimum provisioned transactions per second (minProvisionedTPS) that Amazon Personalize
will support. For more information on minProvisionedTPS, see the CreateCampaign API.

Create a campaign using the AWS Python SDK
1.

Create a solution version to deploy. For more information, see Creating a Solution (p. 64).

2.

Use the following code to create a campaign for a solution version trained using the
User-Personalization recipe with the optional default explorationWeight and
explorationItemAgeCutOff for itemExplorationConfig. For more information, see UserPersonalization (p. 66)
import boto3
personalize = boto3.client('personalize')
create_campaign_response = personalize.create_campaign(
name = 'campaign name',
solutionVersionArn = 'solution version arn',
minProvisionedTPS = 1,
campaignConfig = {"itemExplorationConfig": {"explorationWeight": "0.3",
"explorationItemAgeCutOff": "30"}}
)
arn = response['campaignArn']
description = personalize.describe_campaign(campaignArn = arn)['campaign']
print('Name: ' + description['name'])
print('ARN: ' + description['campaignArn'])
print('Status: ' + description['status'])

The campaign isn't ready for use until its status is active. To get the current status, call
DescribeCampaign (p. 202) and check that the status ﬁeld is ACTIVE.
Amazon Personalize provides operations for managing campaigns such as ListCampaigns (p. 231) to list
the campaigns you have created. You can delete a campaign by calling DeleteCampaign (p. 184). If you
delete a campaign, the solution versions that are part of the campaign are not deleted.
After you have created your campaign, use it to make recommendations. For more information, see
Getting Recommendations (p. 105).

104


Getting Recommendations

After you have created a campaign, you can use it in your applications to get recommendations. The following topics explain how and when to use each recommendation type.

Topics
- How Recommendation Scoring Works (p. 105)
- Getting Real-Time Recommendations (p. 105)
- Getting Batch Recommendations (p. 108)
- Filtering Recommendations (p. 113)

How Recommendation Scoring Works

To make recommendations, Amazon Personalize generates scores for the items in your Items dataset based on a user's interaction data and metadata. These scores represent the relative certainty that Amazon Personalize has in which item the user will select next. Higher scores represent greater certainty.

The formulas that calculate scores depend on the recommendation use case and the recipe that was used to train the model. You can view item scores in the Amazon Personalize console or by using the Amazon Personalize Runtime APIs. For more information on how scores are calculated and what they mean, see Getting Real-Time Recommendations (p. 105) and Getting Batch Recommendations (p. 108).

Getting Real-Time Recommendations

You can get real-time recommendations from Amazon Personalize with a campaign. For example, suppose you have a campaign that is designed to give movie recommendations. You can use the Getting Real-Time Recommendations (API) (p. 106) or Get Personalized Ranking (API) (p. 107) to give real-time movie recommendations to users signed into your application or website. For an example using the AWS CLI, see Step 4: Get Recommendations (p. 28).

Increasing Recommendation Relevance with Contextual Metadata

To increase recommendation relevance, include contextual metadata for a user, such as their device type or the time of day, when using the Getting Real-Time Recommendations (API) (p. 106) or Get Personalized Ranking (API) (p. 107).

To use contextual metadata, you must meet the following requirements:

- Any context metadata fields must be included in the schema of the campaign's Iteractions dataset (see Datasets and Schemas (p. 32)).
- The solution backing the campaign must have been created using a recipe of type USER_PERSONALIZATION or RELATED_ITEMS (see Step 1: Choosing a Recipe (p. 64)).

For more information about the benefits of including contextual metadata, see Contextual Metadata (p. 35).

For examples that show how to include contextual metadata using the Getting Real-Time Recommendations (API) (p. 106) or Get Personalized Ranking (API) (p. 107), see Getting
Getting Real-Time Recommendations (API)

To get recommendations, call the `GetRecommendations` API. Supply either the user ID or item ID, dependent on the recipe type used to create the solution the campaign is based on.

How Scoring Works

To make recommendations, Amazon Personalize generates scores for the items in your Items dataset based on a user’s interaction data and metadata. These scores represent the relative certainty that Amazon Personalize has in which item the user will select next. Higher scores represent greater certainty.

Models that are based on USER_PERSONALIZATION recipes score all of the items in your Items dataset relative to each other on a scale from 0 to 1 (both inclusive), so that the total of all scores equals 1. For example, if you’re getting movie recommendations for a user and there are three movies in the Items dataset, their scores might be 0.6, 0.3, and 0.1. Similarly, if you have 1,000 movies in your inventory, the highest-scoring movies might have very small scores (the average score would be 0.001), but, because scoring is relative, the recommendations are still valid.

In mathematical terms, scores for each user-item pair \((u,i)\) are computed according to the following formula, where “exp” is the exponential function, \(\bar{w}_u\) and \(\bar{w}_i\) are user and item embeddings respectively, and the Greek letter sigma (\(\Sigma\)) represents summation over all items in the item dataset:

\[
\text{score}(u,i) = \frac{\exp(\bar{w}_u^\top \bar{w}_i)}{\sum_j \exp(\bar{w}_u^\top \bar{w}_j)}
\]

**Note**

Scores aren’t shown for SIMS or Popularity-Count-based models.

Getting Recommendations (AWS Python SDK)

Use the following code to get a recommendation. Change the value of `userId` to a user ID that is in the data that you used to train the solution. A list of recommended items for the user is displayed.

```python
import boto3

personalizeRt = boto3.client('personalize-runtime')

response = personalizeRt.get_recommendations(
    campaignArn = 'Campaign ARN',
    userId = 'User ID')

print("Recommended items")
for item in response['itemList']:
    print (item['itemId'])
```

Getting Recommendations using Contextual Metadata (AWS Python SDK)

Use the following code to get a recommendation based on contextual metadata. For context, for each key-value pair, provide the metadata field as the key and the context data as the value. In the following sample code, the key is `DEVICE` and the value is `mobile phone`. Replace these values and the Campaign ARN and User ID with your own. A list of recommended items for the user is displayed.
Get Personalized Ranking (API)

A personalized ranking is a list of recommended items that are re-ranked for a specific user. To get personalized rankings, call the `GetPersonalizedRanking` API.

**Note**
The solution backing the campaign must have been created using a recipe of type `PERSONALIZED_RANKING`. For more information, see Step 1: Choosing a Recipe (p. 64).

**How Scoring Works**

Like the scores returned by the `GetRecommendations` operation, `GetPersonalizedRanking` scores sum to 1, but because the list of considered items is much smaller than your full Items dataset, recommendation scores tend to be higher.

Mathematically, the scoring function for `GetPersonalizedRanking` is identical to `GetRecommendations`, except that it only considers the input items. This means that scores closer to 1 become more likely, as there are fewer other choices to divide up the score:

$$
\text{score}(u, i) = \frac{\exp(w_u^T w_i)}{\sum_{j \in \text{input}} \exp(w_u^T w_j)}
$$

**Getting Personalized Rankings (AWS Python SDK)**

Use the following code to get a personalized ranking. Change the value of `userId` and `inputList` to a user ID and list of item IDs that are in the data that you used to train the solution. A list of ranked recommendations is displayed. Amazon Personalize considers the first item in the list of most interest to the user.

```python
import boto3

personalizeRt = boto3.client('personalize-runtime')

response = personalizeRt.get_personalized_ranking(
    campaignArn = "Campaign arn",
    userId = "UserID",
    inputList = ['ItemID1', 'ItemID2'])

print("Personalized Ranking")
for item in response['personalizedRanking']:
    print (item['itemId'])
```
Getting Personalized Rankings using Contextual Metadata (AWS Python SDK)

Use the following code to get a personalized ranking based on contextual metadata. For context, for each key-value pair, provide the metadata field as the key and the context data as the value. In the following sample code, the key is `DEVICE` and the value is `mobile phone`. Replace these values and the Campaign ARN and User ID with your own. Also change inputList to a list of item IDs that are in the data that you used to train the solution. Amazon Personalize considers the first item in the list of most interest to the user.

```python
import boto3

personalizeRt = boto3.client('personalize-runtime')

response = personalizeRt.get_personalized_ranking(
    campaignArn = "Campaign ARN",
    userId = "User ID",
    inputList = ['ItemID1', 'ItemID2'],
    context = {
        'DEVICE': 'mobile phone'
    }
)

print("Personalized Ranking")
for item in response['personalizedRanking']:
    print(item['itemId'])
```

Personalized-Ranking Sample Notebook

For a sample Jupyter notebook that shows how to use the Personalized-Ranking recipe see Personalize Ranking Example.

Getting Batch Recommendations

Use an asynchronous batch workflow to get recommendations from large datasets that do not require real-time updates. For instance, you might create a batch inference job to get product recommendations for all users on an email list, or to get item-to-item similarities (SIMS) (p. 89) across an inventory. To get batch recommendations, you can create a batch inference job by calling `CreateBatchInferenceJob` (p. 153).

In order to get batch recommendations, the IAM user role that invokes the `CreateBatchInferenceJob` (p. 153) operation must have read and write permissions to your input and output Amazon S3 buckets respectively. For more information on bucket permissions, see User Policy Examples in the Amazon Simple Storage Service (S3) Developer Guide.

**Note**

Amazon S3 buckets and objects must be either encryption free or, if you are using AWS Key Management Service (AWS KMS) for encryption, you must give your IAM user and Amazon Personalize IAM service role permission to use your key. For more information see Using key policies in AWS KMS in the AWS Key Management Service Developer Guide.

You can perform batch inference operations with any of the following tools:

- Amazon Personalize console (p. 110)
- AWS CLI (p. 111)
• AWS SDKs (p. 112)

How scoring works

Item scores calculated by batch recommendation jobs are calculated the same ways as described in Getting Real-Time Recommendations (p. 105), and can be viewed in the batch job's output JSON file. Scores are only returned by models trained with the HRNN and Personalize-Ranking recipes.

Input and Output JSON Examples

The CreateBatchInferenceJob (p. 153) uses a solution version to make recommendations based on data provided in an input JSON file. The result is then returned as a JSON file to an Amazon S3 bucket. The following tab list contains correctly formatted JSON input and output examples for each recipe type.

USER_PERSONALIZATION

Input

```json
{"userId": "4638"}
{"userId": "663"}
{"userId": "3384"}...
```

Output

```json
{"input":{"userId":"4638"}, "output": {"recommendedItems": ["296", "1", "260", "318"]},
{"scores": [0.0009785, 0.000976, 0.0008851]}}
{"input":{"userId":"663"}, "output": {"recommendedItems": ["1393", "3793", "2701", "3826"]},
{"scores": [0.00008149, 0.00007025, 0.000652]}}
{"input":{"userId":"3384"}, "output": {"recommendedItems": ["8368", "5989", "40815", "48780"]},
{"scores": [0.003015, 0.00154, 0.00142]}}...
```

Popularity-Count

Input

```json
{}
{"itemId": "105"}
{"itemId": "41"}...
```

Output

```json
{"input": {}, "output": {"recommendedItems": ["105", "106", "441"]}}
{"input":{"itemId":"105"}, "output": {"recommendedItems": ["105", "106", "441"]}}
{"input":{"itemId":"41"}, "output": {"recommendedItems": ["105", "106", "441"]}}...
```

Personalize-Ranking

Input

```json
{"userId": "891", "itemList": ["27", "886", "101"]}
```
Getting Batch Recommendations (Amazon Personalize Console)

The following procedure outlines the batch inference workflow using the Amazon Personalize console. This procedure assumes that you have already created a solution that is properly formatted to perform the desired batch job on your dataset.

1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/home and sign in to your account.
2. Choose Batch inference jobs in the navigation pane, then choose Create batch inference job.
3. In Batch inference job details, in Batch inference job name, specify a name for your batch inference job.
4. For IAM service role, choose the Amazon Personalize IAM service role that has read and write access to your input and output Amazon S3 buckets respectively.
5. For Solution, choose the solution that you want to use to generate the recommendations. The solution recipe must match the input data's format.
6. In Input data configuration, specify the Amazon S3 path to your input file. In Output data configuration, specify the path to your output Amazon S3 bucket.

Your screen should look similar to the following:
Note
Creating a batch inference job takes time.

8. When the batch inference job’s status changes to **Active**, you can retrieve the job’s output from the designated output Amazon S3 bucket. The output file’s name will be of the format `input-name.out`.

### Getting Batch Recommendations (AWS CLI)

The following is an example of a batch inference workflow using the AWS CLI for a solution trained using the USER_PERSONALIZATION recipe. A JSON file called `batch.json` is passed as input, and the output file, `batch.json.out`, is returned to an Amazon S3 bucket.

For `batch-inference-job-config`, the example includes USER_PERSONALIZATION recipe specific `itemExplorationConfig` hyperparameters: `explorationWeight` and `explorationItemAgeCutOff`. Optionally include `explorationWeight` and `explorationItemAgeCutOff` values to configure exploration. For more information, see User-Personalization Recipe (p. 66).

```bash
aws personalize create-batch-inference-job --job-name batchTest \
    --job-input s3DataSource={path=s3://personalize/batch/input/input.json} \
    --job-output s3DataDestination={path=s3://personalize/batch/output/} \
    --role-arn arn:aws:iam::012345678901:role/import-export-role \
    --batch-inference-job-config itemExplorationConfig={explorationWeight=0.3,
    explorationItemAgeCutOff=30}
```

Once a batch inference job is created, you can inspect it further with the **DescribeBatchInferenceJob** (p. 200) operation.

```bash
aws personalize describe-batch-inference-job --batch-inference-job-arn
```
Getting Batch Recommendations (AWS Python SDK)

Use the following code to get batch recommendations using the AWS Python SDK. The example includes itemExplorationConfig hyperparameters for solution versions trained using the USER_PERSONALIZATION recommendation recipe. Optionally include the itemExplorationConfig hyperparameters to configure exploration. For more information see User-Personalization Recipe (p. 66).

The operation reads an input JSON file from an Amazon S3 bucket and places an output JSON file (input-file-name.out) in an Amazon S3 bucket.

The first item in the response file is considered by Amazon Personalize to be of most interest to the user.

```
import boto3

personalize_rec = boto3.client(service_name='personalize')

personalize_rec.create_batch_inference_job(
    solutionVersionArn = "Solution version ARN",
    jobName = "Batch job name",
    roleArn = "IAM role ARN",
    batchInferenceJobConfig = {
        "itemExplorationConfig": {
            "explorationWeight": "0.3",
            "explorationItemAgeCutOff": "30"
        }
    },
    jobInput = {
        "s3DataSource": {"path": "S3 input path"},
        "jobInput": {
            "s3DataDestination": {"path": "S3 output path"}
        }
    })
```

The command returns the ARN for the batch job (the batchRecommendationsJobArn).

Processing the batch job might take a while to complete. You can check a job's status by calling DescribeBatchInferenceJob (p. 200) and passing a batchRecommendationsJobArn as the input parameter. You can also list all Amazon Personalize batch inference jobs in your AWS environment by calling ListBatchInferenceJobs (p. 228).
Filtering Recommendations

When getting recommendations with Amazon Personalize, you can filter results based on custom criteria. For example, you might not want to recommend products that a user has already purchased, or recommend movies that a user has already watched. By filtering your recommendations, you can control the items that will be recommended to users.

When filtering items, Amazon Personalize recognizes event types that were present during solution version training or that were streamed using the PutEvents (p. 259) operation. For item and user data imported incrementally, Amazon Personalize updates any filters you created in the dataset group with your new item and user data within 20 minutes from the last incremental import. For more information, see Importing Records Incrementally (p. 54).

You create filters for a dataset group and apply filters to real-time and batch recommendations at the campaign level. To filter items, you first create a filter, which consists of a filter name and a SQL-like filter expression. You can either specify filter criteria when you create the filter or pass criteria as a parameter when you get recommendations.

You then apply the filter and specify filter parameter values when you call the GetRecommendations (p. 269) or GetPersonalizedRanking (p. 265) operations, or when you get recommendations from a campaign in the console.

For batch workflows, you include filter parameter values in your input JSON and apply the filter when you call the CreateBatchInferenceJob (p. 153) operation or create a batch inference job in the console. You can create, edit, delete, and apply filters using the Amazon Personalize console, the AWS Command Line Interface (AWS CLI), and the AWS SDKs.

For information about the number of filters you can create and how many parameters you can use in filter expressions, see Service Quotas (p. 147).

Important
To filter recommendations using a filter with parameters and a campaign that you deployed before November 10, 2020, you must redeploy the campaign by using the UpdateCampaign (p. 256) operation or create a new campaign.

Topics
- Filter Expressions (p. 113)
- Filtering Real-time Recommendations (p. 116)
- Filtering Batch Recommendations (p. 123)

Filter Expressions

To configure filters, you must use a properly formatted filter expression. Filter expressions are composed of dataset and property identifiers in dataset.property format, along with logical operators, keywords, and values. For values, you can specify fixed values or add placeholder parameters, which allow you to set the filter criteria when you get recommendations.

For a complete list of filter expression elements, see Filter Expression Elements (p. 114). For examples of filter expressions, see Filter Expression Examples (p. 115).

Note
Amazon Personalize ignores case only when matching event types.
Creating Filter Expressions

You can either manually create filter expressions or get help with expression syntax and structure by using the Expression builder (p. 119) in the console. You can use filter expressions to filter items based on data from the following datasets:

- **Interactions**: Use filter expressions to include or exclude items that a user has interacted with (for example, user events such as click or stream). Amazon Personalize filters recommendations based on the most recent 200 historical and 100 new interaction events recorded in a user's Interactions dataset.

- **Items**: Use filter expressions to include or exclude items based on specific item conditions.

- **Users**: Use filter expressions to include or exclude items based on specific CurrentUser conditions. If you have created a Users dataset, you can add CurrentUser to any expression regardless of the dataset that is being used in the expression.

Note
You can't chain Interaction and Item datasets into one expression. To create a filter that filters by Interaction and then Item datasets (or the opposite), you must chain two or more expressions together. For more information, see Multiple Expressions Example (p. 116).

Filter Expression Elements

Use the following elements to create filter expressions:

- **INCLUDE**: Use INCLUDE to make sure certain items are in recommendations.

- **EXCLUDE**: Use EXCLUDE to remove certain items from recommendations.

- **ItemID**: Use ItemID after the INCLUDE or EXCLUDE element.

- **WHERE**: Use WHERE to check conditions for items. You must use the WHERE element after the ItemID.

- **AND/OR**: Use AND or OR to chain multiple conditions together within the same filter expression. Conditions chained together using AND or OR can only affect properties of the dataset used in the first condition.

- **Dataset.property**: Provide the dataset and the metadata property that you want to filter recommendations by in the dataset.property format. For example, to filter based on the genres property in your Items dataset, you would use Items.genres in your filter expression.

- **IF**: Use IF only to check conditions for the CurrentUser and only once at the end of an expression. However, you can extend an IF condition using AND.

- **IN/NOT IN**: Use IN or NOT IN as comparison operators to filter based on matching (or not matching) one or more string values. Amazon Personalize filters only on exact strings.

- **Use =, <, <=, >, >= operators to test numerical data for equality.**

- **Use * to include or exclude interactions of all types. Use * only for filter expressions that use the EVENT_TYPE property of an Interactions dataset.**

- **Use the pipe separator (|) to chain multiple expressions together. For more information, see Multiple Expressions Example (p. 116).**

- **For expressions that use = and IN operators, use the dollar sign ($) and a parameter name to add a placeholder parameter as a value. For example, $GENRES. For this example, when you get recommendations, you supply the genre or genres to filter by. For information on the number of parameters you can use, see Service Quotas (p. 147).**

Note
You define a parameter name when you add it to an expression. The parameter name does not have to match the property name. We recommend that you use a parameter name that is similar to the property name and easy to remember. You use the parameter name (case sensitive) when you apply the filter to recommendations requests.
Filter Expression Examples

Use the following examples to learn how to build your own filter expressions. They are organized by dataset type.

Interactions

The following expression excludes items based on an event type (such as click) that you specify when you get recommendations using the $EVENT_TYPE parameter.

```
EXCLUDE ItemID WHERE Interactions.EVENT_TYPE IN ($EVENT_TYPE)
```

The following expression excludes items that a user clicked or streamed.

```
EXCLUDE ItemID WHERE Interactions.EVENT_TYPE IN ("click", "stream")
```

The following expression includes items that the user has interacted with in any way.

```
INCLUDE ItemID WHERE Interactions.EVENT_TYPE IN ("")
```

Items

The following expression excludes items based on a category that you specify when you get recommendations using the $CATEGORY parameter.

```
EXCLUDE ItemID WHERE Items.CATEGORY IN ($CATEGORY)
```

The following expression excludes items in the shoe category that do not have a description of boot.

```
EXCLUDE ItemID WHERE Items.CATEGORY IN ("shoe") AND Items.DESCRIPTION NOT IN ("boot")
```

The following expression includes items with a download count equal to a value you specify when you get recommendations using the $NUMBER_OF_DOWNLOADS parameter.

```
INCLUDE ItemID WHERE Items.NUMBER_OF_DOWNLOADS = $NUMBER_OF_DOWNLOADS
```

Users

The following expression excludes items with a genre that you specify when you get recommendations using the $GENRE parameter, but only if the current user's age is equal to the value that you specify when you get recommendations using the $AGE parameter.
EXCLUDE ItemID WHERE Items.GENRE IN ($GENRE) IF CurrentUser.AGE = $AGE

The following expression includes items in the watch category, with a description of luxury, if the current user's age is over 18.

INCLUDE ItemID WHERE Items.CATEGORY IN ("watch") AND Items.DESCRIPTION IN ("luxury") IF CurrentUser.AGE > 18

Multiple Expressions Example

To filter by multiple datasets, chain multiple expressions together using a pipe separator (|). Each expression is first evaluated independently and the result is a union of the two results.

The following example includes two expressions. The first includes items in a genre that you specify when you get recommendations using the $GENRE parameter. The second excludes items that the user has clicked or streamed. Recommendations will include items with a genre that you specify when you get recommendations, but only items that have not have been clicked or streamed.

INCLUDE ItemID WHERE Items.GENRE IN ($GENRE) | EXCLUDE ItemID WHERE Interactions.EVENT_TYPE IN ("click", "stream")

Filtering Real-time Recommendations

You can filter real-time recommendations with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or the AWS SDKs.

Filtering Real-time Recommendations (Console)

To filter real-time recommendations using the console, create a filter and then apply it to a recommendation request.

**Note**

To filter recommendations using a filter with parameters and a campaign that you deployed before November 10, 2020, you must redeploy the campaign by using the UpdateCampaign (p. 256) operation or create a new campaign.

Creating a Filter (Console)

To create a filter in the console, choose the dataset group that contains the campaign you want to filter results for and then provide a filter name and a filter expression.

**To create a filter (console)**

1. Open the Amazon Personalize console at [https://console.aws.amazon.com/personalize/](https://console.aws.amazon.com/personalize/) and sign into your account.
2. Choose the dataset group that contains the campaign that you want to filter results for.
3. In the navigation page, choose Filters and then choose Create new filter. The Create filter page displays.
4. For **Filter name**, enter a name for your filter. You will choose the filter by this name when you apply it to a recommendation request.

5. For **Expression**, choose either **Build expression** or **Add expression manually** and build or insert your expression:

   - To use the expression builder, choose **Build expression**. The expression builder provides structure, fields, and guidelines for building correctly formatted filter expressions. For more information, see Using the Filter Expression Builder (p. 119).
   
   - To input your own expression, choose **Add expression manually**. For more information, see Filter Expression Elements (p. 114).

6. Choose **Finish**. The filter's overview page shows the filter's Amazon Resource Name (ARN), status, and full filter expression. To delete the filter, choose **Delete**. For information about finding and deleting filters after you have left the overview page, see Deleting a Filter (Console) (p. 120).

### Applying a Filter (Console)

To apply a filter, on the **Test campaign results** panel for the campaign, choose the filter and enter any filter parameter values. Then get recommendations for a user.
Important
For filter expressions that use an INCLUDE element to include items, you must provide values for all parameters that are defined in the expression. For filters with expressions that use an EXCLUDE element to exclude items, you can omit the filter-values. In this case, Amazon Personalize doesn’t use that portion of the expression to filter recommendations.

To apply a filter (console)

1. In the navigation pane, choose Campaigns.
2. On the Campaigns page, choose the target campaign.
3. For comparison, start by getting recommendations for a user without applying a filter. Under Test campaign results, enter the ID of a user that you want to get recommendations for, and choose Get recommendations. A table containing the user’s top recommendations appears.

4. From the Filter name menu, choose the filter that you created. If your filter has any placeholder parameters, the associated fields for each parameter appear.
5. If you’re using a filter with placeholder parameters, for each parameter, enter the value to set the filter criteria. To use multiple values for one parameter, separate each value with a comma.
6. Using the same User ID as in the earlier step, choose Get recommendations. The recommendations table appears.

