
Amazon Personalize

Developer Guide



Amazon Personalize: Developer Guide

Copyright © Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

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
Amazon Personalize terms	4
Data import and management	4
Training	5
Model deployment and recommendations	6
Setting up Amazon Personalize	8
Sign up for AWS	8
Regions and endpoints	8
Setting up permissions	9
Creating a new IAM policy	9
Creating an IAM service role for Amazon Personalize	10
Setting up the AWS CLI	11
Setting up the AWS SDKs	11
Determining your use case	13
Amazon Personalize use cases	13
What data to import	13
Getting started	14
Getting started prerequisites	14
Creating the training data	15
Getting started (console)	15
Getting started (AWS CLI)	24
Getting started (SDK for Python (Boto3))	31
Prerequisites	31
Getting started using Amazon Personalize APIs with Jupyter (iPython) notebooks	32
Getting started (SDK for Java 2.x)	32
Prerequisites	33
Complete Amazon Personalize project	33
Cleaning up resources	40
Datasets and schemas	42
Dataset and schema requirements	42
Schema data types	43
Metadata fields	44
Reserved keywords	44
Interactions dataset	44
Required interaction data	45
Contextual metadata	45
Impressions data	46
Interactions schema example	47
Users dataset	48
Required user data	49
Categorical metadata	49
Users schema example	49
Items dataset	50
Required item data	50
Creation timestamp data	50
Categorical metadata	51
Unstructured text metadata	51
Items schema example	51
Schema examples	52
Creating a schema using Python	52

Preparing and importing data	54
Step 1: Creating a dataset group	54
Creating a dataset group (console)	54
Creating a dataset group (AWS CLI)	55
Creating a dataset group (AWS SDKs)	55
Step 2: Creating a dataset and a schema	57
Creating a dataset and a schema (console)	57
Creating a dataset and a schema (AWS CLI)	58
Creating a dataset and a schema (AWS SDKs)	59
Step 3: Importing your data	62
Importing bulk records	62
Importing records incrementally	73
Recording events	81
Requirements for recording events and training a model	81
How real-time events influence recommendations	82
Creating an event tracker	82
PutEvents operation	84
Recording impressions data	87
Event metrics	89
Events and solutions	89
Sample Jupyter notebook	89
Exporting a dataset	90
Dataset export job permissions requirements	90
Service-linked role policy for exporting a dataset	90
Amazon S3 bucket policy for exporting a dataset	91
Creating a dataset export job (console)	91
Creating a dataset export job (AWS CLI)	92
Creating a dataset export job (AWS SDKs)	93
Creating a solution	96
Step 1: Choosing a recipe	96
Amazon Personalize recipes	96
Viewing available Amazon Personalize recipes	97
USER_PERSONALIZATION recipes	97
PERSONALIZED_RANKING recipes	118
RELATED_ITEMS recipes	122
Step 2: Configuring a solution	127
Configuring a solution (console)	128
Configuring a solution (AWS CLI)	128
Configuring a solution (AWS SDKs)	129
Optimizing a solution	130
Hyperparameters and HPO	134
Choosing the interactions data used for training	136
Step 3: Creating a solution version	139
Creating a solution version (console)	139
Creating a solution version (AWS CLI)	140
Creating a solution version (AWS SDKs)	140
Stopping the creation of a solution version	143
Step 4: Evaluating a solution version with metrics	145
Retrieving Metrics	145
Example	147
Creating a campaign	149
Minimum provisioned TPS	149
Creating a campaign (console)	149
Creating a campaign (AWS CLI)	150
Creating a campaign (AWS SDKs)	151
Updating a campaign	153
Updating a campaign (console)	153

Updating a campaign (AWS CLI)	153
Updating a campaign (AWS SDKs)	154
Getting recommendations	156
Recommendation scores	156
Getting real-time recommendations	156
Contextual metadata	157
Getting recommendations	157
Getting a personalized ranking	160
Getting batch recommendations	163
Batch workflow permissions requirements	164
Batch workflow scoring	164
Preparing and importing batch input data	164
Creating a batch inference job (console)	166
Creating a batch inference job (AWS CLI)	167
Creating a batch inference job (AWS SDKs)	167
Filtering recommendations	170
Filter expressions	170
Filtering real-time recommendations	174
Filtering batch recommendations	182
Maintaining recommendation relevance	184
Keeping datasets current	184
Keeping solution versions up to date	185
Security	186
Data protection	186
Data encryption	187
Identity and access management	187
Audience	187
Authenticating with identities	188
Managing access using policies	190
How Amazon Personalize works with IAM	191
Identity-based policy examples	194
Troubleshooting	197
Logging and monitoring	199
Monitoring	199
CloudWatch metrics for Amazon Personalize	202
Logging Amazon Personalize API calls with AWS CloudTrail	204
Compliance validation	206
Resilience	206
Infrastructure security	206
Endpoints and quotas	207
Amazon Personalize endpoints and regions	207
Compliance	207
Service quotas	207
Requesting a quota increase	210
API reference	211
Actions	211
Amazon Personalize	212
Amazon Personalize Events	329
Amazon Personalize Runtime	335
Data Types	343
Amazon Personalize	344
Amazon Personalize Events	428
Amazon Personalize Runtime	432
Common Errors	433
Common Parameters	435
Document history	438
AWS glossary	441

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, the AWS Command Line Interface (AWS CLI), or programmatically by using the AWS SDK. As the developer, you only need to do the following:

1. Format input data and upload the data into an Amazon S3 bucket, or send real-time event data from user interactions with items.
2. Select a training recipe (algorithm) to use on the data.
3. Train a solution version using the recipe.
4. 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 (historical data) to recommend the most relevant items for the user. You can also use Amazon Personalize to collect interactions data for new properties, such as a 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 more relevant movie recommendations for the user in the future.

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 and input data based on your use case. 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. 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're a first-time user of Amazon Personalize, we recommend you read the following sections in order:

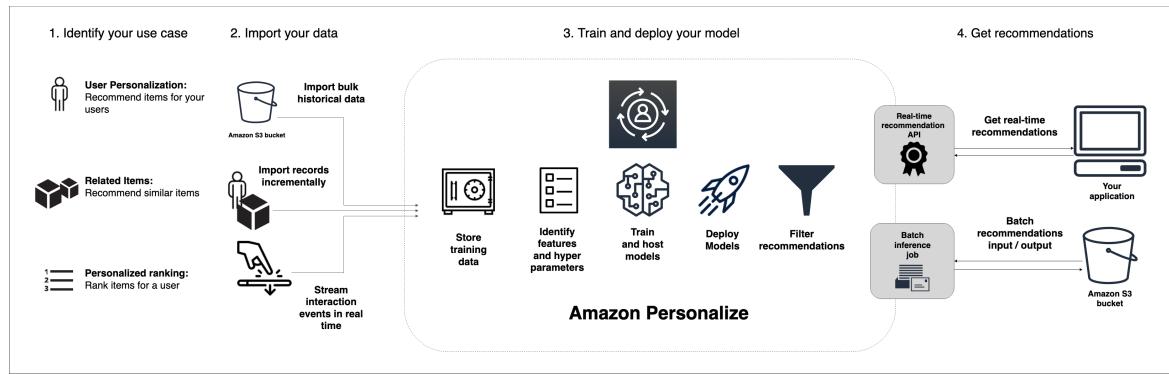
1. **How it works (p. 3)** – This section introduces the Amazon Personalize workflow and walks you through the steps to create personalized experiences for your users. This section also includes common Amazon Personalize terms and their definitions.
 2. **Setting up Amazon Personalize (p. 8)** – In this section you set up your AWS account, set up the required permissions to use Amazon Personalize, and set up the AWS CLI and the AWS SDKs to use and manage Amazon Personalize.
 3. **Determining your use case (p. 13)** – This section introduces various Amazon Personalize components that you work with to create an end-to-end experience.
 4. **Getting started (p. 14)** – In this section you get started using Amazon Personalize with a simple movie dataset. You learn how to create an Amazon Personalize campaign that returns movie recommendations for a user with the Amazon Personalize console, the AWS CLI, and AWS SDKs.
 5. **Preparing and importing data (p. 54)** – This section describes how to prepare and import your training data into Amazon Personalize.
 6. **Recording events (p. 81)** – This section provides information about improving user recommendations by recording user events.
 7. **Creating a solution (p. 96)** – This section provides information about creating a solution version by training a model.
 8. **Creating a campaign (p. 149)** – This section provides information about deploying a solution version as a campaign.
 9. **Getting recommendations (p. 156)** – This section shows how to get recommendations from a campaign.
10. **Maintaining recommendation relevance (p. 184)** – This section shows how to maintain the relevance of Amazon Personalize recommendations as your catalog grows and your customers use your application.