If the user already bought a recommended item, the filter removes it from the recommendation list. In this example, items 2657, 2985 were replaced by the most suitable items that the user had not purchased (items 2641 and 1573).
Using the Filter Expression Builder

The **Expression builder** on the **Create filter** page provides structure, fields, and guidelines for building correctly formatted filter expressions.

To build a filter expression:

- Use the **Action**, **Property**, **Operator**, and **Value** fields to create an expression.

  For the **Value**, enter a fixed value or, to set filter criteria when you get recommendations, enter `$ + a parameter name. For example, `$GENRES. When you get recommendations, you'll supply the value or values to filter by. In this example, you would provide a genre or list of genres when you get recommendations.

  Separate multiple non-parameter values with a comma. You cannot add comma-separated parameters to a filter.

  **Note**
  After you choose a **Property** (in `dataset.property` format), the **Property** value for any succeeding rows chained by AND or OR conditions must use the same dataset.

- Use the + and ‐ buttons to add or delete a row from your expression. You can't delete the first row.

- For new rows, use the AND, IF, or OR operators on the **AND** menu to create a chain of conditions.

  For **IF** conditions:

  - Each expression can contain only one **IF** item. If you remove an **IF** condition, the Expression builder removes any AND conditions following it.

  - You can use **IF** conditions only for expressions that filter by the **CurrentUser**.

  - Choose the **Add expression** button to add an additional filter expression for more precise filtering, including filtering using Items and Interactions datasets. Each expression is first evaluated independently and the result is a union of the two results.

  **Note**
  To create a filter that uses both Item and Interaction datasets, you **must** use multiple expressions.

**Expression Builder Example**

The following example shows how to build a filter that excludes items with a genre that you specify when you get recommendations (note the `$GENRES placeholder parameter), and with a **DOWNLOAD_COUNT** of more than 200, but only if the current user's age is greater than 17.
Deleting a Filter (Console)

Deleting a filter removes the filter from the list of filters for a dataset group.

**Important**
You can't delete a filter while a batch inference job is in progress.

**To delete a filter (console)**

1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/ and sign into your account.
2. From the Dataset groups list, choose the dataset group that contains the filter that you want to delete.
3. In the navigation pane, choose Filters.
4. From the list of filters, choose the filter that you want to delete and choose View Details. The filter details page appears.
5. Choose Delete and confirm the deletion in the confirmation dialog box.

Filtering Real-time Recommendations (AWS CLI)

To filter recommendations using the AWS CLI, you create a filter and then apply it by specifying the filter ARN in a GetRecommendations (p. 269) or GetPersonalizedRanking (p. 265) request.

**Important**
To filter recommendations using a filter with parameters and a campaign you deployed before November 10, 2020, you must re-deploy the campaign by using the UpdateCampaign (p. 256) call or create a new campaign.

Creating a Filter (AWS CLI)

Use the following create-filter operation to create a filter and specify the filter expression.

Replace the **Filter name** with the name of the filter, and the **Dataset group ARN** with the Amazon Resource Name (ARN) of the dataset group. Replace the sample **filterExpression** with your own filter expression.

```bash
aws personalize create-filter
  --name Filter name
  --dataset-group-arn dataset group arn
  --filter-expression "EXCLUDE ItemID WHERE Items.CATEGORY IN ($CATEGORY)"
```
If successful, the filter ARN is displayed. Record it for later use. To verify that the filter is active, use the DescribeFilter (p. 214) operation before you use the filter.

For more information about the API, see CreateFilter (p. 172). For more information about filter expressions, including examples, see Creating Filter Expressions (p. 114).

### Applying a Filter (AWS CLI)

When you use the `get-recommendations` or `get-personalized-ranking` operations, apply a filter by passing the filter-arn and any filter values as parameters.

The following is an example of the `get-recommendations` operation. Replace `Campaign ARN` with the Amazon Resource Name (ARN) of your campaign, `User ID` with the ID of the user that you are getting recommendations for, and `Filter ARN` with the ARN of your filter.

If your expression has any parameters, include the `filter-values` object. For each parameter in your filter expression, provide the parameter name (case sensitive) and the values. For example, if your filter expression has a `$GENRE` parameter, provide "GENRE" as the key, and a genre or genres, such as "Comedy", as the value. Separate multiple values with a comma. For example, "\"comedy\", \"drama\", \"horror\"\".

**Important**
For filter expressions that use an INCLUDE element to include items, you must provide values for all parameters that are defined in the expression. For filters with expressions that use an EXCLUDE element to exclude items, you can omit the `filter-values`. In this case, Amazon Personalize doesn't use that portion of the expression to filter recommendations.

```
aws personalize-runtime get-recommendations
   --campaign-arn Campaign ARN \
   --user-id User ID \
   --filter-arn Filter ARN \
   --filter-values '\{
       "PROPERTY1": "\"value\"",
       "PROPERTY2": "\"value1\",\"value2\",\"value3\""
   }'
```

### Deleting a Filter (AWS CLI)

Use the following `delete-filter` operation to delete a filter. Replace `filter ARN` with the ARN of the filter.

```
aws personalize delete-filter --filter-arn Filter ARN
```

### Filtering Real-time Recommendations (AWS Python SDK)

To filter recommendations using the AWS Python SDK, you create a filter and then apply it by specifying the filter ARN in a `GetRecommendations` (p. 269) or `GetPersonalizedRanking` (p. 265) request.

**Important**
To filter recommendations using a filter with parameters and a campaign you deployed before November 10, 2020, you must re-deploy the campaign by using the `UpdateCampaign` (p. 256) call or create a new campaign.

### Creating a Filter (AWS Python SDK)

Use the following `create_filter` method to create a filter. Replace `Filter Name` with the name of the filter, and `Dataset Group ARN` with the Amazon Resource Name (ARN) of the dataset group. Replace the example `filterExpression` with your own filter expression.
import boto3

personalize = boto3.client('personalize')

response = personalize.create_filter(
    name = 'Filter Name',
    datasetGroupArn = 'Dataset Group ARN',
    filterExpression = 'EXCLUDE ItemID WHERE Items.CATEGORY IN ($CATEGORY)
)

filter_arn = response['filterArn']

print("Filter ARN: " + filter_arn)

Record the filter ARN for later use. To verify that the filter is active, use the DescribeFilter (p. 214) operation before using the filter. For more information about the API, see CreateFilter (p. 172). For more information about filter expressions, including examples, see Creating Filter Expressions (p. 114).

Applying a Filter (AWS Python SDK)

When you use the get_recommendations or get_personalized_ranking methods, apply a filter by passing filterArn and any filter values as parameters.

The following is an example of the get_recommendations method. Replace Campaign ARN with the Amazon Resource Name (ARN) of your campaign, User ID with the ID of the user that you are getting recommendations for, and Filter ARN with the ARN of your filter.

For filterValues, for each optional parameter in your filter expression, provide the parameter name (case sensitive) and the value or values. For example, if your filter expression has a $GENRES parameter, provide "GENRES" as the key, and a genre or genres, such as "Comedy", as the value. For multiple values, separate each value with a comma. For example, "Comedy", "Drama", "Horror".

    Important
    For filter expressions that use an INCLUDE element to include items, you must provide values for all parameters that are defined in the expression. For filters with expressions that use an EXCLUDE element to exclude items, you can omit the filter-values. In this case, Amazon Personalize doesn't use that portion of the expression to filter recommendations.

import boto3

personalize_runtime = boto3.client("personalize-runtime")

response = personalize_runtime.get_recommendations(
    campaignArn = "Campaign ARN",
    userId = "User ID",
    filterArn = "Filter ARN",
    filterValues = {
        "Parameter name": "value1",
        "Parameter name": "value1","value2","value3"
        ....
    }
)

Deleting a Filter (AWS Python SDK)

Use the following delete_filter method to delete a filter. Replace filter ARN with the ARN of the filter.

import boto3

personalize = boto3.client("personalize")
Filtering Batch Recommendations

Filtering batch recommendations works nearly the same as filtering real-time recommendations. To filter batch recommendations, you create a filter (p. 116) and then apply it to a CreateBatchInferenceJob (p. 153) operation or new batch inference job in the Amazon Personalize console. Amazon Personalize then filters the recommendations from the batch job's output JSON file. For more information about batch inference jobs, see Getting Batch Recommendations (Amazon Personalize Console) (p. 110).

For filters with placeholder parameters, such as $GENRE, provide the values in a filterValues object in your input JSON. For a filterValues object, each key is a parameter name and each value is the criteria that you are passing as a parameter. For multiple values, separate each value with a comma. The following is an example of a JSON input file with filter values. The GENRES key corresponds to a $GENRES placeholder in the filter expression.

```
{"userId": "5","filterValues":{"GENRES":""horror","comedy","drama""}}
{"userId": "3","filterValues":{"GENRES":""horror","comedy""}}
{"userId": "34","filterValues":{"GENRES":""drama""}}
```

Filtering Batch Recommendations (Console)

1. Use the console or the SDK to create a filter (p. 116).
2. When you create the batch recommendation job, on the Create batch inference job page, for Filter configuration - optional, Filter name, choose the filter.

Filtering Batch Recommendations (AWS SDK)

1. Use the console or the SDK to create a filter (p. 116).
2. Include the FilterArn parameter in the CreateBatchInferenceJob (p. 153) request.

```python
import boto3
personalize = boto3.client("personalize")
```
```python
personalize_rec.create_batch_inference_job (
    solutionVersionArn = "Solution version ARN",
    jobName = "Batch job name",
    roleArn = "IAM role ARN",
    filterArn = "Filter ARN",
    jobInput =
        {"s3DataSource": {"path": "S3 input path"}}
    jobOutput =
        {"s3DataDestination": {"path": "S3 output path"}}
)
```
Security in Amazon Personalize

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon Personalize, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company’s requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon Personalize. The following topics show you how to configure Amazon Personalize to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon Personalize resources.

**Topics**
- Data Protection in Amazon Personalize (p. 125)
- Identity and Access Management for Amazon Personalize (p. 126)
- Logging and Monitoring in Amazon Personalize (p. 138)
- Compliance Validation for Amazon Personalize (p. 145)
- Resilience in Amazon Personalize (p. 145)
- Infrastructure Security in Amazon Personalize (p. 145)

Data Protection in Amazon Personalize

The AWS shared responsibility model applies to data protection in Amazon Personalize. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the AWS Security Blog.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.
• If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put sensitive identifying information, such as your customers’ account numbers, into free-form fields such as a Name field. This includes when you work with Amazon Personalize or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into Amazon Personalize or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don’t include credentials information in the URL to validate your request to that server.

**Data Encryption**

The following information explains where Amazon Personalize uses data encryption to protect your data.

**Encryption at Rest**

Amazon Personalize uses the default Amazon S3 key (SSE-S3) for server-side encryption of Amazon Personalize data placed in your S3 buckets. You can also use one of your own AWS Key Management Service (AWS KMS) keys.

**Encryption in Transit**

Amazon Personalize copies data out of your account and processes it in an internal AWS system. By default, Amazon Personalize uses TLS 1.2 with AWS certificates to encrypt data in transit.

**Key Management**

The default Amazon S3 key is managed by AWS. It is the responsibility of the customer to manage any customer-provided AWS Key Management Service (AWS KMS) keys.

**Identity and Access Management for Amazon Personalize**

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use Amazon Personalize resources. IAM is an AWS service that you can use with no additional charge.

**Topics**

- **Audience (p. 126)**
- **Authenticating With Identities (p. 127)**
- **Managing Access Using Policies (p. 129)**
- **How Amazon Personalize Works with IAM (p. 130)**
- **Amazon Personalize Identity-Based Policy Examples (p. 133)**
- **Troubleshooting Amazon Personalize Identity and Access (p. 136)**

**Audience**

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in Amazon Personalize.
**Service user** – If you use the Amazon Personalize service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more Amazon Personalize features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in Amazon Personalize, see Troubleshooting Amazon Personalize Identity and Access (p. 136).

**Service administrator** – If you're in charge of Amazon Personalize resources at your company, you probably have full access to Amazon Personalize. It's your job to determine which Amazon Personalize features and resources your employees should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with Amazon Personalize, see How Amazon Personalize Works with IAM (p. 130).

**IAM administrator** – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to Amazon Personalize. To view example Amazon Personalize identity-based policies that you can use in IAM, see Amazon Personalize Identity-Based Policy Examples (p. 133).

### Authenticating With Identities

Authentication is how you sign in to AWS using your identity credentials. For more information about signing in using the AWS Management Console, see Signing in to the AWS Management Console as an IAM user or root user in the IAM User Guide.

You must be authenticated (signed in to AWS) as the AWS account root user, an IAM user, or by assuming an IAM role. You can also use your company's single sign-on authentication or even sign in using Google or Facebook. In these cases, your administrator previously set up identity federation using IAM roles. When you access AWS using credentials from another company, you are assuming a role indirectly.

To sign in directly to the AWS Management Console, use your password with your root user email address or your IAM user name. You can access AWS programmatically using your root user or IAM users access keys. AWS provides SDK and command line tools to cryptographically sign your request using your credentials. If you don't use AWS tools, you must sign the request yourself. Do this using Signature Version 4, a protocol for authenticating inbound API requests. For more information about authenticating requests, see Signature Version 4 signing process in the AWS General Reference.

Regardless of the authentication method that you use, you might also be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Using multi-factor authentication (MFA) in AWS in the IAM User Guide.

### AWS account root user

When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account root user and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

### IAM Users and Groups

An IAM user is an identity within your AWS account that has specific permissions for a single person or application. An IAM user can have long-term credentials such as a user name and password or a set of access keys. To learn how to generate access keys, see Managing access keys for IAM users in the IAM
User Guide. When you generate access keys for an IAM user, make sure you view and securely save the key pair. You cannot recover the secret access key in the future. Instead, you must generate a new access key pair.

An IAM group is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named IAMAdmins and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the IAM User Guide.

IAM Roles

An IAM role is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the IAM User Guide.

IAM roles with temporary credentials are useful in the following situations:

- **Temporary IAM user permissions** – An IAM user can assume an IAM role to temporarily take on different permissions for a specific task.

- **Federated user access** – Instead of creating an IAM user, you can use existing identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as federated users. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated users and roles in the IAM User Guide.

- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the IAM User Guide.

- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.

- **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see Actions, Resources, and Condition Keys for Amazon Personalize in the Service Authorization Reference.

- **Service role** – A service role is an IAM role that a service assumes to perform actions on your behalf. Service roles provide access only within your account and cannot be used to grant access to services in other accounts. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the IAM User Guide.

- **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
• **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

---

## Managing Access Using Policies

You control access in AWS by creating policies and attaching them to IAM identities or AWS resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. You can sign in as the root user or an IAM user, or you can assume an IAM role. When you then make a request, AWS evaluates the related identity-based or resource-based policies. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see *Overview of JSON policies* in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

Every IAM entity (user or role) starts with no permissions. In other words, by default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

### Identity-Based Policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see *Creating IAM policies* in the *IAM User Guide*.

Identity-based policies can be further categorized as **inline policies** or **managed policies**. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see *Choosing between managed policies and inline policies* in the *IAM User Guide*.

### Resource-Based Policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must **specify a principal** in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.
Resource-based policies are inline policies that are located in that service. You can’t use AWS managed policies from IAM in a resource-based policy.

**Access Control Lists (ACLs)**

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the Amazon Simple Storage Service Developer Guide.

**Other Policy Types**

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of the entity’s identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the Principal field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the IAM User Guide.

- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the AWS Organizations User Guide.

- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session’s permissions are the intersection of the user or role’s identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the IAM User Guide.

**Multiple Policy Types**

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the IAM User Guide.

**How Amazon Personalize Works with IAM**

Before you use IAM to manage access to Amazon Personalize, you should understand what IAM features are available to use with Amazon Personalize. To get a high-level view of how Amazon Personalize and other AWS services work with IAM, see AWS Services That Work with IAM in the IAM User Guide.

**Topics**

- Amazon Personalize Identity-Based Policies (p. 131)
- Amazon Personalize Resource-Based Policies (p. 132)
- Authorization Based on Amazon Personalize Tags (p. 132)
- Amazon Personalize IAM Roles (p. 133)
Amazon Personalize Identity-Based Policies

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. Amazon Personalize supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see IAM JSON Policy Elements Reference in the IAM User Guide.

Actions

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

The Action element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as permission-only actions that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called dependent actions.

Include actions in a policy to grant permissions to perform the associated operation.

Policy actions in Amazon Personalize use the following prefix before the action: personalize:. For example, to create a dataset with the Amazon Personalize CreateDataset API operation, you include the personalize:CreateDataset action in the policy. Policy statements must include either an Action or NotAction element. Amazon Personalize defines its own set of actions that describe tasks that you can perform with this service.

To specify multiple actions in a single statement, separate them with commas as shown in the following command.

```
"Action": [  
  "personalize:action1",
  "personalize:action2"
]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word Describe, include the following action.

```
"Action": "personalize:Describe*"
```

To see a list of Amazon Personalize actions, see Actions Defined by Amazon Personalize in the IAM User Guide.

Resources

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

The Resource JSON policy element specifies the object or objects to which the action applies. Statements must include either a Resource or a NotResource element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as resource-level permissions.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

An Amazon Personalize dataset resource has the following ARN.
arn:${Partition}:personalize:${Region}:${Account}:dataset/${dataset-name}

For more information about the format of ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces.

For example, to specify the MyDataset dataset in your statement, use the following ARN.

"Resource": "arn:aws:personalize:us-east-1:123456789012:dataset/MyDataset"

To specify all datasets that belong to a specific account, use the wildcard (*), as shown in the following example.

"Resource": "arn:aws:personalize:us-east-1:123456789012:dataset/*"

Some Amazon Personalize actions, such as those for creating resources, cannot be performed on a specific resource. In those cases, you must use the wildcard (*).

"Resource": "*"

To see a list of Amazon Personalize resource types and their ARNs, see Resources Defined by Amazon Personalize in the IAM User Guide. To learn about the actions you can use to specify the ARN of each resource, see Actions Defined by Amazon Personalize.

### Condition Keys

Administrators can use AWS JSON policies to specify who has access to what. That is, which principal can perform actions on what resources, and under what conditions.

The Condition element (or Condition block) lets you specify conditions in which a statement is in effect. The Condition element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple Condition elements in a statement, or multiple keys in a single Condition element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the IAM User Guide.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the IAM User Guide.

### Examples

To view examples of Amazon Personalize identity-based policies, see Amazon Personalize Identity-Based Policy Examples (p. 133).

### Amazon Personalize Resource-Based Policies

Amazon Personalize does not support resource-based policies.

### Authorization Based on Amazon Personalize Tags

Amazon Personalize does not support tagging resources or controlling access based on tags.
Amazon Personalize IAM Roles

An IAM role is an entity within your AWS account that has specific permissions.

Using Temporary Credentials with Amazon Personalize

You can use temporary credentials to sign in with federation, assume an IAM role, or to assume a cross-account role. You obtain temporary security credentials by calling AWS Security Token Service (AWS STS) API operations such as AssumeRole or GetFederationToken.

Amazon Personalize supports using temporary credentials.

Service-Linked Roles

Service-linked roles allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view but not edit the permissions for service-linked roles.

Amazon Personalize does not support service-linked roles.

Service Roles

This feature allows a service to assume a service role on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your IAM account and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, doing so might break the functionality of the service.

Amazon Personalize supports service roles.

Amazon Personalize Identity-Based Policy Examples

By default, IAM users and roles don't have permission to create or modify Amazon Personalize resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see Creating Policies on the JSON Tab in the IAM User Guide.

Topics

- Policy Best Practices (p. 133)
- AWS Managed Policies (p. 134)
- Using the Amazon Personalize Console (p. 135)
- Allow Users to View Their Own Permissions (p. 135)
- Allowing Full Access to Amazon Personalize Resources (p. 135)
- Allowing Read-Only Access to Amazon Personalize Resources (p. 136)

Policy Best Practices

Identity-based policies are very powerful. They determine whether someone can create, access, or delete Amazon Personalize resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:
• **Get started using AWS managed policies** – To start using Amazon Personalize quickly, use AWS managed policies to give your employees the permissions they need. These policies are already available in your account and are maintained and updated by AWS. For more information, see Get started using permissions with AWS managed policies in the IAM User Guide.

• **Grant least privilege** – When you create custom policies, grant only the permissions required to perform a task. Start with a minimum set of permissions and grant additional permissions as necessary. Doing so is more secure than starting with permissions that are too lenient and then trying to tighten them later. For more information, see Grant least privilege in the IAM User Guide.

• **Enable MFA for sensitive operations** – For extra security, require IAM users to use multi-factor authentication (MFA) to access sensitive resources or API operations. For more information, see Using multi-factor authentication (MFA) in AWS in the IAM User Guide.

• **Use policy conditions for extra security** – To the extent that it's practical, define the conditions under which your identity-based policies allow access to a resource. For example, you can write conditions to specify a range of allowable IP addresses that a request must come from. You can also write conditions to allow requests only within a specified date or time range, or to require the use of SSL or MFA. For more information, see IAM JSON policy elements: Condition in the IAM User Guide.

---

**AWS Managed Policies**

AWS managed policies are policies that are created and managed by AWS. The following are examples of AWS managed policies you can attach to your IAM user or group to grant permissions for Amazon Personalize.

For information on attaching a policy to a user, see Changing Permissions for an IAM User in the IAM User Guide. For information on attaching a policy to a group, see Attaching a Policy to an IAM Group in the IAM User Guide.

**AmazonPersonalizeFullAccess Policy**

Instead of creating a new policy, you can attach the AWS managed AmazonPersonalizeFullAccess policy to your IAM users and roles. However, AmazonPersonalizeFullAccess provides more permissions than are necessary to use Amazon Personalize.

Instead of using the AmazonPersonalizeFullAccess policy, we recommend creating a new IAM policy that only grants the necessary permissions (see Creating a New IAM Policy (p. 8)).

The AmazonPersonalizeFullAccess policy allows IAM users to perform the following actions:

- Access all Amazon Personalize resources
- Publish and list metrics on Amazon CloudWatch
- List, read, write, and delete all objects in an Amazon S3 bucket that contains Personalize or personalize in the bucket name
- Pass a role to Amazon Personalize

**CloudWatchFullAccess**

To give users permission to monitor Amazon Personalize with CloudWatch, attach the CloudWatchFullAccess policy to your Amazon Personalize IAM users or groups. For more information, see Monitoring Amazon Personalize (p. 138).

The CloudWatchFullAccess policy is optional and allows IAM users to perform the following actions:

- Publish and list Amazon Personalize metrics in CloudWatch
- View metrics and metric statistics.
- Set metric based alarms.
Using the Amazon Personalize Console

To access the Amazon Personalize console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the Amazon Personalize resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won’t function as intended for entities (IAM users or roles) with that policy.

To ensure that those entities can still use the Amazon Personalize console, also attach the following AWS managed policy to the entities. For more information, see Adding Permissions to a User in the IAM User Guide.

AWSPersonalizeConsoleAccess

You don’t need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that you’re trying to perform.

Allow Users to View Their Own Permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Allowing Full Access to Amazon Personalize Resources

The following example gives an IAM user in your AWS account full access to all Amazon Personalize resources and actions.
Allowing Read-Only Access to Amazon Personalize Resources

In this example, you grant an IAM user in your AWS account read-only access to your Amazon Personalize resources, including Amazon Personalize datasets, dataset groups, solutions, and campaigns.

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "personalize:DescribeAlgorithm",
        "personalize:DescribeBatchInferenceJob",
        "personalize:DescribeCampaign",
        "personalize:DescribeDataset",
        "personalize:DescribeDatasetGroup",
        "personalize:DescribeDatasetImportJob",
        "personalize:DescribeDatasetImportJobRun",
        "personalize:DescribeEventTracker",
        "personalize:DescribeFeatureExportJob",
        "personalize:DescribeFeatureTransformation",
        "personalize:DescribeRecipe",
        "personalize:DescribeSchema",
        "personalize:DescribeSolution",
        "personalize:DescribeSolutionVersion",
        "personalize:GetSolutionMetrics",
        "personalize:ListBatchInferenceJobs",
        "personalize:ListCampaigns",
        "personalize:ListDatasetGroups",
        "personalize:ListDatasetImportJobs",
        "personalize:ListDatasets",
        "personalize:ListEventTrackers",
        "personalize:ListRecipes",
        "personalize:ListSchemas",
        "personalize:ListSolutions",
        "personalize:ListSolutionVersions"
      ],
      "Resource": "*"
    }
  ]
}
```

Troubleshooting Amazon Personalize Identity and Access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon Personalize and IAM.
I Am Not Authorized to Perform an Action in Amazon Personalize

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the mateojackson IAM user tries to use the console to view details about a widget but does not have personalize:GetWidget permissions.

User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: personalize:GetWidget on resource: my-example-widget

In this case, Mateo asks his administrator to update his policies to allow him to access the my-example-widget resource using the personalize:GetWidget action.

I Am Not Authorized to Perform iam:PassRole

If you receive an error that you're not authorized to perform the iam:PassRole action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password. Ask that person to update your policies to allow you to pass a role to Amazon Personalize.

Some AWS services allow you to pass an existing role to that service, instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named marymajor tries to use the console to perform an action in Amazon Personalize. However, the action requires the service to have permissions granted by a service role. Mary does not have permissions to pass the role to the service.

User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole

In this case, Mary asks her administrator to update her policies to allow her to perform the iam:PassRole action.

I Want to View My Access Keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, AKIAIOSFODNN7EXAMPLE) and a secret access key (for example, wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXBLEKEY). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

**Important**

Do not provide your access keys to a third party, even to help find your canonical user ID. By doing this, you might give someone permanent access to your account.
When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see Managing access keys in the IAM User Guide.

I'm an Administrator and Want to Allow Others to Access Amazon Personalize

To allow others to access Amazon Personalize, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in Amazon Personalize.

To get started right away, see Creating your first IAM delegated user and group in the IAM User Guide.

I Want to Allow People Outside of My AWS Account to Access My Amazon Personalize Resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon Personalize supports these features, see How Amazon Personalize Works with IAM (p. 130).
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the IAM User Guide.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the IAM User Guide.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the IAM User Guide.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the IAM User Guide.

Logging and Monitoring in Amazon Personalize

This section provides information about monitoring and logging Amazon Personalize with Amazon CloudWatch and AWS CloudTrail.

Topics
- Monitoring Amazon Personalize (p. 138)
- CloudWatch Metrics for Amazon Personalize (p. 141)
- Logging Amazon Personalize API Calls with AWS CloudTrail (p. 143)

Monitoring Amazon Personalize

With Amazon CloudWatch, you can get metrics associated with Amazon Personalize. You can set up alarms to notify you when one or more of these metrics fall outside a defined threshold. To see metrics, you can use Amazon CloudWatch, Amazon AWS Command Line Interface, or the CloudWatch API.
You can also see aggregated metrics, for a chosen period of time, by using the Amazon Personalize console.

### Using CloudWatch Metrics for Amazon Personalize

To use metrics, you must specify the following information:

- The metric name.
- The metric dimension. A *dimension* is a name-value pair that helps you to uniquely identify a metric.

You can get monitoring data for Amazon Personalize using the AWS Management Console, the AWS CLI, or the CloudWatch API. You can also use the CloudWatch API through one of the Amazon AWS Software Development Kits (SDKs) or the CloudWatch API tools. The console displays a series of graphs based on the raw data from the CloudWatch API. Depending on your needs, you might prefer to use either the graphs displayed in the console or retrieved from the API.

The following list shows some common uses for the metrics. These are suggestions to get you started, not a comprehensive list.

<table>
<thead>
<tr>
<th>How Do I?</th>
<th>Relevant Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>How do I track the number of events that have been recorded?</td>
<td>Monitor the PutEventsRequests metric.</td>
</tr>
<tr>
<td>How can I monitor the DatasetImportJob errors?</td>
<td>Use the DatasetImportJobError metric.</td>
</tr>
<tr>
<td>How can I monitor the latency of GetRecommendations calls?</td>
<td>Use the GetRecommendationsLatency metric.</td>
</tr>
</tbody>
</table>

You must have the appropriate CloudWatch permissions to monitor Amazon Personalize with CloudWatch. For more information, see [Authentication and Access Control for Amazon CloudWatch](https://aws.amazon.com/documentation/cloudwatch/).  

### Access Amazon Personalize Metrics

The following examples show how to access Amazon Personalize metrics using the CloudWatch console, the AWS CLI, and the CloudWatch API.

#### To view metrics (console)

2. Choose **Metrics**, choose the **All metrics** tab, and then choose **AWS/Personalize**.
3. Choose the metric dimension.
4. Choose the desired metric from the list, and choose a time period for the graph.

#### To view metrics for events received over a period of time (CLI)

- Open the AWS CLI and enter the following command:

```
aws cloudwatch get-metric-statistics \\n   --metric-name PutEventsRequests \\n   --start-time 2019-03-15T00:00:20Z \\n   --period 3600 \\n   --end-time 2019-03-16T00:00:00Z \\n   --namespace AWS/Personalize \
```

---

139
This example shows the events received for the given event tracker ARN over a period of time. For more information, see `get-metric-statistics`.

**To access metrics (CloudWatch API)**

- Call `GetMetricStatistics`. For more information, see the Amazon CloudWatch API Reference.

**Create an Alarm**

You can create a CloudWatch alarm that sends an Amazon Simple Notification Service (Amazon SNS) message when the alarm changes state. An alarm watches a single metric over a time period you specify. The alarm performs one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon SNS topic or an AWS Auto Scaling policy.

Alarms invoke actions for sustained state changes only. CloudWatch alarms do not invoke actions simply because they are in a particular state. The state must have changed and been maintained for a specified number of time periods.

**To set an alarm (console)**

1. Sign in to the AWS Management Console and open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, Choose **Alarms**, and then choose **Create alarm**. This launches the **Create Alarm Wizard**.
3. Choose **Select metric**.
4. In the **All metrics** tab, choose **AWS/Personalize**.
5. Choose **EventTrackerArn**, and then choose **PutEventsRequests** metrics.
6. Choose the **Graphed metrics** tab.
7. For **Statistic** choose **Sum**.
8. Choose **Select metric**.
9. Fill in the **Name** and **Description**. For **Whenever**, choose >, and then enter a maximum value of your choice.
10. If you want CloudWatch to send you email when the alarm state is reached, for **Whenever this alarm**, choose **State is ALARM**. To send alarms to an existing Amazon SNS topic, for **Send notification to**; choose an existing SNS topic. To set the name and email addresses for a new email subscription list, choose **New list**. CloudWatch saves the list and displays it in the field so you can use it to set future alarms.

   **Note**

   If you use **New list** to create a new Amazon SNS topic, the email addresses must be verified before the intended recipients receive notifications. Amazon SNS sends email only when the alarm enters an alarm state. If this alarm state change happens before the email addresses are verified, intended recipients do not receive a notification.

11. Choose **Create alarm**.

**To set an alarm (AWS CLI)**

- Open the AWS CLI, and then enter the following command. Change the value of the `alarm-actions` parameter to reference an Amazon SNS topic that you previously created.
aws cloudwatch put-metric-alarm \
  --alarm-name PersonalizeCLI \
  --alarm-description "Alarm when more than 10 events occur" \
  --metric-name PutEventsRequests \
  --namespace AWS/Personalize \
  --statistic Sum \
  --period 300 \
  --threshold 10 \
  --comparison-operator GreaterThanThreshold \
  --evaluation-periods 1 \
  --unit Count \
  --dimensions Name=EventTrackerArn,Value=EventTrackerArn \
  --alarm-actions SNSTopicArn

This example shows how to create an alarm for when more than 10 events occur for the given event tracker ARN within 5 minutes. For more information, see put-metric-alarm.

To set an alarm (CloudWatch API)

- Call PutMetricAlarm. For more information, see Amazon CloudWatch API Reference.

CloudWatch Metrics for Amazon Personalize

This section contains information about the Amazon CloudWatch metrics available for Amazon Personalize. You can also see an aggregate view of Amazon Personalize metrics from the Amazon Personalize console. For more information, see Monitoring Amazon Personalize (p. 138).

The following table lists the Amazon Personalize metrics. All metrics support these statistics: Average, Minimum, Maximum, Sum.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DatasetImportJobRequests</td>
<td>The number of successful CreateDatasetImportJob (p. 166) API calls.</td>
</tr>
<tr>
<td></td>
<td>Dimensions: DatasetGroupArn, DatasetArn, DatasetImportJobArn</td>
</tr>
<tr>
<td>DatasetImportJobError</td>
<td>The number of CreateDatasetImportJob API calls that resulted in an error.</td>
</tr>
<tr>
<td></td>
<td>Dimensions: DatasetGroupArn, DatasetArn, DatasetImportJobArn</td>
</tr>
<tr>
<td>DatasetImportJobExecutionTime</td>
<td>The time between the CreateDatasetImportJob API call and the completion (or failure) of the operation.</td>
</tr>
<tr>
<td></td>
<td>Dimensions: DatasetGroupArn, DatasetArn, DatasetImportJobArn</td>
</tr>
<tr>
<td></td>
<td>Unit: Seconds</td>
</tr>
<tr>
<td>DatasetSize</td>
<td>The size of data imported by the dataset import job.</td>
</tr>
<tr>
<td></td>
<td>Dimensions: DatasetGroupArn, DatasetArn, DatasetImportJobArn</td>
</tr>
<tr>
<td>Metric</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SolutionTrainingJobRequests</td>
<td>The number of successful CreateSolutionVersion API calls.</td>
</tr>
<tr>
<td></td>
<td>Dimensions: SolutionArn, SolutionVersionArn</td>
</tr>
<tr>
<td>SolutionTrainingJobError</td>
<td>The number of CreateSolutionVersion API calls that resulted in an error.</td>
</tr>
<tr>
<td></td>
<td>Dimensions: SolutionArn, SolutionVersionArn</td>
</tr>
<tr>
<td>SolutionTrainingJobExecutionTime</td>
<td>The time between the CreateSolutionVersion API call and the completion (or failure) of the operation.</td>
</tr>
<tr>
<td></td>
<td>Dimensions: SolutionArn, SolutionVersionArn</td>
</tr>
<tr>
<td></td>
<td>Unit: Seconds</td>
</tr>
<tr>
<td>GetPersonalizedRankingRequests</td>
<td>The number of successful GetPersonalizedRanking API calls.</td>
</tr>
<tr>
<td></td>
<td>Dimension: CampaignArn</td>
</tr>
<tr>
<td>GetPersonalizedRanking4xxErrors</td>
<td>The number of GetPersonalizedRanking API calls that returned a 4xx HTTP response code.</td>
</tr>
<tr>
<td></td>
<td>Dimension: CampaignArn</td>
</tr>
<tr>
<td>GetPersonalizedRanking5xxErrors</td>
<td>The number of GetPersonalizedRanking API calls that returned a 5xx HTTP response code.</td>
</tr>
<tr>
<td></td>
<td>Dimension: CampaignArn</td>
</tr>
<tr>
<td>GetPersonalizedRankingLatency</td>
<td>The time between receiving the GetPersonalizedRanking API call and the sending of recommendations (excludes 4xx and 5xx errors).</td>
</tr>
<tr>
<td></td>
<td>Dimension: CampaignArn</td>
</tr>
<tr>
<td></td>
<td>Unit: Milliseconds</td>
</tr>
<tr>
<td>GetRecommendationsRequests</td>
<td>The number of successful GetRecommendations API calls.</td>
</tr>
<tr>
<td></td>
<td>Dimension: CampaignArn</td>
</tr>
<tr>
<td>GetRecommendations4xxErrors</td>
<td>The number of GetRecommendations API calls that returned a 4xx HTTP response code.</td>
</tr>
<tr>
<td></td>
<td>Dimension: CampaignArn</td>
</tr>
<tr>
<td>GetRecommendations5xxErrors</td>
<td>The number of GetRecommendations API calls that returned a 5xx HTTP response code.</td>
</tr>
<tr>
<td></td>
<td>Dimension: CampaignArn</td>
</tr>
</tbody>
</table>
## Logging Amazon Personalize API Calls with AWS CloudTrail

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

### Amazon Personalize Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When supported event activity occurs in Amazon Personalize, 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 Amazon Personalize, 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

### Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetRecommendationsLatency</td>
<td>The time between receiving the GetRecommendations API call and the sending of recommendations (excludes 4xx and 5xx errors).</td>
</tr>
<tr>
<td></td>
<td>Dimension: CampaignArn</td>
</tr>
<tr>
<td></td>
<td>Unit: Milliseconds</td>
</tr>
<tr>
<td>PutEventsRequests</td>
<td>The number of successful PutEvents (p. 259) API calls.</td>
</tr>
<tr>
<td></td>
<td>Dimension: EventTrackerArn</td>
</tr>
<tr>
<td>PutEvents4xxErrors</td>
<td>The number of PutEvents API calls that returned a 4xx HTTP response code.</td>
</tr>
<tr>
<td></td>
<td>Dimension: EventTrackerArn</td>
</tr>
<tr>
<td>PutEvents5xxErrors</td>
<td>The number of PutEvents API calls that returned a 5xx HTTP response code.</td>
</tr>
<tr>
<td></td>
<td>Dimension: EventTrackerArn</td>
</tr>
<tr>
<td>PutEventsLatency</td>
<td>The time taken for the completion of the PutEvents API call (excludes 4xx and 5xx errors).</td>
</tr>
<tr>
<td></td>
<td>Dimension: EventTrackerArn</td>
</tr>
<tr>
<td></td>
<td>Unit: Milliseconds</td>
</tr>
</tbody>
</table>
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

Amazon Personalize supports logging every action (API operation) as an event in CloudTrail log files. For more information, see Actions (p. 150).

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 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.

**Example: Amazon Personalize Log File Entries**

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, 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 with actions for the ListDatasetGroups API operation:

```json
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "principal-id",
        "arn": "arn:aws:iam::user-arn",
        "accountId": "account-id",
        "accessKeyId": "access-key",
        "userName": "user-name"
    },
    "eventTime": "2018-11-22T02:18:03Z",
    "eventSource": "personalize.amazonaws.com",
    "eventName": "ListDatasetGroups",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "source-ip-address",
    "userAgent": "aws-cli/1.11.16 Python/2.7.11 Darwin/15.6.0 botocore/1.4.73",
    "requestParameters": null,
    "responseElements": {
        "datasetGroups": [
            {
                "name": "testdatasetgroup",
                "status": "ACTIVE",
                "creationDateTime": "Nov 5, 2018 6:06:01 AM"
            }
        ]
    }
}
```
Compliance Validation for Amazon Personalize

Third-party auditors assess the security and compliance of Amazon Personalize as part of multiple AWS compliance programs. These include SOC, PCI, HIPAA, and others.

For a list of AWS services in scope of specific compliance programs, see AWS Services in Scope by Compliance Program. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using Amazon Personalize is determined by the sensitivity of your data, your company’s compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- **Security and Compliance Quick Start Guides** – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- **Architecting for HIPAA Security and Compliance Whitepaper** – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- **AWS Compliance Resources** – This collection of workbooks and guides might apply to your industry and location.
- **Evaluating Resources with Rules** in the AWS Config Developer Guide – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- **AWS Security Hub** – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience in Amazon Personalize

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

Infrastructure Security in Amazon Personalize

As a managed service, Amazon Personalize is protected by the AWS global network security procedures that are described in the Amazon Web Services: Overview of Security Processes whitepaper.
You use AWS published API calls to access Amazon Personalize through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. We recommend TLS 1.2 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.
Quotas in Amazon Personalize

The following sections contain information about Amazon Personalize guidelines and quotas.

**Topics**
- Supported AWS Regions (p. 147)
- Compliance (p. 147)
- Service Quotas (p. 147)
- Requesting a Quota Increase (p. 149)

## Supported AWS Regions


## Compliance

For more information about Amazon Personalize compliance programs, see [AWS Compliance](https://aws.amazon.com/security/compliance/), [AWS Compliance Programs](https://aws.amazon.com/security/compliance/programs/), and [AWS Services in Scope by Compliance Program](https://aws.amazon.com/security/compliance/programs/).  

## Service Quotas

Your AWS account has the following quotas for Amazon Personalize.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Quota</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Interactions</strong></td>
<td></td>
</tr>
<tr>
<td>Minimum number of unique combined historical and event interactions (after filtering by <code>eventType</code> and <code>eventValueThreshold</code>, if provided) required to train a model (create a solution version).</td>
<td>1000</td>
</tr>
<tr>
<td>Maximum number of interactions that are considered by a model during training.</td>
<td>500 million</td>
</tr>
<tr>
<td>Maximum number of distinct event types combined with total number of metadata columns in Interactions datasets.</td>
<td>10</td>
</tr>
<tr>
<td><strong>Users</strong></td>
<td></td>
</tr>
<tr>
<td>Minimum number of unique users, with at least 2 interactions each, required to train a model (create a solution version).</td>
<td>25</td>
</tr>
<tr>
<td>Maximum number of users that are considered by a model during training.</td>
<td>50 million</td>
</tr>
<tr>
<td>Resource</td>
<td>Quota</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>Maximum number of metadata fields for a Users dataset.</td>
<td>5</td>
</tr>
<tr>
<td><strong>Items</strong></td>
<td></td>
</tr>
<tr>
<td>Maximum number of items that are considered by a model during training.</td>
<td>750 thousand</td>
</tr>
<tr>
<td>Maximum number of metadata fields for an Items dataset.</td>
<td>50</td>
</tr>
<tr>
<td><strong>Data Import APIs</strong></td>
<td></td>
</tr>
<tr>
<td>Maximum rate of <code>PutItems</code> requests.</td>
<td>10/second</td>
</tr>
<tr>
<td>Maximum rate of <code>PutUsers</code> requests.</td>
<td>10/second</td>
</tr>
<tr>
<td>Maximum rate of <code>PutEvents</code> requests.</td>
<td>1000/second</td>
</tr>
<tr>
<td>Maximum number of events in a <code>PutEvents</code> call.</td>
<td>10</td>
</tr>
<tr>
<td>Maximum size of an event.</td>
<td>10 KB</td>
</tr>
<tr>
<td><strong>Recipes</strong></td>
<td></td>
</tr>
<tr>
<td>Maximum amount of data for an individual dataset (Users, Items, or Interactions) for HRNN, SIMS, Popularity-Count, and Personalized-Ranking recipes.</td>
<td>100 GB</td>
</tr>
<tr>
<td>Maximum amount of data for Interactions dataset for HRNN-metadata and HRNN-coldstart recipes.</td>
<td>100 GB</td>
</tr>
<tr>
<td>Maximum amount of combined data for Users and Items datasets for HRNN-metadata and HRNN-coldstart recipes.</td>
<td>5 GB</td>
</tr>
<tr>
<td>Maximum number of cold start items the HRNN-Coldstart recipe supports to train a model (create a solution version).</td>
<td>80,000</td>
</tr>
<tr>
<td>Minimum number of cold start items the HRNN-Coldstart recipe requires to train a model (create a solution version).</td>
<td>100</td>
</tr>
<tr>
<td><strong>Filters</strong></td>
<td></td>
</tr>
<tr>
<td>Maximum number of filters per account</td>
<td>10 filters</td>
</tr>
<tr>
<td>Maximum number of parameters for a filter expression.</td>
<td>5 parameters</td>
</tr>
<tr>
<td>Maximum number of parameters across all filters in a dataset group.</td>
<td>10 parameters</td>
</tr>
<tr>
<td><strong>GetRecommendations / GetPersonalizedRanking Requests</strong></td>
<td></td>
</tr>
<tr>
<td>Maximum transaction rate (GetRecommendations and GetPersonalizedRanking requests).</td>
<td>2500/sec</td>
</tr>
</tbody>
</table>
### Requesting a Quota Increase

For adjustable quotas, you can request a quota increase. The following Amazon Personalize quotas are adjustable:

- Total number of active campaigns
- Maximum number of filters per account
- Total number of pending or in progress batch inference jobs
- Total number of pending or in progress solution versions
- Maximum rate of \texttt{PutEvents} requests

To request a quota increase, use the \textit{Service Quotas} console and follow the steps in the \textit{Requesting a Quota Increase} section of the \textit{Service Quotas User Guide}. 

<table>
<thead>
<tr>
<th>Resource</th>
<th>Quota</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of GetRecommendations requests per second per campaign.</td>
<td>500/sec</td>
</tr>
<tr>
<td>Maximum number of GetPersonalizedRanking requests per second per campaign.</td>
<td>500/sec</td>
</tr>
<tr>
<td>Batch Inference Jobs</td>
<td></td>
</tr>
<tr>
<td>Maximum number of input files in a batch inference job.</td>
<td>1000</td>
</tr>
<tr>
<td>Maximum size of batch inference job input.</td>
<td>1 GB</td>
</tr>
<tr>
<td>Maximum number of records per input file in a batch inference job.</td>
<td>50 million</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Resource</th>
<th>Quota</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of active schemas.</td>
<td>500</td>
</tr>
<tr>
<td>Total number of active datasets.</td>
<td>500</td>
</tr>
<tr>
<td>Total number of active dataset groups.</td>
<td>500</td>
</tr>
<tr>
<td>Total number of active event trackers.</td>
<td>500</td>
</tr>
<tr>
<td>Total number of active solutions.</td>
<td>500</td>
</tr>
<tr>
<td>Total number of active campaigns.</td>
<td>5</td>
</tr>
<tr>
<td>Total number of pending or in progress batch inference jobs</td>
<td>5</td>
</tr>
<tr>
<td>Total number of pending or in progress solution versions</td>
<td>20</td>
</tr>
</tbody>
</table>
API Reference

This section provides documentation for the Amazon Personalize API operations.

Actions

The following actions are supported by Amazon Personalize:

- CreateBatchInferenceJob (p. 153)
- CreateCampaign (p. 157)
- CreateDataset (p. 160)
- CreateDatasetGroup (p. 163)
- CreateDatasetImportJob (p. 166)
- CreateEventTracker (p. 169)
- CreateFilter (p. 172)
- CreateSchema (p. 174)
- CreateSolution (p. 176)
- CreateSolutionVersion (p. 181)
- DeleteCampaign (p. 184)
- DeleteDataset (p. 186)
- DeleteDatasetGroup (p. 188)
- DeleteEventTracker (p. 190)
- DeleteFilter (p. 192)
- DeleteSchema (p. 194)
- DeleteSolution (p. 196)
- DescribeAlgorithm (p. 198)
- DescribeBatchInferenceJob (p. 200)
- DescribeCampaign (p. 202)
- DescribeDataset (p. 204)
- DescribeDatasetGroup (p. 206)
- DescribeDatasetImportJob (p. 208)
- DescribeEventTracker (p. 210)
- DescribeFeatureTransformation (p. 212)
- DescribeFilter (p. 214)
- DescribeRecipe (p. 216)
- DescribeSchema (p. 218)
- DescribeSolution (p. 220)
- DescribeSolutionVersion (p. 223)
- GetSolutionMetrics (p. 226)
- ListBatchInferenceJobs (p. 228)
- ListCampaigns (p. 231)
- ListDatasetGroups (p. 234)
- ListDatasetImportJobs (p. 236)
ListDatasets (p. 239)
ListEventTrackers (p. 242)
ListFilters (p. 244)
ListRecipes (p. 246)
ListSchemas (p. 248)
ListSolutions (p. 250)
ListSolutionVersions (p. 253)
UpdateCampaign (p. 256)

The following actions are supported by Amazon Personalize Events:
- PutEvents (p. 259)
- PutItems (p. 261)
- PutUsers (p. 263)

The following actions are supported by Amazon Personalize Runtime:
- GetPersonalizedRanking (p. 265)
- GetRecommendations (p. 269)

Amazon Personalize

The following actions are supported by Amazon Personalize:
- CreateBatchInferenceJob (p. 153)
- CreateCampaign (p. 157)
- CreateDataset (p. 160)
- CreateDatasetGroup (p. 163)
- CreateDatasetImportJob (p. 166)
- CreateEventTracker (p. 169)
- CreateFilter (p. 172)
- CreateSchema (p. 174)
- CreateSolution (p. 176)
- CreateSolutionVersion (p. 181)
- DeleteCampaign (p. 184)
- DeleteDataset (p. 186)
- DeleteDatasetGroup (p. 188)
- DeleteEventTracker (p. 190)
- DeleteFilter (p. 192)
- DeleteSchema (p. 194)
- DeleteSolution (p. 196)
- DescribeAlgorithm (p. 198)
- DescribeBatchInferenceJob (p. 200)
- DescribeCampaign (p. 202)
- DescribeDataset (p. 204)
- DescribeDatasetGroup (p. 206)
- DescribeDatasetImportJob (p. 208)
• DescribeEventTracker (p. 210)
• DescribeFeatureTransformation (p. 212)
• DescribeFilter (p. 214)
• DescribeRecipe (p. 216)
• DescribeSchema (p. 218)
• DescribeSolution (p. 220)
• DescribeSolutionVersion (p. 223)
• GetSolutionMetrics (p. 226)
• ListBatchInferenceJobs (p. 228)
• ListCampaigns (p. 231)
• ListDatasetGroups (p. 234)
• ListDatasetImportJobs (p. 236)
• ListDatasets (p. 239)
• ListEventTrackers (p. 242)
• ListFilters (p. 244)
• ListRecipes (p. 246)
• ListSchemas (p. 248)
• ListSolutions (p. 250)
• ListSolutionVersions (p. 253)
• UpdateCampaign (p. 256)
CreateBatchInferenceJob
Service: Amazon Personalize

Creates a batch inference job. The operation can handle up to 50 million records and the input file must be in JSON format. For more information, see Getting Batch Recommendations (p. 108).