Are you an experienced Amazon Personalize user?

If you're an experienced Amazon Personalize user, you can find in-depth tutorials and code samples in the [amazon-personalize-samples GitHub repository](#).

How it works

Amazon Personalize uses your data to train a custom recommendation model and deploy a recommendation API. You use this API in your application to request real-time recommendations, which Amazon Personalize generates with your custom model. Amazon Personalize also supports batch workflows for use cases that don't require recommendations in real time, such as personalized marketing emails.



Topics

- [Amazon Personalize workflow \(p. 3\)](#)
- [Amazon Personalize terms \(p. 4\)](#)

Amazon Personalize workflow

With Amazon Personalize, you determine your use case, import your data, train and deploy a model, and then get recommendations. Repeat the data import and training processes to sustain and improve the relevance of your recommendations as your catalogue grows. You can complete the Amazon Personalize workflow with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or the AWS SDKs.

1. Determine your use case

Choose your use case from the following and note its corresponding recipe type. (Recipes are Amazon Personalize algorithms prepared for different use cases.)

- Recommending items for users (USER_PERSONALIZATION recipes)
- Ranking items for a given user (PERSONALIZED_RANKING recipes)
- Recommending similar items (RELATED_ITEMS recipes)

For more information see [Determining your use case \(p. 13\)](#).

2. Import data

You import item, user, and interaction records into *datasets* (Amazon Personalize containers for data). You can choose to import records in bulk, or incrementally, or both. With incremental imports, you can add one or more historical records or import data from real-time user activity.

The data that you import depends on your use case. For information about the types of data that you can import, see [Datasets and schemas \(p. 42\)](#) and the sections on each dataset type ([Interactions dataset \(p. 44\)](#), [Items dataset \(p. 50\)](#), [Users dataset \(p. 48\)](#)).

For more information about importing data see [Preparing and importing data \(p. 54\)](#).

3. Train a model

After you've imported your data, Amazon Personalize uses it to train a model. In Amazon Personalize, you start training by creating a *solution*, where you specify your use case by choosing an Amazon Personalize recipe. Then you create a *solution version*, which is the trained model that Amazon Personalize uses to generate recommendations. For more information see [Creating a solution \(p. 96\)](#).

4. Deploy a model (for real-time recommendations)

After Amazon Personalize finishes creating your solution version (trained model), you deploy it in a campaign. A campaign creates and manages a recommendation API that you use in your application to request real-time recommendations from your custom model. For more information about deploying a model see [Creating a campaign \(p. 149\)](#). For batch recommendations, you don't need to create a campaign.

5. Get recommendations

Get recommendations in real time or as part of a batch workflow. Get real-time recommendations when you want to update recommendations as customers use your application. Get batch recommendations when you don't require real-time updates. For more information, see [Getting recommendations \(p. 156\)](#).

6. Refresh your data and repeat

Keep your item and user data current, record new interaction data in real-time, and re-train your model on a regular basis. This allows your model to learn from your user's most recent activity and sustains and improves the relevance of recommendations. For more information see [Maintaining recommendation relevance \(p. 184\)](#).

Amazon Personalize terms

This section introduces the terms used in Amazon Personalize.

Topics

- [Data import and management \(p. 4\)](#)
- [Training \(p. 5\)](#)
- [Model deployment and recommendations \(p. 6\)](#)

Data import and management

The following terms relate to importing, exporting, and formatting data in Amazon Personalize.

contextual metadata

Interactions data that you collect about a user's browsing context (such as device used or location) when an event (such as a click) occurs. Contextual metadata can improve recommendation relevance for new and existing users.

dataset

A container for data that you upload to Amazon Personalize. There are three types of Amazon Personalize datasets: Users, Items, and Interactions.

dataset group

A container for Amazon Personalize components, including datasets, event trackers, solutions, filters, campaigns, and batch inference jobs. A dataset group organizes your resources into independent

collections, so resources from one dataset group can't influence resources in any other dataset group.

dataset export job

A record export tool that outputs the records in a dataset to one or more CSV files in an Amazon S3 bucket. The output CSV file includes a header row with column names that match the fields in the dataset's schema.

dataset import job

A bulk import tool that populates your Amazon Personalize dataset with data from a CSV file in your Amazon S3 bucket.

event

A user action – such as a click, a purchase, or a video viewing – that you record and upload to an Amazon Personalize Interactions dataset. You import events in bulk from a CSV file, incrementally with the Amazon Personalize console, and in real-time.

explicit impressions

A list of items that you manually add to an Amazon Personalize Interactions dataset. Unlike implicit impressions, which Amazon Personalize automatically derives from your recommendation data, you choose what to include in explicit impressions.

implicit impressions

The recommendations that your application shows a user. Unlike explicit impressions, which you manually add to an Interactions dataset, Amazon Personalize automatically derives implicit impressions from your recommendation data.

impressions data

The list of items that you presented to a user when they interacted with a particular item by clicking it, watching it, purchasing it, and so on. Amazon Personalize uses impressions data to calculate the relevance of new items for a user based on how frequently users have selected or ignored the same item.

interactions dataset

A container for historical and real-time data that you collect from interactions between users and items (called [events](#)). Interactions data can include [impressions data](#) and [contextual metadata](#).

items dataset

A container for metadata about your items, such as price, genre, or availability.

schema

A JSON object in [Apache Avro](#) format that tells Amazon Personalize about the structure of your data. Amazon Personalize uses your schema to parse your data.

users dataset

A container for metadata about your users, such as age, gender, or loyalty membership.