Request Syntax

```
{
    "batchInferenceJobConfig": {
        "itemExplorationConfig": {
            "string": "string"
        }
    },
    "filterArn": "string",
    "jobInput": {
        "s3DataSource": {
            "kmsKeyArn": "string",
            "path": "string"
        }
    },
    "jobName": "string",
    "jobOutput": {
        "s3DataDestination": {
            "kmsKeyArn": "string",
            "path": "string"
        }
    },
    "numResults": number,
    "roleArn": "string",
    "solutionVersionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

batchInferenceJobConfig (p. 153)

The configuration details of a batch inference job.

Type: BatchInferenceJobConfig (p. 283) object

Required: No

filterArn (p. 153)

The ARN of the filter to apply to the batch inference job. For more information on using filters, see Using Filters with Amazon Personalize.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+.*+.+

Required: No

jobInput (p. 153)

The Amazon S3 path that leads to the input file to base your recommendations on. The input material must be in JSON format.
Type: `BatchInferenceJobInput` (p. 284) object

**jobName (p. 153)**

The name of the batch inference job to create.

Type: String


Pattern: `^[a-zA-Z0-9][a-zA-Z0-9-\.\_]*`

Required: Yes

**jobOutput (p. 153)**

The path to the Amazon S3 bucket where the job’s output will be stored.

Type: `BatchInferenceJobOutput` (p. 285) object

Required: Yes

**numResults (p. 153)**

The number of recommendations to retrieve.

Type: Integer

Required: No

**roleArn (p. 153)**

The ARN of the Amazon Identity and Access Management role that has permissions to read and write to your input and out Amazon S3 buckets respectively.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:(\[a-z\d-\]+):iam::\d{12}:role/?[a-zA-Z_0-9=,\@-_/]+`

Required: Yes

**solutionVersionArn (p. 153)**

The Amazon Resource Name (ARN) of the solution version that will be used to generate the batch inference recommendations.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:\([a-zA-Z\d-]+\):personalize::\.*\.*\.*\.*\.*`

Required: Yes

**Response Syntax**

```json
{
    "batchInferenceJobArn": "string"
}
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

```
batchInferenceJobArn (p. 154)
```

- The ARN of the batch inference job.
- Type: String
- Length Constraints: Maximum length of 256.
- Pattern: `arn:([a-z\d-]+):personalize::*:*::*`.

Errors

```
InvalidInputException
```

- Provide a valid value for the field or parameter.
- HTTP Status Code: 400

```
LimitExceededException
```

- The limit on the number of requests per second has been exceeded.
- HTTP Status Code: 400

```
ResourceAlreadyExistsException
```

- The specified resource already exists.
- HTTP Status Code: 400

```
ResourceInUseException
```

- The specified resource is in use.
- HTTP Status Code: 400

```
ResourceNotFoundException
```

- Could not find the specified resource.
- HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
• AWS SDK for Ruby V3
CreateCampaign
Service: Amazon Personalize

Creates a campaign by deploying a solution version. When a client calls the GetRecommendations and GetPersonalizedRanking APIs, a campaign is specified in the request.

Minimum Provisioned TPS and Auto-Scaling

A transaction is a single GetRecommendations or GetPersonalizedRanking call. Transactions per second (TPS) is the throughput and unit of billing for Amazon Personalize. The minimum provisioned TPS (minProvisionedTPS) specifies the baseline throughput provisioned by Amazon Personalize, and thus, the minimum billing charge.

If your TPS increases beyond minProvisionedTPS, Amazon Personalize auto-scales the provisioned capacity up and down, but never below minProvisionedTPS. There's a short time delay while the capacity is increased that might cause loss of transactions.

The actual TPS used is calculated as the average requests/second within a 5-minute window. You pay for maximum of either the minimum provisioned TPS or the actual TPS. We recommend starting with a low minProvisionedTPS, track your usage using Amazon CloudWatch metrics, and then increase the minProvisionedTPS as necessary.

Status

A campaign can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

To get the campaign status, call DescribeCampaign (p. 202).

Note

Wait until the status of the campaign is ACTIVE before asking the campaign for recommendations.

Related APIs

- ListCampaigns (p. 231)
- DescribeCampaign (p. 202)
- UpdateCampaign (p. 256)
- DeleteCampaign (p. 184)

Request Syntax

```json
{
  "campaignConfig": {
    "itemExplorationConfig": {
      "string" : "string"
    },
    "minProvisionedTPS": number,
    "name": "string",
    "solutionVersionArn": "string"
  }
}
```

Request Parameters

The request accepts the following data in JSON format.
**campaignConfig (p. 157)**

The configuration details of a campaign.

Type: CampaignConfig (p. 290) object

Required: No

**minProvisionedTPS (p. 157)**

Specifies the requested minimum provisioned transactions (recommendations) per second that Amazon Personalize will support.

Type: Integer

Valid Range: Minimum value of 1.

Required: Yes

**name (p. 157)**

A name for the new campaign. The campaign name must be unique within your account.

Type: String


Pattern: `^[a-zA-Z0-9][a-zA-Z0-9-_]*`

Required: Yes

**solutionVersionArn (p. 157)**

The Amazon Resource Name (ARN) of the solution version to deploy.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.+:..`

Required: Yes

---

**Response Syntax**

```json
{
  "campaignArn": "string"
}
```

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**campaignArn (p. 158)**

The Amazon Resource Name (ARN) of the campaign.

Type: String

Length Constraints: Maximum length of 256.
Pattern: arn:([a-zA-Z0-9\-]+):personalize:.+:.*+:.*+

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

LimitExceededException

The limit on the number of requests per second has been exceeded.

HTTP Status Code: 400

ResourceAlreadyExistsException

The specified resource already exists.

HTTP Status Code: 400

ResourceInUseException

The specified resource is in use.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateDataset
Service: Amazon Personalize

Creates an empty dataset and adds it to the specified dataset group. Use
CreateDatasetImportJob (p. 166) to import your training data to a dataset.

There are three types of datasets:

- Interactions
- Items
- Users

Each dataset type has an associated schema with required field types. Only the Interactions dataset
is required in order to train a model (also referred to as creating a solution).

A dataset can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

To get the status of the dataset, call DescribeDataset (p. 204).

Related APIs

- CreateDatasetGroup (p. 163)
- ListDatasets (p. 239)
- DescribeDataset (p. 204)
- DeleteDataset (p. 186)

Request Syntax

```
{
    "datasetGroupArn": "string",
    "datasetType": "string",
    "name": "string",
    "schemaArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn (p. 160)

The Amazon Resource Name (ARN) of the dataset group to add the dataset to.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.*:.*:+

Required: Yes
**datasetType (p. 160)**

The type of dataset.

One of the following (case insensitive) values:

- Interactions
- Items
- Users

Type: String

Length Constraints: Maximum length of 256.

Required: Yes

**name (p. 160)**

The name for the dataset.

Type: String


Pattern: ^[a-zA-Z0-9][a-zA-Z0-9-\_]*$

Required: Yes

**schemaArn (p. 160)**

The ARN of the schema to associate with the dataset. The schema defines the dataset fields.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:(\[a-z\d-\]+):personalize:\.*:.*:.+

Required: Yes

**Response Syntax**

```
{
  "datasetArn": "string"
}
```

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**datasetArn (p. 161)**

The ARN of the dataset.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.+:.
Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

LimitExceededException

The limit on the number of requests per second has been exceeded.

HTTP Status Code: 400

ResourceAlreadyExistsException

The specified resource already exists.

HTTP Status Code: 400

ResourceInUseException

The specified resource is in use.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateDatasetGroup
Service: Amazon Personalize

Creates an empty dataset group. A dataset group contains related datasets that supply data for training a model. A dataset group can contain at most three datasets, one for each type of dataset:

- Interactions
- Items
- Users

To train a model (create a solution), a dataset group that contains an Interactions dataset is required. Call CreateDataset (p. 160) to add a dataset to the group.

A dataset group can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING

To get the status of the dataset group, call DescribeDatasetGroup (p. 206). If the status shows as CREATE FAILED, the response includes a failureReason key, which describes why the creation failed.

**Note**
You must wait until the status of the dataset group is ACTIVE before adding a dataset to the group.

You can specify an AWS Key Management Service (KMS) key to encrypt the datasets in the group. If you specify a KMS key, you must also include an AWS Identity and Access Management (IAM) role that has permission to access the key.

**APIs that require a dataset group ARN in the request**

- CreateDataset (p. 160)
- CreateEventTracker (p. 169)
- CreateSolution (p. 176)

**Related APIs**

- ListDatasetGroups (p. 234)
- DescribeDatasetGroup (p. 206)
- DeleteDatasetGroup (p. 188)

**Request Syntax**

```json
{
"kmsKeyArn": "string",
"name": "string",
"roleArn": "string"
}
```

**Request Parameters**

The request accepts the following data in JSON format.
kmsKeyArn (p. 163)

The Amazon Resource Name (ARN) of a KMS key used to encrypt the datasets.

Type: String
Required: No

name (p. 163)

The name for the new dataset group.

Type: String
Pattern: ^[a-zA-Z0-9]\[a-zA-Z0-9\-_]*
Required: Yes

roleArn (p. 163)

The ARN of the IAM role that has permissions to access the KMS key. Supplying an IAM role is only valid when also specifying a KMS key.

Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+
Required: No

Response Syntax

```
{
   "datasetGroupArn": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

datasetGroupArn (p. 164)

The Amazon Resource Name (ARN) of the new dataset group.

Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-zA-Z\d-]+):personalize:.+*:.+

Errors

InvalidInputException

Provide a valid value for the field or parameter.
HTTP Status Code: 400

**LimitExceededException**

The limit on the number of requests per second has been exceeded.

HTTP Status Code: 400

**ResourceAlreadyExistsException**

The specified resource already exists.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateDatasetImportJob
Service: Amazon Personalize

Creates a job that imports training data from your data source (an Amazon S3 bucket) to an Amazon Personalize dataset. To allow Amazon Personalize to import the training data, you must specify an AWS Identity and Access Management (IAM) role that has permission to read from the data source, as Amazon Personalize makes a copy of your data and processes it in an internal AWS system.

**Important**
The dataset import job replaces any previous data in the dataset.

**Status**

A dataset import job can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED

To get the status of the import job, call `DescribeDatasetImportJob` (p. 208), providing the Amazon Resource Name (ARN) of the dataset import job. The dataset import is complete when the status shows as ACTIVE. If the status shows as CREATE FAILED, the response includes a `failureReason` key, which describes why the job failed.

**Note**
Importing takes time. You must wait until the status shows as ACTIVE before training a model using the dataset.

**Related APIs**

- `ListDatasetImportJobs` (p. 236)
- `DescribeDatasetImportJob` (p. 208)

**Request Syntax**

```
{
    "datasetArn": "string",
    "dataSource": {
        "dataLocation": "string"
    },
    "jobName": "string",
    "roleArn": "string"
}
```

**Request Parameters**

The request accepts the following data in JSON format.

**datasetArn (p. 166)**

The ARN of the dataset that receives the imported data.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:(a-z\d-\+):personalize:.*:*:*:*`

Required: Yes
**dataSource (p. 166)**

The Amazon S3 bucket that contains the training data to import.

Type: [DataSource (p. 312)] object

Required: Yes

**jobName (p. 166)**

The name for the dataset import job.

Type: String


Pattern: ^\[a-zA-Z0-9][a-zA-Z0-9\-\_]*$

Required: Yes

**roleArn (p. 166)**

The ARN of the IAM role that has permissions to read from the Amazon S3 data source.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:\([a-z\d\-]+\):iam::\d{12}:role/\?[a-zA-Z_0-9+=,.@\-_/]+$

Required: Yes

**Response Syntax**

```
{
    "datasetImportJobArn": "string"
}
```

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**datasetImportJobArn (p. 167)**

The ARN of the dataset import job.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:\([a-z\d\-]+\):personalize:.+:.*:+

**Errors**

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400
LimitExceededException

The limit on the number of requests per second has been exceeded.

HTTP Status Code: 400

ResourceAlreadyExistsException

The specified resource already exists.

HTTP Status Code: 400

ResourceInUseException

The specified resource is in use.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateEventTracker
Service: Amazon Personalize

Creates an event tracker that you use when adding event data to a specified dataset group using the PutEvents API.

Note
Only one event tracker can be associated with a dataset group. You will get an error if you call CreateEventTracker using the same dataset group as an existing event tracker.

When you create an event tracker, the response includes a tracking ID, which you pass as a parameter when you use the PutEvents operation. Amazon Personalize then appends the event data to the Interactions dataset of the dataset group you specify in your event tracker.

The event tracker can be in one of the following states:

• CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
• DELETE PENDING > DELETE IN_PROGRESS

To get the status of the event tracker, call DescribeEventTracker (p. 210).

Note
The event tracker must be in the ACTIVE state before using the tracking ID.

Related APIs
• ListEventTrackers (p. 242)
• DescribeEventTracker (p. 210)
• DeleteEventTracker (p. 190)

Request Syntax

```
{
  "datasetGroupArn": "string",
  "name": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn (p. 169)
The Amazon Resource Name (ARN) of the dataset group that receives the event data.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.+:.+

Required: Yes

name (p. 169)
The name for the event tracker.

Type: String

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-\_]*$

Required: Yes

Response Syntax

```
{
  "eventTrackerArn": "string",
  "trackingId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**eventTrackerArn (p. 170)**

The ARN of the event tracker.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.*+:

**trackingId (p. 170)**

The ID of the event tracker. Include this ID in requests to the PutEvents API.

Type: String

Length Constraints: Maximum length of 256.

Errors

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**LimitExceededException**

The limit on the number of requests per second has been exceeded.

HTTP Status Code: 400

**ResourceAlreadyExistsException**

The specified resource already exists.

HTTP Status Code: 400

**ResourceInUseException**

The specified resource is in use.

HTTP Status Code: 400
ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
**CreateFilter**  
Service: Amazon Personalize

Creates a recommendation filter. For more information, see Filtering Recommendations (p. 113).

**Request Syntax**

```
{
    "datasetGroupArn": "string",
    "filterExpression": "string",
    "name": "string"
}
```

**Request Parameters**

The request accepts the following data in JSON format.

- **datasetGroupArn (p. 172)**
  
  The ARN of the dataset group that the filter will belong to.

  Type: String

  Length Constraints: Maximum length of 256.

  Pattern: arn:([a-z\d-]+):personalize:.+:.*:*:*:*

  Required: Yes

- **filterExpression (p. 172)**

  The filter expression defines which items are included or excluded from recommendations. Filter expression must follow specific format rules. For information about filter expression structure and syntax, see Filter Expressions (p. 113).

  Type: String


  Required: Yes

- **name (p. 172)**

  The name of the filter to create.

  Type: String


  Pattern: ^[a-zA-Z0-9-][a-zA-Z0-9-\_]*

  Required: Yes

**Response Syntax**

```
{
    "filterArn": "string"
}
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response. The following data is returned in JSON format by the service.

**filterArn (p. 172)**

The ARN of the new filter.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.*:+

Errors

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**LimitExceededException**

The limit on the number of requests per second has been exceeded.

HTTP Status Code: 400

**ResourceAlreadyExistsException**

The specified resource already exists.

HTTP Status Code: 400

**ResourceNotFoundException**

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
**CreateSchema**

Service: Amazon Personalize

Creates an Amazon Personalize schema from the specified schema string. The schema you create must be in Avro JSON format.

Amazon Personalize recognizes three schema variants. Each schema is associated with a dataset type and has a set of required field and keywords. You specify a schema when you call CreateDataset (p. 160).

For more information on schemas, see Datasets and Schemas.

**Related APIs**

- ListSchemas (p. 248)
- DescribeSchema (p. 218)
- DeleteSchema (p. 194)

**Request Syntax**

```json
{
   "name": "string",
   "schema": "string"
}
```

**Request Parameters**

The request accepts the following data in JSON format.

**name (p. 174)**

The name for the schema.

Type: String


Pattern: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

Required: Yes

**schema (p. 174)**

A schema in Avro JSON format.

Type: String

Length Constraints: Maximum length of 10000.

Required: Yes

**Response Syntax**

```json
{
   "schemaArn": "string"
}
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

schemaArn (p. 174)

The Amazon Resource Name (ARN) of the created schema.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:*:*:*+

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

LimitExceededException

The limit on the number of requests per second has been exceeded.

HTTP Status Code: 400

ResourceAlreadyExistsException

The specified resource already exists.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateSolution
Service: Amazon Personalize

Creates the configuration for training a model. A trained model is known as a solution. After the configuration is created, you train the model (create a solution) by calling the CreateSolutionVersion (p. 181) operation. Every time you call CreateSolutionVersion, a new version of the solution is created.

After creating a solution version, you check its accuracy by calling GetSolutionMetrics (p. 226). When you are satisfied with the version, you deploy it using CreateCampaign (p. 157). The campaign provides recommendations to a client through the GetRecommendations API.

To train a model, Amazon Personalize requires training data and a recipe. The training data comes from the dataset group that you provide in the request. A recipe specifies the training algorithm and a feature transformation. You can specify one of the predefined recipes provided by Amazon Personalize. Alternatively, you can specify performAutoML and Amazon Personalize will analyze your data and select the optimum USER_PERSONALIZATION recipe for you.

Status
A solution can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

To get the status of the solution, call DescribeSolution (p. 220). Wait until the status shows as ACTIVE before calling CreateSolutionVersion.

Related APIs
- ListSolutions (p. 250)
- CreateSolutionVersion (p. 181)
- DescribeSolution (p. 220)
- DeleteSolution (p. 196)

- ListSolutionVersions (p. 253)
- DescribeSolutionVersion (p. 223)

Request Syntax

```
{
    "datasetGroupArn": "string",
    "eventType": "string",
    "name": "string",
    "performAutoML": boolean,
    "performHPO": boolean,
    "recipeArn": "string",
    "solutionConfig": {
        "algorithmHyperParameters": { 
            "string": "string"
        },
        "autoMLConfig": {
            "metricName": "string",
            "recipeList": [ "string" ]
        },
        "eventValueThreshold": "string",
```
"featureTransformationParameters": {
    "string": "string"
},
"hpoConfig": {
    "algorithmHyperParameterRanges": {
        "categoricalHyperParameterRanges": [
            {
                "name": "string",
                "values": [ "string" ]
            }
        ],
        "continuousHyperParameterRanges": [
            {
                "maxValue": number,
                "minValue": number,
                "name": "string"
            }
        ],
        "integerHyperParameterRanges": [
            {
                "maxValue": number,
                "minValue": number,
                "name": "string"
            }
        ]
    },
    "hpoObjective": {
        "metricName": "string",
        "metricRegex": "string",
        "type": "string"
    },
    "hpoResourceConfig": {
        "maxNumberOfTrainingJobs": "string",
        "maxParallelTrainingJobs": "string"
    }
}

**Request Parameters**

The request accepts the following data in JSON format.

**datasetGroupArn (p. 176)**

The Amazon Resource Name (ARN) of the dataset group that provides the training data.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.*:.*+

Required: Yes

**eventType (p. 176)**

When you have multiple event types (using an EVENT_TYPE schema field), this parameter specifies which event type (for example, ‘click’ or ‘like’) is used for training the model.

If you do not provide an eventType, Amazon Personalize will use all interactions for training with equal weight regardless of type.

Type: String
Length Constraints: Maximum length of 256.
Required: No

**name (p. 176)**

The name for the solution.
Type: String
Pattern: ^[a-zA-Z0-9][a-zA-Z0-9-\-_]*$
Required: Yes

**performAutoML (p. 176)**

Whether to perform automated machine learning (AutoML). The default is false. For this case, you must specify recipeArn.

When set to true, Amazon Personalize analyzes your training data and selects the optimal USER_PERSONALIZATION recipe and hyperparameters. In this case, you must omit recipeArn. Amazon Personalize determines the optimal recipe by running tests with different values for the hyperparameters. AutoML lengthens the training process as compared to selecting a specific recipe.
Type: Boolean
Required: No

**performHPO (p. 176)**

Whether to perform hyperparameter optimization (HPO) on the specified or selected recipe. The default is false.

When performing AutoML, this parameter is always true and you should not set it to false.
Type: Boolean
Required: No

**recipeArn (p. 176)**

The ARN of the recipe to use for model training. Only specified when performAutoML is false.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:\([a-z\|d-]+):personalize:.+:.*+.
Required: No

**solutionConfig (p. 176)**

The configuration to use with the solution. When performAutoML is set to true, Amazon Personalize only evaluates the autoMLConfig section of the solution configuration.
Type: SolutionConfig (p. 340) object
Required: No

---

**Response Syntax**

```json
{
}
```
"solutionArn": "string"
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response. The following data is returned in JSON format by the service.

**solutionArn (p. 178)**

The ARN of the solution.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.*:.*:.*

Errors

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**LimitExceededException**

The limit on the number of requests per second has been exceeded.

HTTP Status Code: 400

**ResourceAlreadyExistsException**

The specified resource already exists.

HTTP Status Code: 400

**ResourceInUseException**

The specified resource is in use.

HTTP Status Code: 400

**ResourceNotFoundException**

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
CreateSolutionVersion
Service: Amazon Personalize

Trains or retrains an active solution. A solution is created using the CreateSolution (p. 176) operation and must be in the ACTIVE state before calling CreateSolutionVersion. A new version of the solution is created every time you call this operation.

Status
A solution version can be in one of the following states:

• CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED

To get the status of the version, call DescribeSolutionVersion (p. 223). Wait until the status shows as ACTIVE before calling CreateCampaign.

If the status shows as CREATE FAILED, the response includes a failureReason key, which describes why the job failed.

Related APIs
• ListSolutionVersions (p. 253)
• DescribeSolutionVersion (p. 223)
• ListSolutions (p. 250)
• CreateSolution (p. 176)
• DescribeSolution (p. 220)
• DeleteSolution (p. 196)

Request Syntax

```
{
    "solutionArn": "string",
    "trainingMode": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

solutionArn (p. 181)

The Amazon Resource Name (ARN) of the solution containing the training configuration information.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.+

Required: Yes

trainingMode (p. 181)

The scope of training to be performed when creating the solution version. The FULL option trains the solution version based on the entirety of the input solution's training data, while the UPDATE
option processes only the data that has changed in comparison to the input solution. Choose
UPDATE when you want to incrementally update your solution version instead of creating an entirely
new one.

**Important**

The UPDATE option can only be used when you already have an active solution version
created from the input solution using the FULL option and the input solution was trained
with the User-Personalization Recipe (p. 66) or the HRNN-Coldstart Recipe (Legacy) (p. 81)
recipe.

Type: String

Valid Values: FULL | UPDATE

Required: No

**Response Syntax**

```json
{
  "solutionVersionArn": "string"
}
```

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**solutionVersionArn (p. 182)**

The ARN of the new solution version.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.*`  

**Errors**

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**LimitExceededException**

The limit on the number of requests per second has been exceeded.

HTTP Status Code: 400

**ResourceInUseException**

The specified resource is in use.

HTTP Status Code: 400

**ResourceNotFoundException**

Could not find the specified resource.
HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DeleteCampaign
Service: Amazon Personalize

Removes a campaign by deleting the solution deployment. The solution that the campaign is based on is not deleted and can be redeployed when needed. A deleted campaign can no longer be specified in a GetRecommendations request. For more information on campaigns, see CreateCampaign (p. 157).

Request Syntax

```
{
  "campaignArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

campaignArn (p. 184)

The Amazon Resource Name (ARN) of the campaign to delete.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.+:.*:*:*`

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

ResourceInUseException

The specified resource is in use.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
• AWS SDK for .NET
• AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V3
DeleteDataset
Service: Amazon Personalize

Deletes a dataset. You can't delete a dataset if an associated DatasetImportJob or SolutionVersion is in the CREATE PENDING or IN PROGRESS state. For more information on datasets, see CreateDataset (p. 160).

Request Syntax

```
{
  "datasetArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetArn (p. 186)

The Amazon Resource Name (ARN) of the dataset to delete.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.*:.*:*+

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

ResourceInUseException

The specified resource is in use.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
• AWS SDK for .NET
• AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V3
**DeleteDatasetGroup**  
Service: Amazon Personalize

Deletes a dataset group. Before you delete a dataset group, you must delete the following:

- All associated event trackers.
- All associated solutions.
- All datasets in the dataset group.

**Request Syntax**

```json
{
    "datasetGroupArn": "string"
}
```

**Request Parameters**

The request accepts the following data in JSON format.

**datasetGroupArn (p. 188)**

The ARN of the dataset group to delete.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.*:.*:.+

Required: Yes

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

**Errors**

- **InvalidInputException**
  
  Provide a valid value for the field or parameter.

  HTTP Status Code: 400

- **ResourceInUseException**
  
  The specified resource is in use.

  HTTP Status Code: 400

- **ResourceNotFoundException**
  
  Could not find the specified resource.

  HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:
- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DeleteEventTracker
Service: Amazon Personalize

Deletes the event tracker. Does not delete the event-interactions dataset from the associated dataset group. For more information on event trackers, see CreateEventTracker (p. 169).

Request Syntax

```json
{
  "eventTrackerArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

**eventTrackerArn (p. 190)**

The Amazon Resource Name (ARN) of the event tracker to delete.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-zA-Z0-9\d\-]+):personalize:.+:.*:.+`

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**ResourceInUseException**

The specified resource is in use.

HTTP Status Code: 400

**ResourceNotFoundException**

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DeleteFilter
Service: Amazon Personalize

Deletes a filter.

Request Syntax

```json
{
  "filterArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

`filterArn (p. 192)`

The ARN of the filter to delete.

Type: String

Length Constraints: Maximum length of 256.

Pattern: \arn:([a-z\d-]+):personalize:.+:.*:*:+

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

`InvalidInputException`

Provide a valid value for the field or parameter.

HTTP Status Code: 400

`ResourceInUseException`

The specified resource is in use.

HTTP Status Code: 400

`ResourceNotFoundException`

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V3
DeleteSchema
Service: Amazon Personalize

Deletes a schema. Before deleting a schema, you must delete all datasets referencing the schema. For more information on schemas, see CreateSchema (p. 174).

Request Syntax

```
{
  "schemaArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

**schemaArn (p. 194)**

The Amazon Resource Name (ARN) of the schema to delete.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.*:+

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**ResourceInUseException**

The specified resource is in use.

HTTP Status Code: 400

**ResourceNotFoundException**

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
• AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V3
DeleteSolution
Service: Amazon Personalize

Deletes all versions of a solution and the Solution object itself. Before deleting a solution, you must delete all campaigns based on the solution. To determine what campaigns are using the solution, call ListCampaigns (p. 231) and supply the Amazon Resource Name (ARN) of the solution. You can't delete a solution if an associated SolutionVersion is in the CREATE PENDING or IN PROGRESS state. For more information on solutions, see CreateSolution (p. 176).

Request Syntax

```json
{
    "solutionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

`solutionArn (p. 196)`

The ARN of the solution to delete.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.+:.*:.+`

Required: Yes

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

`InvalidInputException`

Provide a valid value for the field or parameter.

HTTP Status Code: 400

`ResourceInUseException`

The specified resource is in use.

HTTP Status Code: 400

`ResourceNotFoundException`

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:
- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeAlgorithm
Service: Amazon Personalize

Describes the given algorithm.

Request Syntax

```json
{
   "algorithmArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

**algorithmArn (p. 198)**

The Amazon Resource Name (ARN) of the algorithm to describe.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+.*+:*

Required: Yes

Response Syntax

```json
{
   "algorithm": {
      "algorithmArn": "string",
      "algorithmImage": {
         "dockerURI": "string",
         "name": "string"
      },
      "creationDateTime": number,
      "defaultHyperParameterRanges": {
         "categoricalHyperParameterRanges": [ ]
      }
   }
}
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

algorithm (p. 198)

A listing of the properties of the algorithm.

Type: Algorithm (p. 275) object

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeBatchInferenceJob
Service: Amazon Personalize

Gets the properties of a batch inference job including name, Amazon Resource Name (ARN), status, input and output configurations, and the ARN of the solution version used to generate the recommendations.

Request Syntax

```
{
  "batchInferenceJobArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

**batchInferenceJobArn (p. 200)**

The ARN of the batch inference job to describe.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.+:.*:.*`.

Required: Yes

Response Syntax

```
{
  "batchInferenceJob": {
    "batchInferenceJobArn": "string",
    "batchInferenceJobConfig": {
      "itemExplorationConfig": {
        "string": "string"
      }
    },
    "creationDateTime": number,
    "failureReason": "string",
    "filterArn": "string",
    "jobInput": {
      "s3DataSource": {
        "kmsKeyArn": "string",
        "path": "string"
      }
    },
    "jobName": "string",
    "jobOutput": {
      "s3DataDestination": {
        "kmsKeyArn": "string",
        "path": "string"
      }
    },
    "lastUpdatedDateTime": number,
    "numResults": number,
    "roleArn": "string",
    "solutionVersionArn": "string",
    "status": "string"
  }
}
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response. The following data is returned in JSON format by the service.

**batchInferenceJob (p. 200)**

Information on the specified batch inference job.

Type: BatchInferenceJob (p. 280) object

Errors

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**ResourceNotFoundException**

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
**DescribeCampaign**

Service: Amazon Personalize

Describes the given campaign, including its status.

A campaign can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

When the status is CREATE FAILED, the response includes the failureReason key, which describes why.

For more information on campaigns, see CreateCampaign (p. 157).

**Request Syntax**

```json
{
    "campaignArn": "string"
}
```

**Request Parameters**

The request accepts the following data in JSON format.

**campaignArn (p. 202)**

The Amazon Resource Name (ARN) of the campaign.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.*:.*:.+

Required: Yes

**Response Syntax**

```json
{
    "campaign": {
        "campaignArn": "string",
        "campaignConfig": {
            "itemExplorationConfig": {
                "string": "string"
            }
        },
        "creationDateTime": number,
        "failureReason": "string",
        "lastUpdatedDateTime": number,
        "latestCampaignUpdate": {
            "campaignConfig": {
                "itemExplorationConfig": {
                    "string": "string"
                }
            },
            "creationDateTime": number,
            "failureReason": "string",
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

campaign (p. 202)

The properties of the campaign.

Type: Campaign (p. 288) object

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeDataset
Service: Amazon Personalize

Describes the given dataset. For more information on datasets, see CreateDataset (p. 160).

Request Syntax

```json
{
   "datasetArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetArn (p. 204)
The Amazon Resource Name (ARN) of the dataset to describe.
Type: String
Length Constraints: Maximum length of 256.
Pattern: `arn:([a-z\d-]+):personalize:.+:.*:.*+.+
Required: Yes

Response Syntax

```json
{
   "dataset": {
      "creationDateTime": number,
      "datasetArn": "string",
      "datasetGroupArn": "string",
      "datasetType": "string",
      "lastUpdatedDateTime": number,
      "name": "string",
      "schemaArn": "string",
      "status": "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

dataset (p. 204)
A listing of the dataset's properties.
Type: Dataset (p. 297) object

Errors

InvalidInputException
Provide a valid value for the field or parameter.
HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeDatasetGroup
Service: Amazon Personalize

Describes the given dataset group. For more information on dataset groups, see CreateDatasetGroup (p. 163).

Request Syntax

```json
{
  "datasetGroupArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn (p. 206)

The Amazon Resource Name (ARN) of the dataset group to describe.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:(\[a-z\d-]+):personalize:.+:.+:.*+`

Required: Yes

Response Syntax

```json
{
  "datasetGroup": {
    "creationDateTime": number,
    "datasetGroupArn": "string",
    "failureReason": "string",
    "kmsKeyArn": "string",
    "lastUpdatedDateTime": number,
    "name": "string",
    "roleArn": "string",
    "status": "string"
  }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

datasetGroup (p. 206)

A listing of the dataset group's properties.

Type: DatasetGroup (p. 299) object
Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeDatasetImportJob
Service: Amazon Personalize

Describes the dataset import job created by CreateDatasetImportJob (p. 166), including the import job status.

Request Syntax

```json
{
    "datasetImportJobArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetImportJobArn (p. 208)

The Amazon Resource Name (ARN) of the dataset import job to describe.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:*:.*+.+

Required: Yes

Response Syntax

```json
{
    "datasetImportJob": {
        "creationDateTime": number,
        "datasetArn": "string",
        "datasetImportJobArn": "string",
        "dataSource": {
            "dataLocation": "string"
        },
        "failureReason": "string",
        "jobName": "string",
        "lastUpdatedDateTime": number,
        "roleArn": "string",
        "status": "string"
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

datasetImportJob (p. 208)

Information about the dataset import job, including the status.

The status is one of the following values:

- CREATE PENDING
• CREATE IN_PROGRESS
• ACTIVE
• CREATE FAILED

Type: DatasetImportJob (p. 303) object

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

• AWS Command Line Interface
• AWS SDK for .NET
• AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V3
DescribeEventTracker
Service: Amazon Personalize

Describes an event tracker. The response includes the `trackingId` and `status` of the event tracker. For more information on event trackers, see CreateEventTracker (p. 169).

Request Syntax

```
{
    "eventTrackerArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

**eventTrackerArn (p. 210)**

The Amazon Resource Name (ARN) of the event tracker to describe.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.+*.*+`

Required: Yes

Response Syntax

```
{
    "eventTracker": {
        "accountId": "string",
        "creationDateTime": number,
        "datasetGroupArn": "string",
        "eventTrackerArn": "string",
        "lastUpdatedDateTime": number,
        "name": "string",
        "status": "string",
        "trackingId": "string"
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**eventTracker (p. 210)**

An object that describes the event tracker.

Type: `EventTracker (p. 317)` object
Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeFeatureTransformation
Service: Amazon Personalize

Describes the given feature transformation.

Request Syntax

```
{
   "featureTransformationArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

featureTransformationArn (p. 212)

The Amazon Resource Name (ARN) of the feature transformation to describe.

Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.+:.*
Required: Yes

Response Syntax

```
{
   "featureTransformation": {
       "creationDateTime": number,
       "defaultParameters": {
           "string" : "string"
       },
       "featureTransformationArn": "string",
       "lastUpdatedDateTime": number,
       "name": "string",
       "status": "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

featureTransformation (p. 212)

A listing of the FeatureTransformation properties.

Type: FeatureTransformation (p. 321) object

Errors

InvalidInputException

Provide a valid value for the field or parameter.
HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeFilter
Service: Amazon Personalize
Describes a filter's properties.

Request Syntax

```
{
  "filterArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

**filterArn (p. 214)**

- The ARN of the filter to describe.
- Type: String
- Length Constraints: Maximum length of 256.
- Pattern: `arn:([a-z\d-]+):personalize:.+:.*:*:`
- Required: Yes

Response Syntax

```
{
  "filter": {
    "creationDateTime": number,
    "datasetGroupArn": "string",
    "failureReason": "string",
    "filterArn": "string",
    "filterExpression": "string",
    "lastUpdatedDateTime": number,
    "name": "string",
    "status": "string"
  }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**filter (p. 214)**

- The filter's details.
- Type: `Filter (p. 323)` object

Errors

**InvalidInputException**

- Provide a valid value for the field or parameter.
HTTP Status Code: 400

**ResourceNotFoundException**

Could not find the specified resource.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeRecipe
Service: Amazon Personalize

Describes a recipe.

A recipe contains three items:

- An algorithm that trains a model.
- Hyperparameters that govern the training.
- Feature transformation information for modifying the input data before training.

Amazon Personalize provides a set of predefined recipes. You specify a recipe when you create a solution with the CreateSolution (p. 176) API. CreateSolution trains a model by using the algorithm in the specified recipe and a training dataset. The solution, when deployed as a campaign, can provide recommendations using the GetRecommendations API.

Request Syntax

```json
{
    "recipeArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

**recipeArn (p. 216)**

The Amazon Resource Name (ARN) of the recipe to describe.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:[a-z\d-]+:personalize:.+:.*

Required: Yes

Response Syntax

```json
{
    "recipe": {
        "algorithmArn": "string",
        "creationDateTime": number,
        "description": "string",
        "featureTransformationArn": "string",
        "lastUpdatedDateTime": number,
        "name": "string",
        "recipeArn": "string",
        "recipeType": "string",
        "status": "string"
    }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.
The following data is returned in JSON format by the service.

**recipe (p. 216)**

An object that describes the recipe.

Type: Recipe (p. 332) object

**Errors**

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**ResourceNotFoundException**

Could not find the specified resource.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeSchema
Service: Amazon Personalize

Describes a schema. For more information on schemas, see CreateSchema (p. 174).

Request Syntax

```json
{
   "schemaArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

**schemaArn (p. 218)**

The Amazon Resource Name (ARN) of the schema to retrieve.

- Type: String
- Length Constraints: Maximum length of 256.
- Pattern: arn:([a-z\d-]+):personalize:.*:.*:+
- Required: Yes

Response Syntax

```json
{
   "schema": {
      "creationDateTime": number,
      "lastUpdatedDateTime": number,
      "name": "string",
      "schema": "string",
      "schemaArn": "string"
   }
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**schema (p. 218)**

The requested schema.

- Type: DatasetSchema (p. 307) object

Errors

**InvalidInputException**

Provide a valid value for the field or parameter.
HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeSolution
Service: Amazon Personalize

Describes a solution. For more information on solutions, see CreateSolution (p. 176).

Request Syntax

```json
{
  "solutionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

**solutionArn (p. 220)**

The Amazon Resource Name (ARN) of the solution to describe.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.+:*.+:.*`

Required: Yes

Response Syntax

```json
{
  "solution": {
    "autoMLResult": {
      "bestRecipeArn": "string"
    },
    "creationDateTime": number,
    "datasetGroupArn": "string",
    "eventType": "string",
    "lastUpdatedDateTime": number,
    "latestSolutionVersion": {
      "creationDateTime": number,
      "failureReason": "string",
      "lastUpdatedDateTime": number,
      "solutionVersionArn": "string",
      "status": "string"
    },
    "name": "string",
    "performAutoML": boolean,
    "performHPO": boolean,
    "recipeArn": "string",
    "solutionArn": "string",
    "solutionConfig": {
      "algorithmHyperParameters": {
        "string": "string"
      },
      "autoMLConfig": {
        "metricName": "string",
        "recipeList": [ "string" ]
      },
      "eventValueThreshold": "string",
      "featureTransformationParameters": {
```
"string" : "string"
},
"hpoConfig": {
  "algorithmHyperParameterRanges": {
    "categoricalHyperParameterRanges": [
      {
        "name": "string",
        "values": [ "string" ]
      }
    ],
    "continuousHyperParameterRanges": [
      {
        "maxValue": number,
        "minValue": number,
        "name": "string"
      }
    ],
    "integerHyperParameterRanges": [
      {
        "maxValue": number,
        "minValue": number,
        "name": "string"
      }
    ]
  },
  "hpoObjective": {
    "metricName": "string",
    "metricRegex": "string",
    "type": "string"
  },
  "hpoResourceConfig": {
    "maxNumberOfTrainingJobs": "string",
    "maxParallelTrainingJobs": "string"
  }
},
"status": "string"
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

solution (p. 220)

An object that describes the solution.

Type: Solution (p. 337) object

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.
HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
DescribeSolutionVersion
Service: Amazon Personalize

Describes a specific version of a solution. For more information on solutions, see CreateSolution (p. 176).

Request Syntax

```json
{
    "solutionVersionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

**solutionVersionArn (p. 223)**

The Amazon Resource Name (ARN) of the solution version.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.+:.+

Required: Yes

Response Syntax

```json
{
    "solutionVersion": {
        "creationDateTime": number,
        "datasetGroupArn": "string",
        "eventType": "string",
        "failureReason": "string",
        "lastUpdatedDateTime": number,
        "performAutoML": boolean,
        "performHPO": boolean,
        "recipeArn": "string",
        "solutionArn": "string",
        "solutionConfig": {
            "algorithmHyperParameters": {
                "string": "string"
            },
            "autoMLConfig": {
                "metricName": "string",
                "recipeList": [ "string" ]
            },
            "eventValueThreshold": "string",
            "featureTransformationParameters": {
                "string": "string"
            },
            "hpoConfig": {
                "algorithmHyperParameterRanges": {
                    "categoricalHyperParameterRanges": [
                        { "name": "string", "values": [ "string" ] }
                    ]
                }
            }
        }
    }
}
```
"continuousHyperParameterRanges": [  
  {
    "maxValue": number,
    "minValue": number,
    "name": "string"
  }
],
"integerHyperParameterRanges": [  
  {
    "maxValue": number,
    "minValue": number,
    "name": "string"
  }
],
"hpoObjective": {  
  "metricName": "string",
  "metricRegex": "string",
  "type": "string"
},
"hpoResourceConfig": {  
  "maxNumberOfTrainingJobs": "string",
  "maxParallelTrainingJobs": "string"
},
"solutionVersionArn": "string",
"status": "string",
"trainingHours": number,
"trainingMode": "string",
"tunedHPOParams": {
  "algorithmHyperParameters": {
    "string": "string"
  }
}
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**solutionVersion (p. 223)**

The solution version.

Type: SolutionVersion (p. 344) object

Errors

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**ResourceNotFoundException**

Could not find the specified resource.

HTTP Status Code: 400
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
GetSolutionMetrics
Service: Amazon Personalize

Gets the metrics for the specified solution version.

Request Syntax

```
{
    "solutionVersionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

solutionVersionArn (p. 226)

The Amazon Resource Name (ARN) of the solution version for which to get metrics.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.*+.*

Required: Yes

Response Syntax

```
{
    "metrics": {
        "string": number
    },
    "solutionVersionArn": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

metrics (p. 226)

The metrics for the solution version.

Type: String to double map

Map Entries: Maximum number of 100 items.

Key Length Constraints: Maximum length of 256.

solutionVersionArn (p. 226)

The same solution version ARN as specified in the request.

Type: String

Length Constraints: Maximum length of 256.
Pattern: `arn:([a-z\d-]+):personalize:.+:.*`.

**Errors**

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**ResourceInUseException**

The specified resource is in use.

HTTP Status Code: 400

**ResourceNotFoundException**

Could not find the specified resource.

HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
**ListBatchInferenceJobs**  
Service: Amazon Personalize

Gets a list of the batch inference jobs that have been performed off of a solution version.

**Request Syntax**

```json
{
    "maxResults": number,
    "nextToken": "string",
    "solutionVersionArn": "string"
}
```

**Request Parameters**

The request accepts the following data in JSON format.

**maxResults (p. 228)**

The maximum number of batch inference job results to return in each page. The default value is 100.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

**nextToken (p. 228)**

The token to request the next page of results.

Type: String

Length Constraints: Maximum length of 1300.

Required: No

**solutionVersionArn (p. 228)**

The Amazon Resource Name (ARN) of the solution version from which the batch inference jobs were created.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+.*:.+

Required: No

**Response Syntax**

```json
{
    "batchInferenceJobs": [
        {
            "batchInferenceJobArn": "string",
            "creationDateTime": number,
            "failureReason": "string",
            "jobName": "string",
            "lastUpdatedDateTime": number,
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

batchInferenceJobs (p. 228)

A list containing information on each job that is returned.

Type: Array of BatchInferenceJobSummary (p. 286) objects

Array Members: Maximum number of 100 items.

nextToken (p. 228)

The token to use to retrieve the next page of results. The value is null when there are no more results to return.

Type: String

Length Constraints: Maximum length of 1300.

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

InvalidNextTokenException

The token is not valid.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListCampaigns
Service: Amazon Personalize

Returns a list of campaigns that use the given solution. When a solution is not specified, all the campaigns associated with the account are listed. The response provides the properties for each campaign, including the Amazon Resource Name (ARN). For more information on campaigns, see CreateCampaign (p. 157).

Request Syntax

```
{
    "maxResults": number,
    "nextToken": "string",
    "solutionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

maxResults (p. 231)

The maximum number of campaigns to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken (p. 231)

A token returned from the previous call to ListCampaigns for getting the next set of campaigns (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Required: No

solutionArn (p. 231)

The Amazon Resource Name (ARN) of the solution to list the campaigns for. When a solution is not specified, all the campaigns associated with the account are listed.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.+:.*:*:`

Required: No

Response Syntax

```
{
    "campaigns": [
        {
            "campaignArn": "string",
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

campaigns (p. 231)
A list of the campaigns.
Type: Array of CampaignSummary (p. 291) objects
Array Members: Maximum number of 100 items.

nextToken (p. 231)
A token for getting the next set of campaigns (if they exist).
Type: String
Length Constraints: Maximum length of 1300.

Errors

InvalidInputException
Provide a valid value for the field or parameter.
HTTP Status Code: 400

InvalidNextTokenException
The token is not valid.
HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListDatasetGroups
Service: Amazon Personalize

Returns a list of dataset groups. The response provides the properties for each dataset group, including the Amazon Resource Name (ARN). For more information on dataset groups, see CreateDatasetGroup (p. 163).

Request Syntax

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

Request Parameters

The request accepts the following data in JSON format.

**maxResults (p. 234)**

The maximum number of dataset groups to return.

- Type: Integer
- Valid Range: Minimum value of 1. Maximum value of 100.
- Required: No

**nextToken (p. 234)**

A token returned from the previous call to ListDatasetGroups for getting the next set of dataset groups (if they exist).

- Type: String
- Length Constraints: Maximum length of 1300.
- Required: No

Response Syntax

```json
{
    "datasetGroups": [
        {
            "creationDateTime": number,
            "datasetGroupArn": "string",
            "failureReason": "string",
            "lastUpdatedDateTime": number,
            "name": "string",
            "status": "string"
        }
    ],
    "nextToken": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.
The following data is returned in JSON format by the service.

datasetGroups (p. 234)

The list of your dataset groups.

Type: Array of DatasetGroupSummary (p. 301) objects

Array Members: Maximum number of 100 items.

nextToken (p. 234)

A token for getting the next set of dataset groups (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Errors

InvalidNextTokenException

The token is not valid.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
**ListDatasetImportJobs**  
Service: Amazon Personalize

Returns a list of dataset import jobs that use the given dataset. When a dataset is not specified, all the dataset import jobs associated with the account are listed. The response provides the properties for each dataset import job, including the Amazon Resource Name (ARN). For more information on dataset import jobs, see `CreateDatasetImportJob (p. 166)`. For more information on datasets, see `CreateDataset (p. 160)`.

**Request Syntax**

```json
{
    "datasetArn": "string",
    "maxResults": number,
    "nextToken": "string"
}
```

**Request Parameters**

The request accepts the following data in JSON format.

**datasetArn (p. 236)**

The Amazon Resource Name (ARN) of the dataset to list the dataset import jobs for.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:(\[a-z\d-]+):personalize:.+:.*+`  

Required: No

**maxResults (p. 236)**

The maximum number of dataset import jobs to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

**nextToken (p. 236)**

A token returned from the previous call to `ListDatasetImportJobs` for getting the next set of dataset import jobs (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Required: No

**Response Syntax**

```json
{
    "datasetImportJobs": [
        {
            "creationDateTime": number,
        }
    ]
}
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**datasetImportJobs (p. 236)**

The list of dataset import jobs.

Type: Array of DatasetImportJobSummary (p. 305) objects

Array Members: Maximum number of 100 items.

**nextToken (p. 236)**

A token for getting the next set of dataset import jobs (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Errors

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**InvalidNextTokenException**

The token is not valid.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListDatasets
Service: Amazon Personalize

Returns the list of datasets contained in the given dataset group. The response provides the properties for each dataset, including the Amazon Resource Name (ARN). For more information on datasets, see CreateDataset (p. 160).

Request Syntax

```
{
  "datasetGroupArn": "string",
  "maxResults": number,
  "nextToken": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

**datasetGroupArn (p. 239)**

The Amazon Resource Name (ARN) of the dataset group that contains the datasets to list.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+.*+.+.*+

Required: No

**maxResults (p. 239)**

The maximum number of datasets to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

**nextToken (p. 239)**

A token returned from the previous call to ListDatasetImportJobs for getting the next set of dataset import jobs (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Required: No

Response Syntax

```
{
  "datasets": [
    {
      "creationDateTime": number,
      "datasetArn": "string",
      "datasetType": "string",
    }
  ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response. The following data is returned in JSON format by the service.

datasets (p. 239)

An array of Dataset objects. Each object provides metadata information.

Type: Array of DatasetSummary (p. 310) objects

Array Members: Maximum number of 100 items.

nextToken (p. 239)

A token for getting the next set of datasets (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

InvalidNextTokenException

The token is not valid.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListEventTrackers
Service: Amazon Personalize

Returns the list of event trackers associated with the account. The response provides the properties for each event tracker, including the Amazon Resource Name (ARN) and tracking ID. For more information on event trackers, see CreateEventTracker (p. 169).

Request Syntax

```
{
  "datasetGroupArn": "string",
  "maxResults": number,
  "nextToken": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn (p. 242)

The ARN of a dataset group used to filter the response.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.*:+

Required: No

maxResults (p. 242)

The maximum number of event trackers to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken (p. 242)

A token returned from the previous call to ListEventTrackers for getting the next set of event trackers (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Required: No

Response Syntax

```
{
  "eventTrackers": [  
    {  
        "creationDateTime": number,
        "eventTrackerArn": "string",
        "lastUpdatedDateTime": number,
    
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**eventTrackers (p. 242)**

A list of event trackers.

Type: Array of **EventTrackerSummary (p. 319)** objects

Array Members: Maximum number of 100 items.

**nextToken (p. 242)**

A token for getting the next set of event trackers (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Errors

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**InvalidNextTokenException**

The token is not valid.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
**ListFilters**  
Service: Amazon Personalize

Lists all filters that belong to a given dataset group.

**Request Syntax**

```
{
  "datasetGroupArn": "string",
  "maxResults": number,
  "nextToken": "string"
}
```

**Request Parameters**

The request accepts the following data in JSON format.

**datasetGroupArn (p. 244)**

The ARN of the dataset group that contains the filters.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.+`

Required: No

**maxResults (p. 244)**

The maximum number of filters to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

**nextToken (p. 244)**

A token returned from the previous call to ListFilters for getting the next set of filters (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Required: No

**Response Syntax**

```
{
  "Filters": [
    {
      "creationDateTime": number,
      "datasetGroupArn": "string",
      "failureReason": "string",
      "filterArn": "string",
      "lastUpdatedDateTime": number,
    }
  ]
}
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

Filters (p. 244)

A list of returned filters.