Training

The following terms relate to training a model in Amazon Personalize.

item-to-item similarities (SIMS) recipe

A [RELATED_ITEMS](#) recipe that uses the data from an Interactions dataset to make recommendations for items that are similar to a specified item. The SIMS recipe calculates similarity based on the way users interact with items instead of matching item metadata, such as price or color.

personalized-ranking recipe

A [PERSONALIZED_RANKING](#) recipe that ranks a collection of items that you provide based on the predicted interest level for a specific user. Use the personalized-ranking recipe to personalize the order of curated lists of items or search results that are personalized for a specific user.

popularity-count recipe

A [USER_PERSONALIZATION](#) recipe that recommends the items that have had the most interactions with unique users.

recipe

An Amazon Personalize algorithm that is preconfigured to predict the items that a user will interact with (for `USER_PERSONALIZATION` recipes), or calculate items that are similar to specific items that a user has shown interest in (for `RELATED_ITEMS` recipes), or rank a collection of items that you provide based on the predicted interest for a specific user (for `PERSONALIZED_RANKING` recipes).

solution

The recipe, customized parameters, and trained models (Solution Versions) that Amazon Personalize uses to generate recommendations.

solution version

A trained model that you create as part of a solution in Amazon Personalize. You deploy a solution version in a campaign to activate the personalization API that you use to request recommendations.

training mode

The scope of training to be performed when creating a solution version. There are two different modes: FULL and UPDATE. FULL mode creates a completely new solution version based on the entirety of the training data from the datasets in your dataset group. UPDATE incrementally updates the existing solution version to recommend new items that you added since the last training.

Note

With User-Personalization, Amazon Personalize automatically updates the latest solution version trained with FULL training mode. See [Automatic updates \(p. 98\)](#).

user-personalization recipe

A Hierarchical Recurrent Neural Network (HRNN) based [USER_PERSONALIZATION](#) recipe that predicts the items that a user will interact with. The user-personalization recipe can use item exploration and impressions data to generate recommendations for new items.

Model deployment and recommendations

The following terms relate to deploying and using a model in Amazon Personalize.

batch inference job

A tool that imports your batch input data from an Amazon S3 bucket, uses your solution version to generate recommendations, and exports the recommendations to an Amazon S3 bucket. We recommend using a different location for your output data (either a folder or a different Amazon S3 bucket). Use a batch inference job to get recommendations for large datasets that do not require real-time updates.

campaign

A deployed solution version (trained model) with provisioned dedicated transaction capacity for creating real-time recommendations for your application users. After you create a campaign, you use the `getRecommendations` or `getPersonalizedRanking` API operations to get recommendations.

item exploration

The process that Amazon Personalize uses to test different item recommendations and learn how users respond to new items with no or very little interaction data. Amazon Personalize uses exploration when you train the model with the user-personalization recipe. For real-time recommendations, you configure exploration at the campaign level. For batch recommendations, you configure exploration when you create a batch inference job.

recommendations

A list of items that Amazon Personalize predicts a user will interact with. Depending on the Amazon Personalize recipe used, recommendations can be either a list of items (USER_PERSONALIZATION recipes and RELATED_ITEMS recipes), or a ranking of a collection of items you provided (PERSONALIZED_RANKING recipes).

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, SDK for Python (`Boto3`), and SDK for Java 2.x.

Topics

- [Sign up for AWS \(p. 8\)](#)
- [Regions and endpoints \(p. 8\)](#)
- [Setting up permissions \(p. 9\)](#)
- [Setting up the AWS CLI \(p. 11\)](#)
- [Setting up the AWS SDKs \(p. 11\)](#)

Sign up for AWS

When you sign up for 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://portal.aws.amazon.com/billing/signup>.
2. Follow the online instructions.

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

3. Create an AWS Identity and Access Management (IAM) admin user. See [Creating your first IAM user and group](#) in the *AWS Identity and Access Management User Guide* for instructions.

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 \(p. 191\)](#)

4. Create an IAM user for use with Amazon Personalize. The account requires certain permissions. For more information, see [Setting up permissions \(p. 9\)](#).

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 \(p. 191\)](#).

Note

We recommend [creating a new IAM policy \(p. 9\)](#) 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 \(p. 9\)](#)).
 - or
 - Attach the `AmazonPersonalizeFullAccess` AWS managed policy to your IAM user or group (see [AWS managed policies \(p. 195\)](#)).
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 \(p. 195\)](#).
4. Create an IAM role for Amazon Personalize and attach the policy from step 1 to the new role. See [Creating an IAM service role for Amazon Personalize \(p. 10\)](#).
5. If you are using AWS Key Management Service (AWS KMS) for encryption, you must give your IAM user and Amazon Personalize IAM service-linked role permission to use your key. You must also add Amazon Personalize as a Principle in your AWS KMS 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

1. Sign in to the IAM console (<https://console.aws.amazon.com/iam>).
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.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "personalize:*"  
            ],  
            "Resource": "*"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iam:PassRole"  
            ],  
            "Resource": "*",  
            "Condition": {  
                "StringEquals": {  
                    "iam:PassedToService": "personalize.amazonaws.com"  
                }  
            }  
        }  
    ]  
}
```

6. Choose **Next: Tags**. Optionally add any tags and choose **Review**.
7. On the **Review policy** page, for **Name**, enter a name for the policy. Optionally, enter a description for **Description**.
8. In **Summary**, review the policy to see the permissions it grants, then choose **Create policy**.
9. Attach the new policy to your IAM user or group.

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*.

10. If you are using AWS KMS for encryption, give your user or group permission to use your key. For more information see [Using key policies in AWS KMS](#) in the *AWS Key Management Service Developer Guide*.

Creating an IAM service role for Amazon Personalize

To use Amazon Personalize, you must create an AWS Identity and Access Management service role for Amazon Personalize. For information on how to create an IAM service role, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*. As you create your role, configure the following for Amazon Personalize:

- For **Choose the service that will use this role**, choose **Personalize**.
- For **Attach permissions policies**, either choose the policy you created in [Creating a new IAM policy \(p. 9\)](#) or choose **AmazonPersonalizeFullAccess** (see [AWS managed policies \(p. 195\)](#)).
- If you are using AWS KMS for encryption, give your Amazon Personalize service-linked role permission to use your key. For more information see [Using key policies in AWS KMS](#) in the *AWS Key Management Service Developer Guide*.

Next, if you are completing the getting started exercise, you are ready to create your training data and grant Amazon Personalize access to your Amazon S3 bucket. See [Creating the training data \(p. 15\)](#).

If you are not completing the getting started exercise, you are ready to import your data. See [Preparing and importing data \(p. 54\)](#).

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](#) 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](#) in the *AWS Command Line Interface User Guide*.
3. To confirm that the AWS CLI profile is configured properly, run the following command.

```
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 [*****xyz]:  
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.

```
aws personalize help
```

and

```
aws personalize-runtime help
```

and

```
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.