Type: Array of FilterSummary (p. 325) objects

Array Members: Maximum number of 100 items.

nextToken (p. 244)

A token for getting the next set of filters (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

InvalidNextTokenException

The token is not valid.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListRecipes
Service: Amazon Personalize

Returns a list of available recipes. The response provides the properties for each recipe, including the recipe's Amazon Resource Name (ARN).

Request Syntax

```json
{
  "maxResults": number,
  "nextToken": "string",
  "recipeProvider": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

**maxResults (p. 246)**

The maximum number of recipes to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

**nextToken (p. 246)**

A token returned from the previous call to ListRecipes for getting the next set of recipes (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Required: No

**recipeProvider (p. 246)**

The default is SERVICE.

Type: String

Valid Values: SERVICE

Required: No

Response Syntax

```json
{
  "nextToken": "string",
  "recipes": [
    {
      "creationDateTime": number,
      "lastUpdatedDateTime": number,
      "name": "string",
      "recipeArn": "string",
      "status": "string"
    }
  ]
}
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**nextToken (p. 246)**

A token for getting the next set of recipes.

Type: String

Length Constraints: Maximum length of 1300.

**recipes (p. 246)**

The list of available recipes.

Type: Array of RecipeSummary (p. 334) objects

Array Members: Maximum number of 100 items.

Errors

**InvalidNextTokenException**

The token is not valid.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListSchemas
Service: Amazon Personalize

Returns the list of schemas associated with the account. The response provides the properties for each schema, including the Amazon Resource Name (ARN). For more information on schemas, see CreateSchema (p. 174).

Request Syntax

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

Request Parameters

The request accepts the following data in JSON format.

maxResults (p. 248)

The maximum number of schemas to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken (p. 248)

A token returned from the previous call to ListSchemas for getting the next set of schemas (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Required: No

Response Syntax

```json
{
    "nextToken": "string",
    "schemas": [
        {
            "creationDateTime": number,
            "lastUpdatedDateTime": number,
            "name": "string",
            "schemaArn": "string"
        }
    ]
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.
nextToken (p. 248)

A token used to get the next set of schemas (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

schemas (p. 248)

A list of schemas.

Type: Array of DatasetSchemaSummary (p. 309) objects

Array Members: Maximum number of 100 items.

Errors

InvalidNextTokenException

The token is not valid.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

* AWS Command Line Interface
* AWS SDK for .NET
* AWS SDK for C++
* AWS SDK for Go
* AWS SDK for Java
* AWS SDK for JavaScript
* AWS SDK for PHP V3
* AWS SDK for Python
* AWS SDK for Ruby V3
ListSolutions
Service: Amazon Personalize

Returns a list of solutions that use the given dataset group. When a dataset group is not specified, all the solutions associated with the account are listed. The response provides the properties for each solution, including the Amazon Resource Name (ARN). For more information on solutions, see CreateSolution (p. 176).

Request Syntax

```json
{
   "datasetGroupArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn (p. 250)

The Amazon Resource Name (ARN) of the dataset group.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.*:.*

Required: No

maxResults (p. 250)

The maximum number of solutions to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken (p. 250)

A token returned from the previous call to ListSolutions for getting the next set of solutions (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Required: No

Response Syntax

```json
{
   "nextToken": "string",
   "solutions": [ 
   
   }
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**nextToken (p. 250)**

A token for getting the next set of solutions (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

**solutions (p. 250)**

A list of the current solutions.

Type: Array of SolutionSummary (p. 342) objects

Array Members: Maximum number of 100 items.

Errors

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**InvalidNextTokenException**

The token is not valid.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
ListSolutionVersions
Service: Amazon Personalize

Returns a list of solution versions for the given solution. When a solution is not specified, all the solution versions associated with the account are listed. The response provides the properties for each solution version, including the Amazon Resource Name (ARN). For more information on solutions, see CreateSolution (p. 176).

Request Syntax

```json
{
  "maxResults": number,
  "nextToken": "string",
  "solutionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

**maxResults (p. 253)**

The maximum number of solution versions to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

**nextToken (p. 253)**

A token returned from the previous call to ListSolutionVersions for getting the next set of solution versions (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Required: No

**solutionArn (p. 253)**

The Amazon Resource Name (ARN) of the solution.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.*`

Required: No

Response Syntax

```json
{
  "nextToken": "string",
  "solutionVersions": [
    {
      "creationDateTime": number,
      "solutionArn": "string"
    }
  ]
}
```
Response Elements

If the action is successful, the service sends back an HTTP 200 response. The following data is returned in JSON format by the service.

**nextToken (p. 253)**

A token for getting the next set of solution versions (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

**solutionVersions (p. 253)**

A list of solution versions describing the version properties.

Type: Array of SolutionVersionSummary (p. 347) objects

Array Members: Maximum number of 100 items.

Errors

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**InvalidNextTokenException**

The token is not valid.

HTTP Status Code: 400

**ResourceNotFoundException**

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
• AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V3
UpdateCampaign
Service: Amazon Personalize

Updates a campaign by either deploying a new solution or changing the value of the campaign's `minProvisionedTPS` parameter.

To update a campaign, the campaign status must be ACTIVE or CREATE FAILED. Check the campaign status using the DescribeCampaign (p. 202) API.

**Note**
You must wait until the status of the updated campaign is ACTIVE before asking the campaign for recommendations.

For more information on campaigns, see CreateCampaign (p. 157).

Request Syntax

```json
{
    "campaignArn": "string",
    "campaignConfig": {
        "itemExplorationConfig": {
            "string": "string"
        }
    },
    "minProvisionedTPS": number,
    "solutionVersionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

**campaignArn (p. 256)**

The Amazon Resource Name (ARN) of the campaign.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.*+.+

Required: Yes

**campaignConfig (p. 256)**

The configuration details of a campaign.

Type: `CampaignConfig (p. 290)` object

Required: No

**minProvisionedTPS (p. 256)**

Specifies the requested minimum provisioned transactions (recommendations) per second that Amazon Personalize will support.

Type: Integer

Valid Range: Minimum value of 1.

Required: No
solutionVersionArn (p. 256)

The ARN of a new solution version to deploy.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.:+

Required: No

Response Syntax

{
   "campaignArn": "string"
}

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

campaignArn (p. 257)

The same campaign ARN as given in the request.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.:+

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

ResourceInUseException

The specified resource is in use.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
Amazon Personalize Events

The following actions are supported by Amazon Personalize Events:

- PutEvents (p. 259)
- PutItems (p. 261)
- PutUsers (p. 263)
PutEvents
Service: Amazon Personalize Events

Records user interaction event data. For more information see PutEvents Operation (p. 60).

Request Syntax

```
POST /events HTTP/1.1
Content-type: application/json

{
  "eventList": [
    {
      "eventId": "string",
      "eventType": "string",
      "eventValue": number,
      "impression": [ "string" ],
      "itemId": "string",
      "properties": "string",
      "recommendationId": "string",
      "sentAt": number
    }
  ],
  "sessionId": "string",
  "trackingId": "string",
  "userId": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

**eventList (p. 259)**

A list of event data from the session.

Type: Array of Event (p. 350) objects

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Required: Yes

**sessionId (p. 259)**

The session ID associated with the user's visit. Your application generates the sessionId when a user first visits your website or uses your application. Amazon Personalize uses the sessionId to associate events with the user before they log in. For more information see PutEvents Operation (p. 60).

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: Yes

**trackingId (p. 259)**

The tracking ID for the event. The ID is generated by a call to the CreateEventTracker API.
Type: String
Length Constraints: Minimum length of 1. Maximum length of 256.
Required: Yes

**userId (p. 259)**

The user associated with the event.
Type: String
Length Constraints: Minimum length of 1. Maximum length of 256.
Required: No

**Response Syntax**

```
HTTP/1.1 200
```

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

**Errors**

**InvalidInputException**

Provide a valid value for the field or parameter.
HTTP Status Code: 400

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
PutItems
Service: Amazon Personalize Events

Adds one or more items to an Items dataset. For more information see Importing Items Incrementally (p. 56).

Request Syntax

```
POST /items HTTP/1.1
Content-type: application/json
{
    "datasetArn": "string",
    "items": [
        {
            "itemId": "string",
            "properties": "string"
        }
    ]
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

datasetArn (p. 261)

The Amazon Resource Name (ARN) of the Items dataset you are adding the item or items to.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.*:.*

Required: Yes

items (p. 261)

A list of item data.

Type: Array of Item (p. 352) objects

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Required: Yes

Response Syntax

HTTP/1.1 200

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.
Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3
PutUsers
Service: Amazon Personalize Events

Adds one or more users to a Users dataset. For more information see Importing Users Incrementally (p. 55).

Request Syntax

POST /users HTTP/1.1
Content-type: application/json

{
   "datasetArn": "string",
   "users": [  
      {  
         "properties": "string",
         "userId": "string"
      }
   ]
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

datasetArn (p. 263)

The Amazon Resource Name (ARN) of the Users dataset you are adding the user or users to.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.*:.*:.*

Required: Yes

users (p. 263)

A list of user data.

Type: Array of User (p. 353) objects

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Required: Yes

Response Syntax

HTTP/1.1 200

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.
Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Amazon Personalize Runtime

The following actions are supported by Amazon Personalize Runtime:

- GetPersonalizedRanking (p. 265)
- GetRecommendations (p. 269)
GetPersonalizedRanking
Service: Amazon Personalize Runtime

Re-ranks a list of recommended items for the given user. The first item in the list is deemed the most likely item to be of interest to the user.

Note
The solution backing the campaign must have been created using a recipe of type PERSONALIZED_RANKING.

Request Syntax

POST /personalize-ranking HTTP/1.1
Content-type: application/json

{
    "campaignArn": "string",
    "context": {
        "string": "string"
    },
    "filterArn": "string",
    "filterValues": {
        "string": "string"
    },
    "inputList": [ "string" ],
    "userId": "string"
}

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

campaignArn (p. 265)

The Amazon Resource Name (ARN) of the campaign to use for generating the personalized ranking.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:+.+

Required: Yes

context (p. 265)

The contextual metadata to use when getting recommendations. Contextual metadata includes any interaction information that might be relevant when getting a user's recommendations, such as the user's current location or device type.

Type: String to string map

Map Entries: Maximum number of 150 items.

Key Length Constraints: Maximum length of 150.

Key Pattern: [A-Za-z\d_]+
Value Length Constraints: Maximum length of 1000.
Required: No

**filterArn (p. 265)**

The Amazon Resource Name (ARN) of a filter you created to include items or exclude items from recommendations for a given user. For more information, see Filtering Recommendations.

Type: String
Length Constraints: Maximum length of 256.
Pattern: `arn:([a-z\d-]+):personalize:.+:.

Required: No

**filterValues (p. 265)**

The values to use when filtering recommendations. For each placeholder parameter in your filter expression, provide the parameter name (in matching case) as a key and the filter value(s) as the corresponding value. Separate multiple values for one parameter with a comma.

For filter expressions that use an INCLUDE element to include items, you must provide values for all parameters that are defined in the expression. For filters with expressions that use an EXCLUDE element to exclude items, you can omit the filter-values. In this case, Amazon Personalize doesn't use that portion of the expression to filter recommendations.

For more information, see Filtering Recommendations.

Type: String to string map
Map Entries: Maximum number of 25 items.
Key Length Constraints: Maximum length of 50.
Key Pattern: `[A-Za-z0-9]+`
Value Length Constraints: Maximum length of 1000.
Required: No

**inputList (p. 265)**

A list of items (by `itemId`) to rank. If an item was not included in the training dataset, the item is appended to the end of the reranked list. The maximum is 500.

Type: Array of strings
Length Constraints: Maximum length of 256.
Required: Yes

**userId (p. 265)**

The user for which you want the campaign to provide a personalized ranking.

Type: String
Length Constraints: Maximum length of 256.
Required: Yes
Response Syntax

```json
HTTP/1.1 200
Content-type: application/json

{
    "personalizedRanking": [
        {
            "itemId": "string",
            "score": number
        }
    ],
    "recommendationId": "string"
}
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**personalizedRanking (p. 267)**

A list of items in order of most likely interest to the user. The maximum is 500.

Type: Array of PredictedItem (p. 354) objects

**recommendationId (p. 267)**

The ID of the recommendation.

Type: String

Errors

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400

**ResourceNotFoundException**

The specified resource does not exist.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
• AWS SDK for Python
• AWS SDK for Ruby V3
GetRecommendations
Service: Amazon Personalize Runtime

Returns a list of recommended items. The required input depends on the recipe type used to create the solution backing the campaign, as follows:

- RELATED_ITEMS - itemId required, userId not used
- USER_PERSONALIZATION - itemId optional, userId required

**Note**
Campaigns that are backed by a solution created using a recipe of type PERSONALIZED_RANKING use the GetPersonalizedRanking (p. 265) API.

**Request Syntax**

```json
POST /recommendations HTTP/1.1
Content-type: application/json

{}

"campaignArn": "string",
"context": {
  "string": "string"
},
"filterArn": "string",
"filterValues": {
  "string": "string"
},
"itemId": "string",
"numResults": number,
"userId": "string"
}
```

**URI Request Parameters**

The request does not use any URI parameters.

**Request Body**

The request accepts the following data in JSON format.

**campaignArn (p. 269)**

The Amazon Resource Name (ARN) of the campaign to use for getting recommendations.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+.*:.+

Required: Yes

**context (p. 269)**

The contextual metadata to use when getting recommendations. Contextual metadata includes any interaction information that might be relevant when getting a user's recommendations, such as the user's current location or device type.

Type: String to string map
Map Entries: Maximum number of 150 items.
Key Length Constraints: Maximum length of 150.
Key Pattern: [A-Za-z\d_]+
Value Length Constraints: Maximum length of 1000.
Required: No

**filterArn (p. 269)**

The ARN of the filter to apply to the returned recommendations. For more information, see Filtering Recommendations.

When using this parameter, be sure the filter resource is **ACTIVE**.

Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.+:.*:.+
Required: No

**filterValues (p. 269)**

The values to use when filtering recommendations. For each placeholder parameter in your filter expression, provide the parameter name (in matching case) as a key and the filter value(s) as the corresponding value. Separate multiple values for one parameter with a comma.

For filter expressions that use an **INCLUDE** element to include items, you must provide values for all parameters that are defined in the expression. For filters with expressions that use an **EXCLUDE** element to exclude items, you can omit the **filter-values**. In this case, Amazon Personalize doesn't use that portion of the expression to filter recommendations.

For more information, see Filtering Recommendations.

Type: String to string map
Map Entries: Maximum number of 25 items.
Key Length Constraints: Maximum length of 50.
Key Pattern: [A-Za-z0-9]+
Value Length Constraints: Maximum length of 1000.
Required: No

**itemId (p. 269)**

The item ID to provide recommendations for.
Required for **RELATED_ITEMS** recipe type.

Type: String
Length Constraints: Maximum length of 256.
Required: No

**numResults (p. 269)**

The number of results to return. The default is 25. The maximum is 500.
Type: Integer
Valid Range: Minimum value of 0.
Required: No

**userId (p. 269)**

The user ID to provide recommendations for.

Required for USER_PERSONALIZATION recipe type.

Type: String
Length Constraints: Maximum length of 256.
Required: No

**Response Syntax**

```
HTTP/1.1 200
Content-type: application/json

{
   "itemList": [
      {
         "itemId": "string",
         "score": number
      }
   ],
   "recommendationId": "string"
}
```

**Response Elements**

If the action is successful, the service sends back an HTTP 200 response.

The following data is returned in JSON format by the service.

**itemList (p. 271)**

A list of recommendations sorted in ascending order by prediction score. There can be a maximum of 500 items in the list.

Type: Array of PredictedItem (p. 354) objects

**recommendationId (p. 271)**

The ID of the recommendation.

Type: String

**Errors**

**InvalidInputException**

Provide a valid value for the field or parameter.

HTTP Status Code: 400
ResourceNotFoundException

The specified resource does not exist.

HTTP Status Code: 404

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

Data Types

The following data types are supported by Amazon Personalize:

- Algorithm (p. 275)
- AlgorithmImage (p. 277)
- AutoMLConfig (p. 278)
- AutoMLResult (p. 279)
- BatchInferenceJob (p. 280)
- BatchInferenceJobConfig (p. 283)
- BatchInferenceJobInput (p. 284)
- BatchInferenceJobOutput (p. 285)
- BatchInferenceJobSummary (p. 286)
- Campaign (p. 288)
- CampaignConfig (p. 290)
- CampaignSummary (p. 291)
- CampaignUpdateSummary (p. 293)
- CategoricalHyperParameterRange (p. 295)
- ContinuousHyperParameterRange (p. 296)
- Dataset (p. 297)
- DatasetGroup (p. 299)
- DatasetGroupSummary (p. 301)
- DatasetImportJob (p. 303)
- DatasetImportJobSummary (p. 305)
- DatasetSchema (p. 307)
- DatasetSchemaSummary (p. 309)
- DatasetSummary (p. 310)
The following data types are supported by Amazon Personalize Events:

- Event (p. 350)
- Item (p. 352)
- User (p. 353)

The following data types are supported by Amazon Personalize Runtime:

- PredictedItem (p. 354)

Amazon Personalize

The following data types are supported by Amazon Personalize:

- Algorithm (p. 275)
- AlgorithmImage (p. 277)
- AutoMLConfig (p. 278)
- AutoMLResult (p. 279)
- BatchInferenceJob (p. 280)
- BatchInferenceJobConfig (p. 283)
- BatchInferenceJobInput (p. 284)
- BatchInferenceJobOutput (p. 285)
• BatchInferenceJobSummary (p. 286)
• Campaign (p. 288)
• CampaignConfig (p. 290)
• CampaignSummary (p. 291)
• CampaignUpdateSummary (p. 293)
• CategoricalHyperParameterRange (p. 295)
• ContinuousHyperParameterRange (p. 296)
• Dataset (p. 297)
• DatasetGroup (p. 299)
• DatasetGroupSummary (p. 301)
• DatasetImportJob (p. 303)
• DatasetImportJobSummary (p. 305)
• DatasetSchema (p. 307)
• DatasetSchemaSummary (p. 309)
• DatasetSummary (p. 310)
• DataSource (p. 312)
• DefaultCategoricalHyperParameterRange (p. 313)
• DefaultContinuousHyperParameterRange (p. 314)
• DefaultHyperParameterRanges (p. 315)
• DefaultIntegerHyperParameterRange (p. 316)
• EventTracker (p. 317)
• EventTrackerSummary (p. 319)
• FeatureTransformation (p. 321)
• Filter (p. 323)
• FilterSummary (p. 325)
• HPOConfig (p. 327)
• HPOObjective (p. 328)
• HPOResourceConfig (p. 329)
• HyperParameterRanges (p. 330)
• IntegerHyperParameterRange (p. 331)
• Recipe (p. 332)
• RecipeSummary (p. 334)
• S3DataConfig (p. 336)
• Solution (p. 337)
• SolutionConfig (p. 340)
• SolutionSummary (p. 342)
• SolutionVersion (p. 344)
• SolutionVersionSummary (p. 347)
• TunedHPOParams (p. 349)
Algorithm
Service: Amazon Personalize
Describes a custom algorithm.

Contents

algorithmArn
The Amazon Resource Name (ARN) of the algorithm.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.+.*+.+
Required: No

algorithmImage
The URI of the Docker container for the algorithm image.
Type: AlgorithmImage (p. 277) object
Required: No

creationDateTime
The date and time (in Unix time) that the algorithm was created.
Type: Timestamp
Required: No

defaultHyperParameterRanges
Specifies the default hyperparameters, their ranges, and whether they are tunable. A tunable hyperparameter can have its value determined during hyperparameter optimization (HPO).
Type: DefaultHyperParameterRanges (p. 315) object
Required: No

defaultHyperParameters
Specifies the default hyperparameters.
Type: String to string map
Map Entries: Maximum number of 100 items.
Key Length Constraints: Maximum length of 256.
Value Length Constraints: Maximum length of 1000.
Required: No

defaultResourceConfig
Specifies the default maximum number of training jobs and parallel training jobs.
Type: String to string map
Map Entries: Maximum number of 100 items.
Key Length Constraints: Maximum length of 256.
Value Length Constraints: Maximum length of 1000.
Required: No

**lastUpdatedDateTime**
The date and time (in Unix time) that the algorithm was last updated.
Type: Timestamp
Required: No

**name**
The name of the algorithm.
Type: String
Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-\_]*$
Required: No

**roleArn**
The Amazon Resource Name (ARN) of the role.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-zA-Z\d-]):personalize:.+.*:.+.*
Required: No

**trainingInputMode**
The training input mode.
Type: String
Length Constraints: Maximum length of 256.
Required: No

**See Also**
For more information about using this API in one of the language-specific AWS SDKs, see the following:
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
**AlgorithmImage**
Service: Amazon Personalize

Describes an algorithm image.

**Contents**

**dockerURI**

The URI of the Docker container for the algorithm image.

Type: String
Length Constraints: Maximum length of 256.
Required: Yes

**name**

The name of the algorithm image.

Type: String
Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_.]*
Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
**AutoMLConfig**

Service: Amazon Personalize

When the solution performs AutoML (performAutoML is true in `CreateSolution (p. 176)`), Amazon Personalize determines which recipe, from the specified list, optimizes the given metric. Amazon Personalize then uses that recipe for the solution.

**Contents**

**metricName**

The metric to optimize.

Type: String

Length Constraints: Maximum length of 256.

Required: No

**recipeList**

The list of candidate recipes.

Type: Array of strings

Array Members: Maximum number of 100 items.

Length Constraints: Maximum length of 256.

Pattern: `arn:([-a-z\d-]+):personalize:.+:.+:.+`

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
AutoMLResult
Service: Amazon Personalize

When the solution performs AutoML (performAutoML is true in CreateSolution (p. 176)), specifies the recipe that best optimized the specified metric.

Contents

bestRecipeArn

The Amazon Resource Name (ARN) of the best recipe.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.*+:.*+

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
BatchInferenceJob
Service: Amazon Personalize

Contains information on a batch inference job.

Contents

batchInferenceJobArn
The Amazon Resource Name (ARN) of the batch inference job.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.+:.*+.+
Required: No

batchInferenceJobConfig
A string to string map of the configuration details of a batch inference job.
Type: BatchInferenceJobConfig (p. 283) object
Required: No

creationDateTime
The time at which the batch inference job was created.
Type: Timestamp
Required: No

failureReason
If the batch inference job failed, the reason for the failure.
Type: String
Required: No

filterArn
The ARN of the filter used on the batch inference job.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.+:.*+.+
Required: No

jobInput
The Amazon S3 path that leads to the input data used to generate the batch inference job.
Type: BatchInferenceJobInput (p. 284) object
Required: No

jobName
The name of the batch inference job.
Type: String
Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_\*]*
Required: No

**jobOutput**

The Amazon S3 bucket that contains the output data generated by the batch inference job.

Type: BatchInferenceJobOutput (p. 285) object
Required: No

**lastUpdatedDateTime**

The time at which the batch inference job was last updated.

Type: Timestamp
Required: No

**numResults**

The number of recommendations generated by the batch inference job. This number includes the error messages generated for failed input records.

Type: Integer
Required: No

**roleArn**

The ARN of the Amazon Identity and Access Management (IAM) role that requested the batch inference job.

Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:\([\(a-z\d-]+):iam::\d\(12\):role/?([a-zA-Z0-9=,.@\-_\/]+)

Required: No

**solutionVersionArn**

The Amazon Resource Name (ARN) of the solution version from which the batch inference job was created.

Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:\([\(a-z\d-]+\):personalize:.+:.*:.+:

Required: No

**status**

The status of the batch inference job. The status is one of the following values:

- PENDING
- IN PROGRESS
- ACTIVE
CREATE FAILED

Type: String

Length Constraints: Maximum length of 256.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
BatchInferenceJobConfig

Service: Amazon Personalize

The configuration details of a batch inference job.

Contents

itemExplorationConfig

A string to string map specifying the exploration configuration hyperparameters, including `explorationWeight` and `explorationItemAgeCutOff`, you want to use to configure the amount of item exploration Amazon Personalize uses when recommending items. See User-Personalization Recipe (p. 66).

Type: String to string map

Map Entries: Maximum number of 100 items.

Key Length Constraints: Maximum length of 256.

Value Length Constraints: Maximum length of 1000.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
BatchInferenceJobInput
Service: Amazon Personalize

The input configuration of a batch inference job.

Contents

s3DataSource

The URI of the Amazon S3 location that contains your input data. The Amazon S3 bucket must be in the same region as the API endpoint you are calling.

Type: S3DataConfig (p. 336) object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
BatchInferenceJobOutput
Service: Amazon Personalize

The output configuration parameters of a batch inference job.

Contents

s3DataDestination

Information on the Amazon S3 bucket in which the batch inference job's output is stored.

Type: S3DataConfig (p. 336) object

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
BatchInferenceJobSummary
Service: Amazon Personalize

A truncated version of the BatchInferenceJob (p. 280) datatype. The ListBatchInferenceJobs (p. 228) operation returns a list of batch inference job summaries.

Contents

batchInferenceJobArn

The Amazon Resource Name (ARN) of the batch inference job.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:(\[a-z\d-]+):personalize:.*:.*:.+

Required: No

creationDateTime

The time at which the batch inference job was created.

Type: Timestamp

Required: No

failureReason

If the batch inference job failed, the reason for the failure.

Type: String

Required: No

jobName

The name of the batch inference job.

Type: String


Pattern: `^[a-zA-Z0-9][a-zA-Z0-9\-\_]*$

Required: No

lastUpdatedDateTime

The time at which the batch inference job was last updated.

Type: Timestamp

Required: No

solutionVersionArn

The ARN of the solution version used by the batch inference job.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.\*:\*:+.+
Required: No

**status**

The status of the batch inference job. The status is one of the following values:
- PENDING
- IN PROGRESS
- ACTIVE
- CREATE FAILED

Type: String

Length Constraints: Maximum length of 256.

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
Campaign
Service: Amazon Personalize

Describes a deployed solution version, otherwise known as a campaign. For more information on campaigns, see CreateCampaign (p. 157).

Contents

campaignArn

The Amazon Resource Name (ARN) of the campaign.

Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.*:.*:.*
Required: No

campaignConfig

The configuration details of a campaign.

Type: CampaignConfig (p. 290) object
Required: No

creationDateTime

The date and time (in Unix format) that the campaign was created.

Type: Timestamp
Required: No

failureReason

If a campaign fails, the reason behind the failure.

Type: String
Required: No

lastUpdatedDateTime

The date and time (in Unix format) that the campaign was last updated.

Type: Timestamp
Required: No

latestCampaignUpdate

Provides a summary of the properties of a campaign update. For a complete listing, call the DescribeCampaign (p. 202) API.

Type: CampaignUpdateSummary (p. 293) object
Required: No

minProvisionedTPS

Specifies the requested minimum provisioned transactions (recommendations) per second.
Type: Integer
Valid Range: Minimum value of 1.
Required: No

name
The name of the campaign.
Type: String
Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*
Required: No

solutionVersionArn
The Amazon Resource Name (ARN) of a specific version of the solution.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([-a-z\d-]+):personalize:.+:.*
Required: No

status
The status of the campaign.
A campaign can be in one of the following states:
- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS
Type: String
Length Constraints: Maximum length of 256.
Required: No

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
**CampaignConfig**
Service: Amazon Personalize

The configuration details of a campaign.

**Contents**

**itemExplorationConfig**

A string to string map specifying the exploration configuration hyperparameters, including `explorationWeight` and `explorationItemAgeCutOff`, you want to use to configure the amount of item exploration Amazon Personalize uses when recommending items. See User-Personalization Recipe (p. 66).

Type: String to string map

Map Entries: Maximum number of 100 items.

Key Length Constraints: Maximum length of 256.

Value Length Constraints: Maximum length of 1000.

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
CampaignSummary
Service: Amazon Personalize

Provides a summary of the properties of a campaign. For a complete listing, call the DescribeCampaign (p. 202) API.

Contents

campaignArn

The Amazon Resource Name (ARN) of the campaign.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-zA-Z0-9-]+):personalize:.+:.*:.+
Required: No

creationDateTime

The date and time (in Unix time) that the campaign was created.
Type: Timestamp
Required: No

failureReason

If a campaign fails, the reason behind the failure.
Type: String
Required: No

lastUpdatedDateTime

The date and time (in Unix time) that the campaign was last updated.
Type: Timestamp
Required: No

name

The name of the campaign.
Type: String
Pattern: ^[^a-zA-Z0-9][a-zA-Z0-9-]*
Required: No

status

The status of the campaign.
A campaign can be in one of the following states:
• CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
• DELETE PENDING > DELETE IN_PROGRESS
Type: String

Length Constraints: Maximum length of 256.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
CampaignUpdateSummary
Service: Amazon Personalize

Provides a summary of the properties of a campaign update. For a complete listing, call the DescribeCampaign (p. 202) API.

Contents

campaignConfig

The configuration details of a campaign.
Type: CampaignConfig (p. 290) object
Required: No

creationDateTime

The date and time (in Unix time) that the campaign update was created.
Type: Timestamp
Required: No

failureReason

If a campaign update fails, the reason behind the failure.
Type: String
Required: No

lastUpdatedDateTime

The date and time (in Unix time) that the campaign update was last updated.
Type: Timestamp
Required: No

minProvisionedTPS

Specifies the requested minimum provisioned transactions (recommendations) per second that Amazon Personalize will support.
Type: Integer
Valid Range: Minimum value of 1.
Required: No

solutionVersionArn

The Amazon Resource Name (ARN) of the deployed solution version.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.+::
Required: No

status

The status of the campaign update.
A campaign update can be in one of the following states:
- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

Type: String

Length Constraints: Maximum length of 256.

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
CategoricalHyperParameterRange
Service: Amazon Personalize

Provides the name and range of a categorical hyperparameter.

Contents

name

The name of the hyperparameter.
Type: String
Length Constraints: Maximum length of 256.
Required: No

values

A list of the categories for the hyperparameter.
Type: Array of strings
Array Members: Maximum number of 100 items.
Length Constraints: Maximum length of 1000.
Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
ContinuousHyperParameterRange
Service: Amazon Personalize

Provides the name and range of a continuous hyperparameter.

Contents

**maxValue**

The maximum allowable value for the hyperparameter.

Type: Double

Valid Range: Minimum value of -1000000.

Required: No

**minValue**

The minimum allowable value for the hyperparameter.

Type: Double

Valid Range: Minimum value of -1000000.

Required: No

**name**

The name of the hyperparameter.

Type: String

Length Constraints: Maximum length of 256.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
Dataset
Service: Amazon Personalize

Provides metadata for a dataset.

Contents

creationDateTime
The creation date and time (in Unix time) of the dataset.
Type: Timestamp
Required: No

datasetArn
The Amazon Resource Name (ARN) of the dataset that you want metadata for.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([-a-z\d]+):personalize:.*:.*:+
Required: No

datasetGroupArn
The Amazon Resource Name (ARN) of the dataset group.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([-a-z\d]+):personalize:.*:.*:+
Required: No

datasetType
One of the following values:
• Interactions
• Items
• Users
Type: String
Length Constraints: Maximum length of 256.
Required: No

lastUpdatedDateTime
A time stamp that shows when the dataset was updated.
Type: Timestamp
Required: No

name
The name of the dataset.
Type: String
Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*
Required: No

schemaArn

The ARN of the associated schema.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.+:.*+.*+.*
Required: No

status

The status of the dataset.
A dataset can be in one of the following states:
• CREATE_PENDING > CREATE_IN_PROGRESS > ACTIVE -or- CREATE_FAILED
• DELETE_PENDING > DELETE_IN_PROGRESS

Type: String
Length Constraints: Maximum length of 256.
Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:
• AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for Ruby V3
DatasetGroup
Service: Amazon Personalize

A dataset group is a collection of related datasets (Interactions, User, and Item). You create a dataset group by calling `CreateDatasetGroup` (p. 163). You then create a dataset and add it to a dataset group by calling `CreateDataset` (p. 160). The dataset group is used to create and train a solution by calling `CreateSolution` (p. 176). A dataset group can contain only one of each type of dataset.

You can specify an AWS Key Management Service (KMS) key to encrypt the datasets in the group.

Contents

**creationDateTime**

The creation date and time (in Unix time) of the dataset group.

Type: Timestamp

Required: No

**datasetGroupArn**

The Amazon Resource Name (ARN) of the dataset group.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([^a-z\d-]+):personalize:.*:.*:.+`

Required: No

**failureReason**

If creating a dataset group fails, provides the reason why.

Type: String

Required: No

**kmsKeyArn**

The Amazon Resource Name (ARN) of the KMS key used to encrypt the datasets.

Type: String

Required: No

**lastUpdatedDateTime**

The last update date and time (in Unix time) of the dataset group.

Type: Timestamp

Required: No

**name**

The name of the dataset group.

Type: String


Pattern: `^[a-zA-Z0-9-\d]*$`
**roleArn**

The ARN of the IAM role that has permissions to create the dataset group.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_]+`

Required: No

**status**

The current status of the dataset group.

A dataset group can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING

Type: String

Length Constraints: Maximum length of 256.

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
DatasetGroupSummary
Service: Amazon Personalize

Provides a summary of the properties of a dataset group. For a complete listing, call the DescribeDatasetGroup (p. 206) API.

Contents

creationDateTime
The date and time (in Unix time) that the dataset group was created.
Type: Timestamp
Required: No
datasetGroupArn
The Amazon Resource Name (ARN) of the dataset group.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:[a-zA-Z\d-]*:personalize:.+.*:.*
Required: No
failureReason
If creating a dataset group fails, the reason behind the failure.
Type: String
Required: No
lastUpdatedDateTime
The date and time (in Unix time) that the dataset group was last updated.
Type: Timestamp
Required: No
name
The name of the dataset group.
Type: String
Pattern: ^[a-zA-Z\d-]{1,63}$
Required: No
status
The status of the dataset group.
A dataset group can be in one of the following states:
• CREATE PENDING > CREATE IN_PROGRESS > ACTIVE - or - CREATE FAILED
• DELETE PENDING
Type: String
Length Constraints: Maximum length of 256.
Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
DatasetImportJob
Service: Amazon Personalize

Describes a job that imports training data from a data source (Amazon S3 bucket) to an Amazon Personalize dataset. For more information, see CreateDatasetImportJob (p. 166).

A dataset import job can be in one of the following states:
• CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED

Contents

creationDateTime
  The creation date and time (in Unix time) of the dataset import job.
  Type: Timestamp
  Required: No

datasetArn
  The Amazon Resource Name (ARN) of the dataset that receives the imported data.
  Type: String
  Length Constraints: Maximum length of 256.
  Pattern: arn:([a-z\d-]+):personalize:.*:.*:+
  Required: No

datasetImportJobArn
  The ARN of the dataset import job.
  Type: String
  Length Constraints: Maximum length of 256.
  Pattern: arn:([a-z\d-]+):personalize:.*:.*:+
  Required: No

dataSource
  The Amazon S3 bucket that contains the training data to import.
  Type: DataSource (p. 312) object
  Required: No

failureReason
  If a dataset import job fails, provides the reason why.
  Type: String
  Required: No

jobName
  The name of the import job.
Type: String
Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*
Required: No

lastUpdatedDateTime
The date and time (in Unix time) the dataset was last updated.
Type: Timestamp
Required: No

roleArn
The ARN of the AWS Identity and Access Management (IAM) role that has permissions to read from the Amazon S3 data source.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:[\d-]+:personalize:.+:.*:.*
Required: No

status
The status of the dataset import job.
A dataset import job can be in one of the following states:
• CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
Type: String
Length Constraints: Maximum length of 256.
Required: No

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:
• AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for Ruby V3
## DatasetImportJobSummary

**Service:** Amazon Personalize

Provides a summary of the properties of a dataset import job. For a complete listing, call the DescribeDatasetImportJob (p. 208) API.

### Contents

**creationDateTime**

The date and time (in Unix time) that the dataset import job was created.

- **Type:** Timestamp
- **Required:** No

**datasetImportJobArn**

The Amazon Resource Name (ARN) of the dataset import job.

- **Type:** String
- **Length Constraints:** Maximum length of 256.
- **Pattern:** `arn:([a-z\d-]+):personalize:.+:.*:+

- **Required:** No

**failureReason**

If a dataset import job fails, the reason behind the failure.

- **Type:** String
- **Required:** No

**jobName**

The name of the dataset import job.

- **Type:** String
- **Length Constraints:** Minimum length of 1. Maximum length of 63.
- **Pattern:** `^[a-zA-Z0-9\-\_]*$`

- **Required:** No

**lastUpdatedDateTime**

The date and time (in Unix time) that the dataset was last updated.

- **Type:** Timestamp
- **Required:** No

**status**

The status of the dataset import job.

A dataset import job can be in one of the following states:
- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED

- **Type:** String

---

305
Length Constraints: Maximum length of 256.
Required: No

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
**DatasetSchema**
Service: Amazon Personalize

Describes the schema for a dataset. For more information on schemas, see *CreateSchema (p. 174).*

**Contents**

- **creationDateTime**
  - The date and time (in Unix time) that the schema was created.
  - Type: Timestamp
  - Required: No

- **lastUpdatedDateTime**
  - The date and time (in Unix time) that the schema was last updated.
  - Type: Timestamp
  - Required: No

- **name**
  - The name of the schema.
  - Type: String
  - Pattern: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`
  - Required: No

- **schema**
  - The schema.
  - Type: String
  - Length Constraints: Maximum length of 10000.
  - Required: No

- **schemaArn**
  - The Amazon Resource Name (ARN) of the schema.
  - Type: String
  - Length Constraints: Maximum length of 256.
  - Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.*`
  - Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
DatasetSchemaSummary
Service: Amazon Personalize

Provides a summary of the properties of a dataset schema. For a complete listing, call the DescribeSchema (p. 218) API.

Contents

- **creationDateTime**
  - The date and time (in Unix time) that the schema was created.
  - Type: Timestamp
  - Required: No

- **lastUpdatedDateTime**
  - The date and time (in Unix time) that the schema was last updated.
  - Type: Timestamp
  - Required: No

- **name**
  - The name of the schema.
  - Type: String
  - Pattern: ^[a-zA-Z0-9-\_]*$*\[a-zA-Z0-9-\_]*$
  - Required: No

- **schemaArn**
  - The Amazon Resource Name (ARN) of the schema.
  - Type: String
  - Length Constraints: Maximum length of 256.
  - Pattern: arn:\([a-z\d-]+\)\:\personalize:::*\.*\.*\.*\.
  - Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
DatasetSummary
Service: Amazon Personalize

Provides a summary of the properties of a dataset. For a complete listing, call the DescribeDataset (p. 204) API.

Contents

creationDateTime
   The date and time (in Unix time) that the dataset was created.
   Type: Timestamp
   Required: No

datasetArn
   The Amazon Resource Name (ARN) of the dataset.
   Type: String
   Length Constraints: Maximum length of 256.
   Pattern: arn:([a-z\d-]+):personalize:.*:.*:.*
   Required: No

datasetType
   The dataset type. One of the following values:
   • Interactions
   • Items
   • Users
   • Event-Interactions
   Type: String
   Length Constraints: Maximum length of 256.
   Required: No

lastUpdatedDateTime
   The date and time (in Unix time) that the dataset was last updated.
   Type: Timestamp
   Required: No

name
   The name of the dataset.
   Type: String
   Pattern: ^[a-zA-Z0-9]+[a-zA-Z0-9\-\_]*$
   Required: No
**status**

The status of the dataset.

A dataset can be in one of the following states:
- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

Type: String

Length Constraints: Maximum length of 256.

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
DataSource
Service: Amazon Personalize

Describes the data source that contains the data to upload to a dataset.

Contents

dataLocation

The path to the Amazon S3 bucket where the data that you want to upload to your dataset is stored. For example:

s3://bucket-name/training-data.csv

Type: String

Length Constraints: Maximum length of 256.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
DefaultCategoricalHyperParameterRange
Service: Amazon Personalize

Provides the name and default range of a categorical hyperparameter and whether the hyperparameter is tunable. A tunable hyperparameter can have its value determined during hyperparameter optimization (HPO).

Contents

isTunable

Whether the hyperparameter is tunable.
Type: Boolean
Required: No

name

The name of the hyperparameter.
Type: String
Length Constraints: Maximum length of 256.
Required: No

values

A list of the categories for the hyperparameter.
Type: Array of strings
Array Members: Maximum number of 100 items.
Length Constraints: Maximum length of 1000.
Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
DefaultContinuousHyperParameterRange
Service: Amazon Personalize

Provides the name and default range of a continuous hyperparameter and whether the hyperparameter is tunable. A tunable hyperparameter can have its value determined during hyperparameter optimization (HPO).

Contents

isTunable
Whether the hyperparameter is tunable.
Type: Boolean
Required: No

maxValue
The maximum allowable value for the hyperparameter.
Type: Double
Valid Range: Minimum value of -1000000.
Required: No

minValue
The minimum allowable value for the hyperparameter.
Type: Double
Valid Range: Minimum value of -1000000.
Required: No

name
The name of the hyperparameter.
Type: String
Length Constraints: Maximum length of 256.
Required: No

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
**DefaultHyperParameterRanges**

Service: Amazon Personalize

Specifies the hyperparameters and their default ranges. Hyperparameters can be categorical, continuous, or integer-valued.

**Contents**

**categoricalHyperParameterRanges**

The categorical hyperparameters and their default ranges.

Type: Array of `DefaultCategoricalHyperParameterRange (p. 313)` objects

Array Members: Maximum number of 100 items.

Required: No

**continuousHyperParameterRanges**

The continuous hyperparameters and their default ranges.

Type: Array of `DefaultContinuousHyperParameterRange (p. 314)` objects

Array Members: Maximum number of 100 items.

Required: No

**integerHyperParameterRanges**

The integer-valued hyperparameters and their default ranges.

Type: Array of `DefaultIntegerHyperParameterRange (p. 316)` objects

Array Members: Maximum number of 100 items.

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
DefaultIntegerHyperParameterRange
Service: Amazon Personalize

Provides the name and default range of a integer-valued hyperparameter and whether the hyperparameter is tunable. A tunable hyperparameter can have its value determined during hyperparameter optimization (HPO).

Contents

isTunable
 Indicates whether the hyperparameter is tunable.
 Type: Boolean
 Required: No

maxValue
 The maximum allowable value for the hyperparameter.
 Type: Integer
 Valid Range: Maximum value of 1000000.
 Required: No

minValue
 The minimum allowable value for the hyperparameter.
 Type: Integer
 Valid Range: Minimum value of -1000000.
 Required: No

name
 The name of the hyperparameter.
 Type: String
 Length Constraints: Maximum length of 256.
 Required: No

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
EventTracker
Service: Amazon Personalize

Provides information about an event tracker.

Contents

accountId
The Amazon AWS account that owns the event tracker.
Type: String
Length Constraints: Maximum length of 256.
Required: No

creationDateTime
The date and time (in Unix format) that the event tracker was created.
Type: Timestamp
Required: No

datasetGroupArn
The Amazon Resource Name (ARN) of the dataset group that receives the event data.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.*:.*:+
Required: No

eventTrackerArn
The ARN of the event tracker.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.*:.*:+
Required: No

lastUpdatedDateTime
The date and time (in Unix time) that the event tracker was last updated.
Type: Timestamp
Required: No

name
The name of the event tracker.
Type: String
Pattern: \^[a-zA-Z0-9][a-zA-Z0-9\-_]*

Required: No

**status**

The status of the event tracker.

An event tracker can be in one of the following states:
- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

Type: String

Length Constraints: Maximum length of 256.

Required: No

**trackingId**

The ID of the event tracker. Include this ID in requests to the PutEvents API.

Type: String

Length Constraints: Maximum length of 256.

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
EventTrackerSummary
Service: Amazon Personalize

Provides a summary of the properties of an event tracker. For a complete listing, call the DescribeEventTracker (p. 210) API.

Contents

creationDateTime

The date and time (in Unix time) that the event tracker was created.

Type: Timestamp

Required: No

eventTrackerArn

The Amazon Resource Name (ARN) of the event tracker.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.*:*:*+

Required: No

lastUpdatedDateTime

The date and time (in Unix time) that the event tracker was last updated.

Type: Timestamp

Required: No

name

The name of the event tracker.

Type: String


Pattern: ^[a-zA-Z0-9][a-zA-Z0-9-._]*

Required: No

status

The status of the event tracker.

An event tracker can be in one of the following states:
• CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
• DELETE PENDING > DELETE IN_PROGRESS

Type: String

Length Constraints: Maximum length of 256.

Required: No
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
FeatureTransformation
Service: Amazon Personalize

Provides feature transformation information. Feature transformation is the process of modifying raw input data into a form more suitable for model training.

Contents

creationDateTime
The creation date and time (in Unix time) of the feature transformation.
Type: Timestamp
Required: No
defaultParameters
Provides the default parameters for feature transformation.
Type: String to string map
Map Entries: Maximum number of 100 items.
Key Length Constraints: Maximum length of 256.
Value Length Constraints: Maximum length of 1000.
Required: No
featureTransformationArn
The Amazon Resource Name (ARN) of the FeatureTransformation object.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:\([a-z\d-]+\):personalize:.+\.:\.:\.:+
Required: No
lastUpdatedDateTime
The last update date and time (in Unix time) of the feature transformation.
Type: Timestamp
Required: No
name
The name of the feature transformation.
Type: String
Pattern: ^[a-zA-Z0-9-]a-zA-Z0-9\-\_]*$
Required: No
status
The status of the feature transformation.
A feature transformation can be in one of the following states:
• CREATE PENDING > CREATE IN_PROGRESS > ACTIVE - or - CREATE FAILED

Type: String

Length Constraints: Maximum length of 256.

Required: No

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:

• AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for Ruby V3
Filter
Service: Amazon Personalize

Contains information on a recommendation filter, including its ARN, status, and filter expression.

Contents

creationDateTime
- The time at which the filter was created.
- Type: Timestamp
- Required: No

datasetGroupArn
- The ARN of the dataset group to which the filter belongs.
- Type: String
- Length Constraints: Maximum length of 256.
- Pattern: arn:([a-z\d-]+):personalize:.+:.
- Required: No

failureReason
- If the filter failed, the reason for its failure.
- Type: String
- Required: No

filterArn
- The ARN of the filter.
- Type: String
- Length Constraints: Maximum length of 256.
- Pattern: arn:([a-z\d-]+):personalize:.+.*
- Required: No

filterExpression
- Specifies the type of item interactions to filter out of recommendation results. The filter expression must follow specific format rules. For information about filter expression structure and syntax, see Filter Expressions (p. 113).
- Type: String
- Required: No

lastUpdatedDateTime
- The time at which the filter was last updated.
- Type: Timestamp
name

The name of the filter.

Type: String


Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_.]*

Required: No

status

The status of the filter.

Type: String

Length Constraints: Maximum length of 256.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
FilterSummary
Service: Amazon Personalize

A short summary of a filter's attributes.

Contents

creationDateTime
The time at which the filter was created.
Type: Timestamp
Required: No

datasetGroupArn
The ARN of the dataset group to which the filter belongs.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.+:.*
Required: No

failureReason
If the filter failed, the reason for the failure.
Type: String
Required: No

filterArn
The ARN of the filter.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.+:.*
Required: No

lastUpdatedDateTime
The time at which the filter was last updated.
Type: Timestamp
Required: No

name
The name of the filter.
Type: String
Pattern: ^[a-zA-Z0-9\-\_]*$
Required: No

status

The status of the filter.