```
pip3 install awscli --upgrade --user
```

For more information, see [Installing the AWS CLI using pip](#).

Setting up the AWS SDKs

Download and install the AWS SDKs that you want to use. This guide provides examples for SDK for Python (Boto3) and SDK for Java 2.x. 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 \(SDK for Python \(Boto3\)\) \(p. 31\)](#).

- [SDK for Java 2.x](#)

To learn about setting up the SDK for Java 2.x, see [Get started with the SDK for Java 2.x](#) topic in the [AWS SDK for Java 2.x Developer Guide](#).

- [AWS Amplify](#)

Determining your use case

Before you use Amazon Personalize, determine your use case to identify what recipe to use to train the model and what data to import. *Recipes* are Amazon Personalize algorithms that are prepared for specific use cases. After you import data, you tell Amazon Personalize what recipe to use.

Topics

- [Amazon Personalize use cases \(p. 13\)](#)
- [What data to import \(p. 13\)](#)

Amazon Personalize use cases

Before you get started providing personalized experiences for your users, choose your use case from the following and note its corresponding recipe type.

- Recommending items for users (USER_PERSONALIZATION recipes)

To provide recommendations for your users, train your model with a USER_PERSONALIZATION recipe. Recommendations can either be *personalized* recommendations, such as recommending movies for a user based on interactions, items, and user data, or *popular* items based on interactions data.

- Ranking items for a user (PERSONALIZED_RANKING recipes)

To personalize the order of curated lists or search results for your users, train your model with a PERSONALIZED_RANKING recipe. PERSONALIZED_RANKING recipes create a personalized list by re-ranking a collection of input items based on predicted interest level for a given user. Personalized lists improve the customer experience and increase customer loyalty and engagement.

- Recommending similar items (RELATED_ITEMS recipes)

To recommend similar items, such as items frequently bought together or movies that other users have also watched, you should use a RELATED_ITEMS recipe. Recommending similar items can help your customers discover items and can increase user conversion rate.

What data to import

If you are providing personalized recommendations for your users, import data that helps Amazon Personalize recommend items that are new or have fewer user interactions. This data includes creation timestamp data and unstructured text metadata about your items, as well as impressions data and contextual metadata from your customers' interactions with items.

For personalized recommendations or ranked items, import categorical data, such as a user's gender or an item's genre, to help Amazon Personalize identify underlying patterns that reveal the most relevant items for your users. For all use cases, you can use categorical metadata to filter recommendations from Amazon Personalize based on a user or item's attributes.

For general information about how Amazon Personalize reads and stores your data, see [Datasets and schemas \(p. 42\)](#). For more information on the types of data you can import, see [Interactions dataset \(p. 44\)](#), [Items dataset \(p. 50\)](#), and [Users dataset \(p. 48\)](#).

Getting started

This getting started guide shows you how to provide personalized movie recommendations for your user. The tutorial uses 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 \(p. 81\)](#).

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 \(console\) \(p. 15\)](#) or [Getting started \(AWS CLI\) \(p. 24\)](#).

When you finish the getting started exercise, to avoid incurring unnecessary charges, follow the steps in [Cleaning up resources \(p. 40\)](#) to delete the resources you created.

Topics

- [Getting started prerequisites \(p. 14\)](#)
- [Getting started \(console\) \(p. 15\)](#)
- [Getting started \(AWS CLI\) \(p. 24\)](#)
- [Getting started \(SDK for Python \(Boto3\)\) \(p. 31\)](#)
- [Getting started \(SDK for Java 2.x\) \(p. 32\)](#)
- [Cleaning up resources \(p. 40\)](#)

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 \(p. 8\)](#).
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 \(p. 9\)](#).
3. Create an AWS Identity and Access Management (IAM) service role, as specified in [Creating an IAM service role for Amazon Personalize \(p. 10\)](#). 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 \(p. 15\)](#). 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 \(p. 65\)](#).

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 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 \(p. 65\)](#).

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 \(p. 14\)](#).

When you finish the getting started exercise, to avoid incurring unnecessary charges, follow the steps in [Cleaning up resources \(p. 40\)](#) 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/home> 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:

Create dataset group [Info](#)

A dataset group contains the datasets, solutions, and an event ingestion API for related solutions.

Dataset group details

Dataset group name

The name you enter here can help you distinguish this dataset group from others.

ratings-dsgroup

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

[Cancel](#)

[Next](#)

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 \(p. 15\)](#).
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.

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 [Info](#)

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.

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

Schema definition [Info](#)
Ensure your dataset's schema matches the following schema.