Type: String

Length Constraints: Maximum length of 256.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
HPOConfig
Service: Amazon Personalize

Describes the properties for hyperparameter optimization (HPO).

Contents
algorithmHyperParameterRanges
  The hyperparameters and their allowable ranges.
  Type: HyperParameterRanges (p. 330) object
  Required: No
hpoObjective
  The metric to optimize during HPO.
  Type: HPOObjective (p. 328) object
  Required: No
hpoResourceConfig
  Describes the resource configuration for HPO.
  Type: HPOResourceConfig (p. 329) object
  Required: No

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
HPOObjective
Service: Amazon Personalize

The metric to optimize during hyperparameter optimization (HPO).

Contents

metricName
The name of the metric.
Type: String
Length Constraints: Maximum length of 256.
Required: No

metricRegex
A regular expression for finding the metric in the training job logs.
Type: String
Length Constraints: Maximum length of 256.
Required: No

type
The type of the metric. Valid values are Maximize and Minimize.
Type: String
Length Constraints: Maximum length of 256.
Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
HPOResourceConfig
Service: Amazon Personalize

Describes the resource configuration for hyperparameter optimization (HPO).

Contents

maxNumberOfTrainingJobs

The maximum number of training jobs when you create a solution version. The maximum value for maxNumberOfTrainingJobs is 40.

Type: String
Length Constraints: Maximum length of 256.
Required: No

maxParallelTrainingJobs

The maximum number of parallel training jobs when you create a solution version. The maximum value for maxParallelTrainingJobs is 10.

Type: String
Length Constraints: Maximum length of 256.
Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
HyperParameterRanges
Service: Amazon Personalize

Specifies the hyperparameters and their ranges. Hyperparameters can be categorical, continuous, or integer-valued.

Contents

categoricalHyperParameterRanges

The categorical hyperparameters and their ranges.

Type: Array of CategoricalHyperParameterRange (p. 295) objects

Array Members: Maximum number of 100 items.

Required: No

continuousHyperParameterRanges

The continuous hyperparameters and their ranges.

Type: Array of ContinuousHyperParameterRange (p. 296) objects

Array Members: Maximum number of 100 items.

Required: No

integerHyperParameterRanges

The integer-valued hyperparameters and their ranges.

Type: Array of IntegerHyperParameterRange (p. 331) objects

Array Members: Maximum number of 100 items.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
**IntegerHyperParameterRange**

Service: Amazon Personalize

Provides the name and range of an integer-valued hyperparameter.

**Contents**

**maxValue**

The maximum allowable value for the hyperparameter.

Type: Integer

Valid Range: Maximum value of 1000000.

Required: No

**minValue**

The minimum allowable value for the hyperparameter.

Type: Integer

Valid Range: Minimum value of -1000000.

Required: No

**name**

The name of the hyperparameter.

Type: String

Length Constraints: Maximum length of 256.

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
Recipe
Service: Amazon Personalize

Provides information about a recipe. Each recipe provides an algorithm that Amazon Personalize uses in model training when you use the CreateSolution (p. 176) operation.

Contents

algorithmArn
  The Amazon Resource Name (ARN) of the algorithm that Amazon Personalize uses to train the model.
  Type: String
  Length Constraints: Maximum length of 256.
  Pattern: arn:([a-z\d-]+):personalize:.+:.*:+:+
  Required: No

creationDateTime
  The date and time (in Unix format) that the recipe was created.
  Type: Timestamp
  Required: No

description
  The description of the recipe.
  Type: String
  Required: No

featureTransformationArn
  The ARN of the FeatureTransformation object.
  Type: String
  Length Constraints: Maximum length of 256.
  Pattern: arn:([a-z\d-]+):personalize:.+:.*:+:+
  Required: No

lastUpdatedDateTime
  The date and time (in Unix format) that the recipe was last updated.
  Type: Timestamp
  Required: No

name
  The name of the recipe.
  Type: String
Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

Required: No

**recipeArn**

The Amazon Resource Name (ARN) of the recipe.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+.*:.*

Required: No

**recipeType**

One of the following values:

- PERSONALIZED_RANKING
- RELATED_ITEMS
- USER_PERSONALIZATION

Type: String

Length Constraints: Maximum length of 256.

Required: No

**status**

The status of the recipe.

Type: String

Length Constraints: Maximum length of 256.

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
RecipeSummary
Service: Amazon Personalize

Provides a summary of the properties of a recipe. For a complete listing, call the DescribeRecipe (p. 216) API.

Contents

creationDateTime
The date and time (in Unix time) that the recipe was created.
Type: Timestamp
Required: No

lastUpdatedDateTime
The date and time (in Unix time) that the recipe was last updated.
Type: Timestamp
Required: No

name
The name of the recipe.
Type: String
Pattern: ^[a-zA-Z0-9][a-zA-Z0-9-\_]*$
Required: No

recipeArn
The Amazon Resource Name (ARN) of the recipe.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.+:.*+.+
Required: No

status
The status of the recipe.
Type: String
Length Constraints: Maximum length of 256.
Required: No

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:

• AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for Ruby V3
S3DataConfig
Service: Amazon Personalize

The configuration details of an Amazon S3 input or output bucket.

Contents

kmsKeyArn

The Amazon Resource Name (ARN) of the Amazon Key Management Service (KMS) key that Amazon Personalize uses to encrypt or decrypt the input and output files of a batch inference job.

Type: String
Required: No

path

The file path of the Amazon S3 bucket.

Type: String
Length Constraints: Maximum length of 256.
Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
Solution
Service: Amazon Personalize

An object that provides information about a solution. A solution is a trained model that can be deployed as a campaign.

Contents

autoMLResult

When `performAutoML` is true, specifies the best recipe found.

Type: AutoMLResult (p. 279) object

Required: No

creationDateTime

The creation date and time (in Unix time) of the solution.

Type: Timestamp

Required: No

datasetGroupArn

The Amazon Resource Name (ARN) of the dataset group that provides the training data.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.+:.*::` +

Required: No

eventType

The event type (for example, 'click' or 'like') that is used for training the model. If no `eventType` is provided, Amazon Personalize uses all interactions for training with equal weight regardless of type.

Type: String

Length Constraints: Maximum length of 256.

Required: No

lastUpdatedDateTime

The date and time (in Unix time) that the solution was last updated.

Type: Timestamp

Required: No

latestSolutionVersion

Describes the latest version of the solution, including the status and the ARN.

Type: SolutionVersionSummary (p. 347) object

Required: No

name

The name of the solution.
Type: String
Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-\_]*$
Required: No

performAutoML

When true, Amazon Personalize performs a search for the best USER_PERSONALIZATION recipe from the list specified in the solution configuration (recipeArn must not be specified). When false (the default), Amazon Personalize uses recipeArn for training.

Type: Boolean
Required: No

performHPO

Whether to perform hyperparameter optimization (HPO) on the chosen recipe. The default is false.

Type: Boolean
Required: No

recipeArn

The ARN of the recipe used to create the solution.

Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.+:.*:.*
Required: No

solutionArn

The ARN of the solution.

Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.+:.*:.*
Required: No

solutionConfig

Describes the configuration properties for the solution.

Type: SolutionConfig (p. 340) object
Required: No

status

The status of the solution.

A solution can be in one of the following states:
- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS
Type: String

Length Constraints: Maximum length of 256.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
SolutionConfig
Service: Amazon Personalize

Describes the configuration properties for the solution.

Contents

algorithmHyperParameters

Lists the hyperparameter names and ranges.
Type: String to string map
Map Entries: Maximum number of 100 items.
Key Length Constraints: Maximum length of 256.
Value Length Constraints: Maximum length of 1000.
Required: No

autoMLConfig

The AutoMLConfig (p. 278) object containing a list of recipes to search when AutoML is performed.
Type: AutoMLConfig (p. 278) object
Required: No

eventValueThreshold

Only events with a value greater than or equal to this threshold are used for training a model.
Type: String
Length Constraints: Maximum length of 256.
Required: No

featureTransformationParameters

Lists the feature transformation parameters.
Type: String to string map
Map Entries: Maximum number of 100 items.
Key Length Constraints: Maximum length of 256.
Value Length Constraints: Maximum length of 1000.
Required: No

hpoConfig

Describes the properties for hyperparameter optimization (HPO).
Type: HPOConfig (p. 327) object
Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:
• AWS SDK for C++
• AWS SDK for Go
• AWS SDK for Java
• AWS SDK for Ruby V3
SolutionSummary
Service: Amazon Personalize

Provides a summary of the properties of a solution. For a complete listing, call the DescribeSolution (p. 220) API.

Contents

creationDateTime
The date and time (in Unix time) that the solution was created.
Type: Timestamp
Required: No

lastUpdatedDateTime
The date and time (in Unix time) that the solution was last updated.
Type: Timestamp
Required: No

name
The name of the solution.
Type: String
Pattern: ^[a-zA-Z0-9\-_]*$
Required: No

descriptionArn
The Amazon Resource Name (ARN) of the solution.
Type: String
Length Constraints: Maximum length of 256.
Pattern: arn:([a-z\d-]+):personalize:.+:.*+.
Required: No

status
The status of the solution.
A solution can be in one of the following states:
• CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
• DELETE PENDING > DELETE IN_PROGRESS
Type: String
Length Constraints: Maximum length of 256.
Required: No
See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
SolutionVersion

Service: Amazon Personalize

An object that provides information about a specific version of a Solution (p. 337).

Contents

creationDateTime

The date and time (in Unix time) that this version of the solution was created.

Type: Timestamp

Required: No

datasetGroupArn

The Amazon Resource Name (ARN) of the dataset group providing the training data.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.+:.*

Required: No

eventType

The event type (for example, 'click' or 'like') that is used for training the model.

Type: String

Length Constraints: Maximum length of 256.

Required: No

failureReason

If training a solution version fails, the reason for the failure.

Type: String

Required: No

lastUpdatedDateTime

The date and time (in Unix time) that the solution was last updated.

Type: Timestamp

Required: No

performAutoML

When true, Amazon Personalize searches for the most optimal recipe according to the solution configuration. When false (the default), Amazon Personalize uses recipeArn.

Type: Boolean

Required: No

performHPO

Whether to perform hyperparameter optimization (HPO) on the chosen recipe. The default is false.
Type: Boolean
Required: No

**recipeArn**

The ARN of the recipe used in the solution.

Type: String
Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.+:*:*:`

Required: No

**solutionArn**

The ARN of the solution.

Type: String
Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.+:*:*:`

Required: No

**solutionConfig**

Describes the configuration properties for the solution.

Type: SolutionConfig (p. 340) object

Required: No

**solutionVersionArn**

The ARN of the solution version.

Type: String
Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.+:*:*:`

Required: No

**status**

The status of the solution version.

A solution version can be in one of the following states:
- CREATE PENDING
- CREATE_IN_PROGRESS
- ACTIVE
- CREATE FAILED

Type: String
Length Constraints: Maximum length of 256.

Required: No
**trainingHours**

The time used to train the model. You are billed for the time it takes to train a model. This field is visible only after Amazon Personalize successfully trains a model.

Type: Double

Valid Range: Minimum value of 0.

Required: No

**trainingMode**

The scope of training used to create the solution version. The **FULL** option trains the solution version based on the entirety of the input solution's training data, while the **UPDATE** option processes only the training data that has changed since the creation of the last solution version. Choose **UPDATE** when you want to start recommending items added to the dataset without retraining the model.

**Important**

The **UPDATE** option can only be used after you've created a solution version with the **FULL** option and the training solution uses the **HRNN-Coldstart Recipe (Legacy)** (p. 81).

Type: String

Valid Values: **FULL** | **UPDATE**

Required: No

**tunedHPOParams**

If hyperparameter optimization was performed, contains the hyperparameter values of the best performing model.

Type: TunedHPOParams (p. 349) object

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
SolutionVersionSummary
Service: Amazon Personalize

Provides a summary of the properties of a solution version. For a complete listing, call the DescribeSolutionVersion (p. 223) API.

Contents

creationDateTime
The date and time (in Unix time) that this version of a solution was created.

Type: Timestamp

Required: No

failureReason
If a solution version fails, the reason behind the failure.

Type: String

Required: No

lastUpdatedDateTime
The date and time (in Unix time) that the solution version was last updated.

Type: Timestamp

Required: No

solutionVersionArn
The Amazon Resource Name (ARN) of the solution version.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-zA-Z0-9]+):personalize:.*:.*:.*

Required: No

status
The status of the solution version.

A solution version can be in one of the following states:
  • CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED

Type: String

Length Constraints: Maximum length of 256.

Required: No

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:

• AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
**TunedHPOParams**
Service: Amazon Personalize

If hyperparameter optimization (HPO) was performed, contains the hyperparameter values of the best performing model.

**Contents**

**algorithmHyperParameters**

A list of the hyperparameter values of the best performing model.

Type: String to string map

Map Entries: Maximum number of 100 items.

Key Length Constraints: Maximum length of 256.

Value Length Constraints: Maximum length of 1000.

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3

**Amazon Personalize Events**

The following data types are supported by Amazon Personalize Events:

- Event (p. 350)
- Item (p. 352)
- User (p. 353)
Event
Service: Amazon Personalize Events

Represents user interaction event information sent using the PutEvents API.

Contents

eventId

An ID associated with the event. If an event ID is not provided, Amazon Personalize generates a unique ID for the event. An event ID is not used as an input to the model. Amazon Personalize uses the event ID to distinguish unique events. Any subsequent events after the first with the same event ID are not used in model training.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: No

eventType

The type of event, such as click or download. This property corresponds to the EVENT_TYPE field of your Interactions schema and depends on the types of events you are tracking.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: Yes

eventValue

The event value that corresponds to the EVENT_VALUE field of the Interactions schema.

Type: Float

Required: No

impression

A list of item IDs that represents the sequence of items you have shown the user. For example, ["itemId1", "itemId2", "itemId3"].

Type: Array of strings

Array Members: Minimum number of 1 item. Maximum number of 25 items.

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: No

itemId

The item ID key that corresponds to the ITEM_ID field of the Interactions schema.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: No
properties

A string map of event-specific data that you might choose to record. For example, if a user rates a movie on your site, other than movie ID (itemId) and rating (eventValue), you might also send the number of movie ratings made by the user.

Each item in the map consists of a key-value pair. For example,

{"numberOfRatings": "12"}

The keys use camel case names that match the fields in the Interactions schema. In the above example, the numberOfRatings would match the 'NUMBER_OF_RATINGS' field defined in the Interactions schema.

Type: String
Required: No

recommendationId

The ID of the recommendation.

Type: String
Required: No

sentAt

The timestamp (in Unix time) on the client side when the event occurred.

Type: Timestamp
Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
Item
Service: Amazon Personalize Events

Represents item metadata added to an Items dataset using the PutItems API.

Contents

itemId
The ID associated with the item.

Type: String
Length Constraints: Minimum length of 1. Maximum length of 256.
Required: Yes

properties
A string map of item-specific metadata. Each element in the map consists of a key-value pair. For example,

{"numberOfRatings": "12"}

The keys use camel case names that match the fields in the Items schema. In the above example, the numberOfRatings would match the 'NUMBER_OF_RATINGS' field defined in the Items schema.

Type: String
Required: No

See Also
For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3
User
Service: Amazon Personalize Events

Represents user metadata added to a Users dataset using the PutUsers API.

Contents

properties

A string map of user-specific metadata. Each element in the map consists of a key-value pair. For example,

{"numberOfVideosWatched": "45"}

The keys use camel case names that match the fields in the Users schema. In the above example, the numberOfVideosWatched would match the 'NUMBER_OF_VIDEOS_WATCHED' field defined in the Users schema.

Type: String

Required: No

userId

The ID associated with the user.

Type: String
Length Constraints: Minimum length of 1. Maximum length of 256.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3

Amazon Personalize Runtime

The following data types are supported by Amazon Personalize Runtime:

- PredictedItem (p. 354)
**PredictedItem**
Service: Amazon Personalize Runtime

An object that identifies an item.

The GetRecommendations (p. 269) and GetPersonalizedRanking (p. 265) APIs return a list of PredictedItems.

**Contents**

**itemId**

The recommended item ID.

Type: String

Length Constraints: Maximum length of 256.

Required: No

**score**

A numeric representation of the model's certainty that the item will be the next user selection. For more information on scoring logic, see How Recommendation Scoring Works (p. 105).

Type: Double

Required: No

**See Also**

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for Ruby V3

**Common Errors**

This section lists the errors common to the API actions of all AWS services. For errors specific to an API action for this service, see the topic for that API action.

**AccessDeniedException**

You do not have sufficient access to perform this action.

HTTP Status Code: 400

**IncompleteSignature**

The request signature does not conform to AWS standards.

HTTP Status Code: 400

**InternalFailure**

The request processing has failed because of an unknown error, exception or failure.
HTTP Status Code: 500

**InvalidAction**

The action or operation requested is invalid. Verify that the action is typed correctly.

HTTP Status Code: 400

**InvalidClientTokenId**

The X.509 certificate or AWS access key ID provided does not exist in our records.

HTTP Status Code: 403

**InvalidParameterCombination**

Parameters that must not be used together were used together.

HTTP Status Code: 400

**InvalidParameterValue**

An invalid or out-of-range value was supplied for the input parameter.

HTTP Status Code: 400

**InvalidQueryParameter**

The AWS query string is malformed or does not adhere to AWS standards.

HTTP Status Code: 400

**MalformedQueryString**

The query string contains a syntax error.

HTTP Status Code: 404

**MissingAction**

The request is missing an action or a required parameter.

HTTP Status Code: 400

**MissingAuthenticationToken**

The request must contain either a valid (registered) AWS access key ID or X.509 certificate.

HTTP Status Code: 403

**MissingParameter**

A required parameter for the specified action is not supplied.

HTTP Status Code: 400

**NotAuthorized**

You do not have permission to perform this action.

HTTP Status Code: 400

**OptInRequired**

The AWS access key ID needs a subscription for the service.

HTTP Status Code: 403
RequestExpired

The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.

HTTP Status Code: 400

ServiceUnavailable

The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

ValidationError

The input fails to satisfy the constraints specified by an AWS service.

HTTP Status Code: 400

Common Parameters

The following list contains the parameters that all actions use for signing Signature Version 4 requests with a query string. Any action-specific parameters are listed in the topic for that action. For more information about Signature Version 4, see Signature Version 4 Signing Process in the Amazon Web Services General Reference.

Action

The action to be performed.

Type: string

Required: Yes

Version

The API version that the request is written for, expressed in the format YYYY-MM-DD.

Type: string

Required: Yes

X-Amz-Algorithm

The hash algorithm that you used to create the request signature.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Valid Values: AWS4-HMAC-SHA256

Required: Conditional
X-Amz-Credential

The credential scope value, which is a string that includes your access key, the date, the region you are targeting, the service you are requesting, and a termination string ("aws4_request"). The value is expressed in the following format: access_key/YYYYMMDD/region/service/aws4_request.

For more information, see Task 2: Create a String to Sign for Signature Version 4 in the Amazon Web Services General Reference.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-Date

The date that is used to create the signature. The format must be ISO 8601 basic format (YYYYMMDD'T'HHMMSS'Z'). For example, the following date time is a valid X-Amz-Date value: 20120325T120000Z.

Condition: X-Amz-Date is optional for all requests; it can be used to override the date used for signing requests. If the Date header is specified in the ISO 8601 basic format, X-Amz-Date is not required. When X-Amz-Date is used, it always overrides the value of the Date header. For more information, see Handling Dates in Signature Version 4 in the Amazon Web Services General Reference.

Type: string

Required: Conditional

X-Amz-Security-Token

The temporary security token that was obtained through a call to AWS Security Token Service (AWS STS). For a list of services that support temporary security credentials from AWS Security Token Service, go to AWS Services That Work with IAM in the IAM User Guide.

Condition: If you're using temporary security credentials from the AWS Security Token Service, you must include the security token.

Type: string

Required: Conditional

X-Amz-Signature

Specifies the hex-encoded signature that was calculated from the string to sign and the derived signing key.

Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional

X-Amz-SignedHeaders

Specifies all the HTTP headers that were included as part of the canonical request. For more information about specifying signed headers, see Task 1: Create a Canonical Request For Signature Version 4 in the Amazon Web Services General Reference.
Condition: Specify this parameter when you include authentication information in a query string instead of in the HTTP authorization header.

Type: string

Required: Conditional
# Document History for Amazon Personalize

The following table describes important changes in each release of the *Amazon Personalize Developer Guide*. For notification about updates to this documentation, you can subscribe to an RSS feed.

<table>
<thead>
<tr>
<th>update-history-change</th>
<th>update-history-description</th>
<th>update-history-date</th>
</tr>
</thead>
<tbody>
<tr>
<td>New feature (p. 359)</td>
<td>Amazon Personalize now automatically updates the latest model (solution version) you trained with User-Personalization every two hours to include new data. For more information, see User-Personalization Recipe.</td>
<td>November 17, 2020</td>
</tr>
<tr>
<td>New feature (p. 359)</td>
<td>Amazon Personalize can now filter recommendations based on criteria you specify when you get recommendations. For more information, see Filtering Recommendations.</td>
<td>November 10, 2020</td>
</tr>
<tr>
<td>New feature (p. 359)</td>
<td>Amazon Personalize now supports the ability to incrementally import users and items. For more information, see Importing Records Incrementally.</td>
<td>October 2, 2020</td>
</tr>
<tr>
<td>New feature (p. 359)</td>
<td>Amazon Personalize now supports a new USER_PERSONALIZATION recommendation recipe. USER_PERSONALIZATION features include modeling impression data, automatic item exploration, and automatic cold item selection. For more information, see User-Personalization Recipe.</td>
<td>August 5, 2020</td>
</tr>
<tr>
<td>New feature (p. 359)</td>
<td>Amazon Personalize can now filter recommendations based on item and user metadata using custom filter expressions. For more information, see Filtering Recommendations.</td>
<td>July 31, 2020</td>
</tr>
<tr>
<td>Date</td>
<td>Event</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>June 10, 2019</td>
<td>Amazon Personalize general availability (p. 359)</td>
<td>Amazon Personalize is now available for general use.</td>
</tr>
<tr>
<td>November 28, 2018</td>
<td>Amazon Personalize preview release (p. 359)</td>
<td>This is the first preview release of the documentation for Amazon Personalize.</td>
</tr>
<tr>
<td>December 18, 2019</td>
<td>New Regions (p. 359)</td>
<td>Amazon Personalize adds support for the Asia Pacific (Mumbai), Asia Pacific (Sydney), and Canada (Central) Regions. For a complete list of the AWS Regions supported by Amazon Personalize, see the AWS Region Table or AWS Regions and Endpoints in the Amazon Web Services General Reference.</td>
</tr>
<tr>
<td>December 19, 2019</td>
<td>New feature (p. 359)</td>
<td>Amazon Personalize can now get recommendations based on contextual metadata. For more information, see Getting Recommendations.</td>
</tr>
<tr>
<td>November 14, 2019</td>
<td>New feature (p. 359)</td>
<td>Amazon Personalize now supports batch recommendation workflows. For more information, see Get Batch Recommendations.</td>
</tr>
<tr>
<td>December 18, 2019</td>
<td>New Regions (p. 359)</td>
<td>Amazon Personalize adds support for the Asia Pacific (Mumbai), Asia Pacific (Sydney), and Canada (Central) Regions. For a complete list of the AWS Regions supported by Amazon Personalize, see the AWS Region Table or AWS Regions and Endpoints in the Amazon Web Services General Reference.</td>
</tr>
<tr>
<td>January 21, 2020</td>
<td>New Region (p. 359)</td>
<td>Amazon Personalize adds support for the Asia Pacific (Seoul) Region. For a complete list of the AWS Regions supported by Amazon Personalize, see the AWS Region Table or AWS Regions and Endpoints in the Amazon Web Services General Reference.</td>
</tr>
<tr>
<td>April 3, 2020</td>
<td>New feature (p. 359)</td>
<td>Amazon Personalize now exposes scores for recommended items. Scores represent the Amazon Personalize model's certainty that a user will next choose a certain item. For more information, see Getting Recommendations.</td>
</tr>
<tr>
<td>June 3, 2020</td>
<td>New feature (p. 359)</td>
<td>Amazon Personalize now allows you to filter results based on which items a user has interacted with. For more information, see Filtering Recommendations.</td>
</tr>
</tbody>
</table>
AWS glossary

For the latest AWS terminology, see the AWS glossary in the AWS General Reference.