```
1 {  
2   "type": "record",  
3   "name": "Interactions",  
4   "namespace": "com.amazonaws.personalize.schema",  
5   "fields": [  
6     {  
7       "name": "USER_ID",  
8       "type": "string"  
9     },  
10    {  
11      "name": "ITEM_ID",  
12      "type": "string"  
13    },  
14    {  
15      "name": "TIMESTAMP",  
16      "type": "long"  
17    }  
18  ],  
19  "version": "1.0"  
20}
```

[Cancel](#) [Next](#)

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 service role for Amazon Personalize \(p. 10\)](#).
13. In the informational dialog box named **Additional S3 bucket policy required**, follow the [instructions \(p. 64\)](#) 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:

Import user-item interaction data Info

In this step, you create a dataset import job which imports your data from S3.

Dataset import job details

Dataset import job name

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

ratings-dsimport-job

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

IAM service role

Amazon Personalize requires permissions to access your S3 bucket. Choose an existing role with access or create a role in the IAM console with the [AmazonPersonalizeFullAccess](#) IAM policy attached.

Enter a custom IAM role ARN

Custom IAM role ARN

arn:aws:iam::<account-id>:role/PersonalizeRole



Additional S3 bucket policy required

In addition to the IAM service role defined above, Amazon Personalize also requires you to add a bucket policy to the S3 bucket containing your data files so that it can process them. Follow the instructions [described here](#) to add the required bucket policy to your S3 bucket..

Data location Info

Choose the S3 location of your data.

s3://aws-personalize-demo-bucket/ratings.csv

Your file needs to be in a CSV format and reflect the schema.

Cancel

Previous

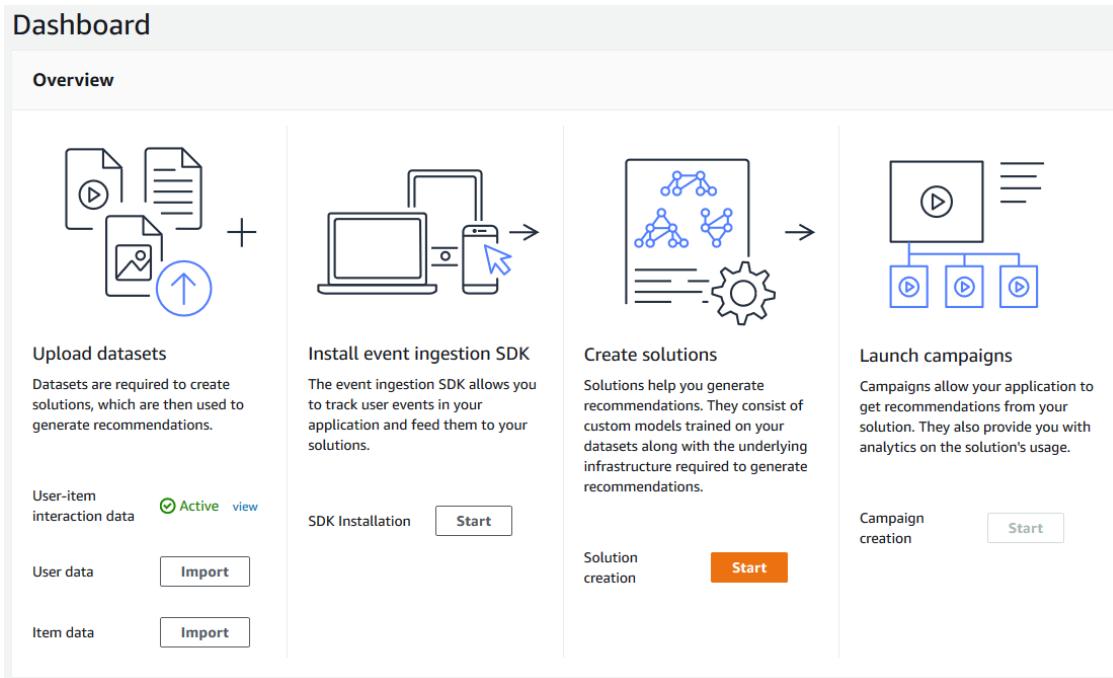
Finish

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** and **Advanced configuration** fields unchanged.

Your screen should look similar to the following:

Create solution Info

You create a solution using a recipe that is tailored to a specific use case.

Solution detail

Solution name
The solution name that you enter here can help you distinguish this solution from others.

The solution name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).

Recipe Info
Recipes are preconfigured algorithms tailored to specific use cases.

Predicts items a user will interact with and performs exploration on cold items. Based on Hierarchic...

▼ Solution configuration - optional Info

Additional objective - optional Info
The primary objective is to predict relevant items for your users. You can add an additional objective, such as maximizing revenue. Choose a column containing numerical data from your items dataset that matches your objective.

▼ Advanced configuration - optional

▼ Hyperparameter optimization (HPO) - optional Info

Perform HPO
Choose whether you would like Amazon Personalize to find optimal hyperparameters.

► Featurization transformation hyperparameters - optional Info

► Algorithm hyperparameters - optional Info

Cancel **Next**

4. Choose **Next** to display the **Create solution version** screen.

Your screen should look similar to the following:

Create solution version

Now that you have created a solution, click Finish button to create a solution version (train your solution).

Solution overview		
Solution name my-movie-solution	Perform HPO false	Event type -
Solution ARN arn:aws:personalize:us-west-2: <account number>:solution/my-movie-solution	Perform AutoML false	Latest solution version ARN -
Status Active	Recipe aws-user-personalization	Created Wed, 12 Aug 2020 22:14:27 GMT
► Solution config		

Cancel **Previous** **Finish**

5. 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, which deploys 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. Leave the **Campaign configuration** fields unchanged.

Your screen should look similar to the following:

Campaign details

Campaign name
The text you enter here appears in the Campaign dashboard and detail page. It can help you distinguish this campaign from others.

my-movie-campaign

The campaign name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).

Solution
The selected solution is used to generate the recommendations provided in your campaign.

my-movie-solution

Solution version ID
The selected solution version is used to generate the recommendations provided in your campaign.

bd4f3130

Wed, 25 Aug 2021 19:06:47 GMT

Minimum provisioned transactions per second [Info](#)
The minimum amount of throughput in transactions per second (TPS) that is provisioned for this campaign.

1

Enter a number from 1-500.

Campaign configuration

Exploration weight [Info](#)
Configure how frequently recommendations include items with less interactions data or relevance. The greater the value (closer to 1), the more exploration. At 0, no exploration occurs.

0.3

Choose a value between 0 and 1.

Exploration item age cut off
Enter the item age cut off, in days since the latest interaction, to define the scope of item exploration.

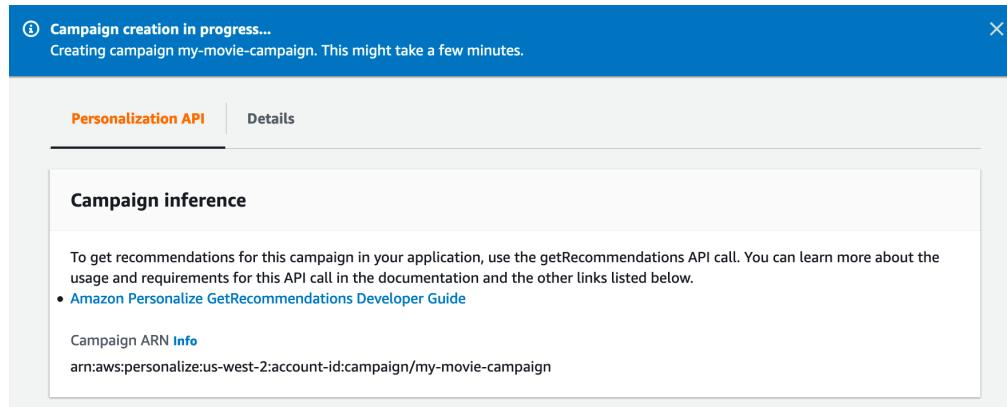
30.0

Choose a positive value.

[Cancel](#) [Create campaign](#)

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

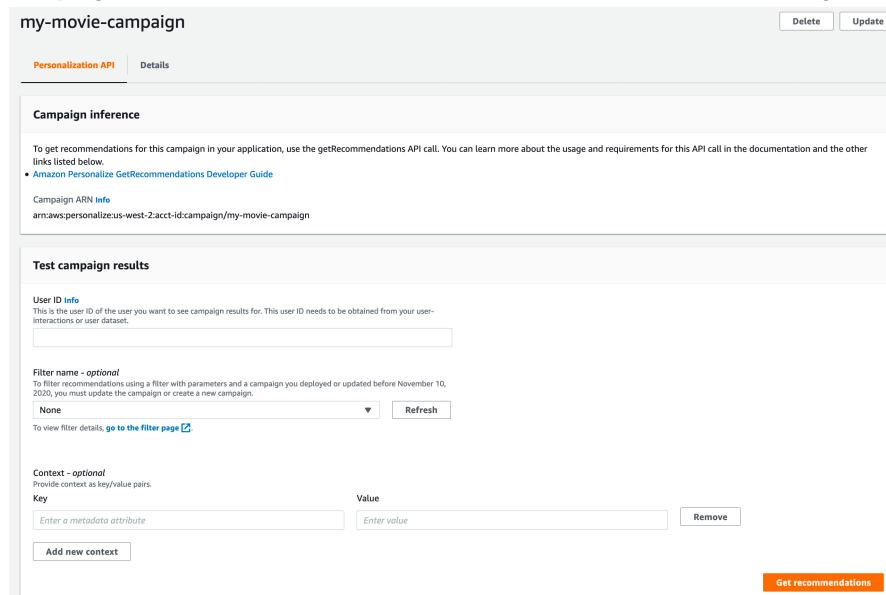
Your screen should look similar to the following:



Note

Creating a campaign takes time.

After Amazon Personalize finishes creating your campaign, 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** and leave the **Context** fields empty.
2. Choose **Get recommendations**. The **Recommendations** panel lists the item IDs and scores for the recommended items.

Your screen should look similar to the following:

Item ID	Score
1391	0.0117211
1302	0.0077976
2012	0.0072628
1676	0.0061814

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 \(p. 14\)](#).
- Set up the AWS CLI, as specified in [Setting up the AWS CLI \(p. 11\)](#).

When you finish the getting started exercise, to avoid incurring unnecessary charges, follow the steps in [Cleaning up resources \(p. 40\)](#) to delete the resources you created.

Note

The AWS CLI commands in this exercise were tested on Linux. For information about using the AWS 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 \(p. 227\)](#).

```
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 \  
--dataset-group-arn arn:aws:personalize:us-west-2:acct-id:dataset-group/  
MovieRatingDatasetGroup
```

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

```
{  
    "datasetGroup": {  
        "name": "MovieRatingDatasetGroup",  
        "datasetGroupArn": "arn:aws:personalize:us-west-2:acct-id:dataset-group/  
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 schemas \(p. 42\)](#).

```
{  
    "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"
}

```

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 \(p. 238\)](#).

```

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 \(p. 221\)](#).

```

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

```

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. Provide the dataset ARN and Amazon S3 bucket name that were returned in the previous steps. Supply the AWS Identity and Access Management (IAM) role ARN you created in [Creating an IAM service role for Amazon Personalize \(p. 10\)](#). For more information about the API, see [CreateDatasetImportJob \(p. 230\)](#).

```

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 \  
--role-arn roleArn
```

The dataset import job ARN is displayed, for example:

```
{  
    "datasetImportJobArn": "arn:aws:personalize:us-west-2:acct-id:dataset-import-job/  
MovieRatingImportJob"  
}
```

- 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 \(p. 274\)](#).

```
aws personalize describe-dataset-import-job \  
--dataset-import-job-arn arn:aws:personalize:us-west-2:acct-id:dataset-import-  
job/MovieRatingImportJob
```

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

```
{  
    "datasetImportJob": {  
        "jobName": "MovieRatingImportJob",  
        "datasetImportJobArn": "arn:aws:personalize:us-west-2:acct-id:dataset-import-  
job/MovieRatingImportJob",  
        "datasetArn": "arn:aws:personalize:us-west-2:acct-id:dataset/  
MovieRatingDatasetGroup/INTERACTIONS",  
        "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 \(p. 240\)](#) operation. Second, you train the model using the [CreateSolutionVersion \(p. 245\)](#) 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 \(p. 96\)](#). For this exercise, you use the User-Personalization recipe.

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

```
aws personalize create-solution \  
  --name MovieSolution \  
  --dataset-group-arn arn:aws:personalize:us-west-2:acct-id:dataset-group/  
MovieRatingDatasetGroup \  
  --recipe-arn arn:aws:personalize:::recipe/aws-user-personalization
```

The solution ARN is displayed, for example:

```
{  
  "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 \(p. 286\)](#).

```
aws personalize describe-solution \  
  --solution-arn arn:aws:personalize:us-west-2:acct-id:solution/MovieSolution
```

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

```
{  
  "solution": {  
    "name": "MovieSolution",  
    "solutionArn": "arn:aws:personalize:us-west-2:acct-id:solution/MovieSolution",  
    "performHPO": false,  
    "performAutoML": false,  
    "recipeArn": "arn:aws:personalize:::recipe/aws-user-personalization",  
    "datasetGroupArn": "arn:aws:personalize:us-west-2:acct-id:dataset-group/  
MovieRatingDatasetGroup",  
    "solutionConfig": {},  
    "status": "ACTIVE",  
    "creationDateTime": "2021-05-12T16:27:59.819000-07:00",  
    "lastUpdatedDateTime": "2021-05-12T16:27:59.819000-07:00"  
  }  
}
```

3. When the solution is `ACTIVE`, train the model by running the following command.

```
aws personalize create-solution-version \  
  --solution-arn arn:aws:personalize:us-west-2:acct-id:solution/MovieSolution
```

The solution version ARN is displayed, for example:

```
{  
  "solutionVersionArn": "arn:aws:personalize:us-west-2:acct-id:solution/MovieSolution/  
<version-id>"  
}
```

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 \(p. 289\)](#).

```
aws personalize describe-solution-version \  
  --solution-version-arn arn:aws:personalize:us-west-2:acct-id:solution/MovieSolution/<version-id>
```

```
--solution-version-arn arn:aws:personalize:us-west-2:acct-id:solution/  
MovieSolution/version-id
```

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

```
{
  "solutionVersion": {
    "solutionVersionArn": "arn:aws:personalize:us-west-2:acct-id:solution/  
MovieSolution/<version-id>",
    ...
    "status": "CREATE PENDING"
  }
}
```

- When the solution version status is ACTIVE, the training is complete.

Now you can review training 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.

- 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 \(p. 292\)](#).

```
aws personalize get-solution-metrics \  
  --solution-version-arn arn:aws:personalize:us-west-2:acct-id:solution/  
MovieSolution/version-id
```

A sample response is shown:

```
{
  "solutionVersionArn": "arn:aws:personalize:us-west-2:acct-id:solution/www-solution/  
<version-id>",
  "metrics": {
    "coverage": 0.0485,
    "mean_reciprocal_rank_at_25": 0.0381,
    "normalized_discounted_cumulative_gain_at_10": 0.0363,
    "normalized_discounted_cumulative_gain_at_25": 0.0984,
    "normalized_discounted_cumulative_gain_at_5": 0.0175,
    "precision_at_10": 0.0107,
    "precision_at_25": 0.0207,
    "precision_at_5": 0.0107
  }
}
```

Step 3: Create a campaign (deploy the solution)

Before you can get recommendations, you must deploy a solution version. 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 \(p. 340\)](#) API.

- 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 \(p. 218\)](#).

```
aws personalize create-campaign \  
  --name MovieRecommendationCampaign \  
  --solution-version-arn arn:aws:personalize:us-west-2:acct-id:solution/  
MovieSolution/version-id
```

```
--solution-version-arn arn:aws:personalize:us-west-2:acct-id:solution/  
MovieSolution/version-id \  
--min-provisioned-tps 1
```

A sample response is shown:

```
{  
    "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 \(p. 266\)](#).

```
aws personalize describe-campaign \  
--campaign-arn arn:aws:personalize:us-west-2:acct-id:campaign/  
MovieRecommendationCampaign
```

A sample response is shown:

```
{  
    "campaign": {  
        "name": "MovieRecommendationCampaign",  
        "campaignArn": "arn:aws:personalize:us-west-2:acct-id:campaign/  
MovieRecommendationCampaign",  
        "solutionVersionArn": "arn:aws:personalize:us-west-2:acct-id:solution/  
MovieSolution/<version-id>",  
        "minProvisionedTPS": "1",  
        "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 \(p. 340\)](#).

Note

Not all recipes support the GetRecommendations API. For more information, see [Step 1: Choosing a recipe \(p. 96\)](#).

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 (SDK for Python (Boto3))

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 prerequisites \(p. 14\)](#). You create the training data used in this tutorial when you complete the prerequisite steps. If you are using your own source data, make sure your data is formatted like in the prerequisite step [Creating the training data \(p. 15\)](#).
- 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 \(p. 40\)](#) 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.

```
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 \(p. 52\)](#).
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 \(AWS SDKs\) \(p. 71\)](#).
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 \(p. 82\)](#).
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 \(p. 84\)](#).

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 \(p. 96\)](#).

When you create a solution version, evaluate its performance before proceeding. For a code sample, see [Step 4: Evaluating a solution version with metrics \(p. 145\)](#).

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 \(p. 149\)](#).

Step 5: Get recommendations

After you create a campaign, you can use it to get recommendations. For a code sample, see [Getting recommendations \(p. 157\)](#).

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`

Getting started (SDK for Java 2.x)

This tutorial shows you how to complete the Amazon Personalize workflow from start to finish with the AWS SDK for Java 2.x.

To avoid incurring unnecessary charges, when you finish the getting started exercise follow the steps in [Cleaning up resources \(p. 40\)](#) to delete the resources you create in the tutorial.

Topics

- [Prerequisites \(p. 33\)](#)
- [Complete Amazon Personalize project \(p. 33\)](#)

Prerequisites

The following are prerequisite steps for completing this tutorial:

- Complete the [Getting started prerequisites \(p. 14\)](#). You create the training data for this tutorial when you complete the prerequisite steps. You can use the same source data used in the [Getting started \(console\) \(p. 15\)](#) or [Getting started \(AWS CLI\) \(p. 24\)](#) exercises. If you are using your own source data, make sure that your data is formatted like in the prerequisite step [Creating the training data \(p. 15\)](#).
- Set up your SDK for Java 2.x environment and AWS credentials as specified in the [Setting up the AWS SDK for Java 2.x](#) procedure in the *AWS SDK for Java 2.x Developer Guide*.

Step 1: Set up your project to use Amazon Personalize packages

After you complete the prerequisites, add Amazon Personalize dependencies to your pom.xml file and import Amazon Personalize packages.

1. Add the following dependencies to your pom.xml file. The latest version numbers may be different than the example code.

```
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>personalize</artifactId>
  <version>2.16.83</version>
</dependency>
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>personalizeruntime</artifactId>
  <version>2.16.83</version>
</dependency>
<dependency>
  <groupId>software.amazon.awssdk</groupId>
  <artifactId>personalizeevents</artifactId>
  <version>2.16.83</version>
</dependency>
```

2. Add the following import statements to your project.

```
// import client packages
import software.amazon.awssdk.services.personalize.PersonalizeClient;
import software.amazon.awssdk.services.personalizeruntime.PersonalizeRuntimeClient;
// The PersonalizeEventsClient is optional. Import if you are going to add interactions
// to the Interactions dataset in real time.
import software.amazon.awssdk.services.personalizeevents.PersonalizeEventsClient;
// Amazon Personalize exception package
import software.amazon.awssdk.services.personalize.model.PersonalizeException;
// schema packages
import software.amazon.awssdk.services.personalize.model.CreateSchemaRequest;
// dataset group packages
import software.amazon.awssdk.services.personalize.model.CreateDatasetGroupRequest;
import software.amazon.awssdk.services.personalize.model.DescribeDatasetGroupRequest;
// dataset packages
import software.amazon.awssdk.services.personalize.model.CreateDatasetRequest;
// dataset import job packages
```

```
import software.amazon.awssdk.services.personalize.model.CreateDatasetImportJobRequest;
import software.amazon.awssdk.services.personalize.model.DataSource;
import software.amazon.awssdk.services.personalize.model.DatasetImportJob;
import software.amazon.awssdk.services.personalize.model.DescribeDatasetImportJobRequest;
// solution packages
import software.amazon.awssdk.services.personalize.model.CreateSolutionRequest;
import software.amazon.awssdk.services.personalize.model.CreateSolutionResponse;
// solution version packages
import software.amazon.awssdk.services.personalize.model.DescribeSolutionRequest;
import software.amazon.awssdk.services.personalize.model.CreateSolutionVersionRequest;
import software.amazon.awssdk.services.personalize.model.CreateSolutionVersionResponse;
import software.amazon.awssdk.services.personalize.model.DescribeSolutionVersionRequest;
// campaign packages
import software.amazon.awssdk.services.personalize.model.CreateCampaignRequest;
import software.amazon.awssdk.services.personalize.model.CreateCampaignResponse;
// get recommendations packages
import
    software.amazon.awssdk.services.personalizeruntime.model.GetRecommendationsRequest;
import
    software.amazon.awssdk.services.personalizeruntime.model.GetRecommendationsResponse;
import software.amazon.awssdk.services.personalizeruntime.model.PredictedItem;
// Java time utility package
import java.time.Instant;
```

Step 2: Instantiate Amazon Personalize clients

After you add Amazon Personalize dependencies to your pom.xml file and imported the required packages, instantiate the following Amazon Personalize clients:

```
PersonalizeClient personalizeClient = PersonalizeClient.builder()
    .region(region)
    .build();

PersonalizeEventsClient personalizeEventsClient = PersonalizeEventsClient.builder()
    .region(region)
    .build();

// a PersonalizeRuntimeClient is optional for this tutorial. Optionally use this client if
// you want to add new interactions to the Interactions dataset in real-time.
PersonalizeRuntimeClient personalizeRuntimeClient = PersonalizeRuntimeClient.builder()
    .region(region)
    .build();
```

Step 3: Import data

After you initialize your Amazon Personalize clients, import the historical data you created when you completed the [Getting started prerequisites \(p. 14\)](#). To import historical data into Amazon Personalize, do the following:

1. Save the following Avro schema as a JSON file in your working directory. This schema matches the columns in the CSV file you created when you completed the [Getting started prerequisites \(p. 14\)](#).

```
{
    "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. Use the following `createSchema` method to create a schema in Amazon Personalize. Pass the following as parameters: an Amazon Personalize service client, the name for your schema, and the file path for the schema JSON file you created in the previous step. The method returns the Amazon Resource Name (ARN) of your new schema. Store it for later use.

```

public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

    String schema = null;
    try {
        schema = new String(Files.readAllBytes(Paths.get(filePath)));
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }

    try {
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
            .name(schemaName)
            .schema(schema)
            .build();

        String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();

        System.out.println("Schema arn: " + schemaArn);

        return schemaArn;

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}

```

3. Create a dataset group. Use the following `createDatasetGroup` method to create a dataset group. Pass the following as parameters: an Amazon Personalize service client and the name for the dataset group. The method returns the ARN of your new dataset group. Store it for later use.

```

public static String createDatasetGroup(PersonalizeClient personalizeClient, String
datasetGroupName) {

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
            .name(datasetGroupName)
            .build();
        return
personalizeClient.createDatasetGroup(createDatasetGroupRequest).datasetGroupArn();
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}

```

```
    }  
    return "";  
}
```

4. Create an Interactions dataset. Use the following `createDataset` method to create an Interactions dataset. Pass the following as parameters: an Amazon Personalize service client, the name for your dataset, your schema's ARN, your dataset group's ARN, and `Interactions` for the dataset type. The method returns the ARN of your new dataset. Store it for later use.

```
public static String createDataset(PersonalizeClient personalizeClient,
                                    String datasetName,
                                    String datasetGroupArn,
                                    String datasetType,
                                    String schemaArn) {
    try {
        CreateDatasetRequest request = CreateDatasetRequest.builder()
            .name(datasetName)
            .datasetGroupArn(datasetGroupArn)
            .datasetType(datasetType)
            .schemaArn(schemaArn)
            .build();

        String datasetArn = personalizeClient.createDataset(request)
            .datasetArn();
        System.out.println("Dataset " + datasetName + " created.");
        return datasetArn;
    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

5. Import your data with a dataset import job. Use the following `createPersonalizeDatasetImportJob` method to create a dataset import job.

Pass the following as parameters: an Amazon Personalize service client, your dataset group's ARN, a name for the job, your Interactions dataset's ARN, the Amazon S3 bucket path (`s3://bucket name/folder name/ratings.csv`) where you stored the training data, and your service-linked role's ARN (you created this role as part of the [Getting started prerequisites \(p. 14\)](#)). The method returns the ARN of your dataset import job. Optionally store it for later use.

```
public static String createPersonalizeDatasetImportJob(PersonalizeClient
personalizeClient,
                                                       String jobName,
                                                       String datasetArn,
                                                       String s3BucketPath,
                                                       String roleArn) {

    long waitInMilliseconds = 60 * 1000;
    String status;
    String datasetImportJobArn;

    try {
        DataSource importDataSource = DataSource.builder()
            .dataLocation(s3BucketPath)
            .build();

        CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
            .datasetArn(datasetArn)
            .dataSource(importDataSource)
            .jobName(jobName)
```

```

        .roleArn(roleArn)
        .build();

        datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
        .datasetImportJobArn();
        DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
        .datasetImportJobArn(datasetImportJobArn)
        .build();

        long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        while (Instant.now().getEpochSecond() < maxTime) {

            DatasetImportJob datasetImportJob = personalizeClient
                .describeDatasetImportJob(describeDatasetImportJobRequest)
                .datasetImportJob();

            status = datasetImportJob.status();
            System.out.println("Dataset import job status: " + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        return datasetImportJobArn;

    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}

```

6. (Optional) Add an event tracker and record events. For more information, see [Recording events \(p. 81\)](#).

For more information about importing data into Amazon Personalize, see [Preparing and importing data \(p. 54\)](#).

Step 4: Create a solution

After you import your data, you create a solution and solution version as follows. The *solution* contains the configurations to train a model and a *solution version* is a trained model.

1. Create a new solution with the following `createPersonalizeSolution` method. Pass the following as parameters: an Amazon Personalize service client, your dataset groups Amazon Resource Name (ARN), a name for the solution, and the ARN for the User-Personalization recipe (`arn:aws:personalize:::recipe/aws-user-personalization`). The method returns the ARN for your new solution. Store it for later use.

```

public static String createPersonalizeSolution(PersonalizeClient personalizeClient,
                                                String datasetGroupArn,
                                                String solutionName,
                                                String recipeArn) {

    try {

```

```

CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
    .name(solutionName)
    .datasetGroupArn(datasetGroupArn)
    .recipeArn(recipeArn)
    .build();

CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);
return solutionResponse.solutionArn();

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

```

2. Create a solution version with the following `createPersonalizeSolutionVersion` method. Pass as a parameter the ARN of the solution the previous step. The following code first checks to see if your solution is ready and then creates a solution version. During training, the code uses the [DescribeSolutionVersion \(p. 289\)](#) operation to retrieve the solution version's status. When training is complete, the method returns the ARN of your new solution version. Store it for later use.

```

public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String solutionStatus = "";
    String solutionVersionStatus = "";
    String solutionVersionArn = "";

    try {
        DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
    .solutionArn(solutionArn)
    .build();

        maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;;

        // Wait until solution is active.
        while (Instant.now().getEpochSecond() < maxTime) {

            solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
            System.out.println("Solution status: " + solutionStatus);

            if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }

        if (solutionStatus.equals("ACTIVE")) {

            CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
    .solutionArn(solutionArn)
    .build();

```

```
        CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient.createSolutionVersion(createSolutionVersionRequest);
        solutionVersionArn = createSolutionVersionResponse.solutionVersionArn();

        System.out.println("Solution version ARN: " + solutionVersionArn);

        DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
        .solutionVersionArn(solutionVersionArn)
        .build();

        while (Instant.now().getEpochSecond() < maxTime) {

            solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest).solutionVersion().status();
            System.out.println("Solution version status: " +
solutionVersionStatus);

            if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
                break;
            }
            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
        return solutionVersionArn;
    }
} catch(PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}
```

For more information, see [Creating a solution \(p. 96\)](#). When you create a solution version, you can evaluate its performance before proceeding. For more information, see [Step 4: Evaluating a solution version with metrics \(p. 145\)](#).

Step 5: Create a campaign

After you train and evaluate your solution version, deploy it with an Amazon Personalize campaign. Use the following `createPersonalCampaign` method to deploy a solution version. Pass the following as parameters: an Amazon Personalize service client, the Amazon Resource Name (ARN) of the solution version you created in the previous step, and a name for the campaign. The method returns the ARN of your new campaign. Store it for later use.

```
public static String createPersonalCompanion(PersonalizeClient personalizeClient, String  
solutionVersionArn, String name) {  
  
    try {  
        CreateCampaignRequest createCampaignRequest = CreateCampaignRequest.builder()  
            .minProvisionedTPS(1)  
            .solutionVersionArn(solutionVersionArn)  
            .name(name)  
            .build();  
  
        CreateCampaignResponse campaignResponse =  
personalizeClient.createCampaign(createCampaignRequest);  
        System.out.println("The campaign ARN is "+campaignResponse.campaignArn());  
    } catch (Exception e) {  
        System.out.println("Error creating campaign: " + e.getMessage());  
    }  
}
```

```
        return campaignResponse.campaignArn();

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

For more information about Amazon Personalize campaigns, see [Creating a campaign \(p. 149\)](#).

Step 6: Get recommendations

After you create a campaign, you use it to get recommendations. Use the following `getRecs` method to get recommendations for a user. Pass as parameters an Amazon Personalize runtime client, the Amazon Resource Name (ARN) of the campaign you created in the previous step, and a user ID (for example, 123) from the historical data you imported. The method prints the list of recommended items to the screen.

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient, String
campaignArn, String userId){

    try {
        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
            .campaignArn(campaignArn)
            .numResults(20)
            .userId(userId)
            .build();

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient.getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();

        for (PredictedItem item: items) {
            System.out.println("Item Id is : "+item.itemId());
            System.out.println("Item score is : "+item.score());
        }
    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Complete Amazon Personalize project

For an all-in-one project that shows you how to complete the Amazon Personalize workflow with the SDK for Java 2.x, see [create_amazon_personalize_app](#) in the [AWS SDK examples](#) repository. This project includes training multiple solution versions with different recipes, and recording events with the `PutEvents` operation.

For additional examples, see code found in the `personalize` folder of the AWS SDK examples repository.

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 \(p. 248\)](#) 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 \(p. 266\)](#).

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?](#).

Resource to be deleted	Delete this first	Notes
Campaign (p. 360)		
DatasetImportJob (p. 381)		Can not be deleted.
EventTracker (p. 395)		The event-interactions dataset that is associated with the event tracker is not deleted and continues to be used by the solution version.
Dataset (p. 369)		No associated <code>DatasetImportJob</code> can have a status of CREATE PENDING or IN PROGRESS. No associated <code>SolutionVersion</code> can have a status of CREATE PENDING or IN PROGRESS.
DatasetSchema (p. 385)	All datasets that reference the schema.	
Solution (p. 416)	All campaigns based on the solution version.	No associated <code>SolutionVersion</code> can have a status of CREATE PENDING or IN PROGRESS.
SolutionVersion (p. 423)		Deleted when the associated <code>Solution</code> is deleted.
DatasetGroup (p. 377)	All associated event trackers. All associated solutions. All datasets in the dataset group.	

Datasets and schemas

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

- [Users \(p. 48\)](#) – 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 \(p. 50\)](#) – This dataset stores metadata about your items. This might include information such as price, SKU type, or availability.
- [Interactions \(p. 44\)](#) – 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 \(p. 96\)](#).

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 \(p. 54\)](#).

Topics

- [Dataset and schema requirements \(p. 42\)](#)
- [Interactions dataset \(p. 44\)](#)
- [Users dataset \(p. 48\)](#)
- [Items dataset \(p. 50\)](#)
- [Schema examples \(p. 52\)](#)
- [Creating a schema using the AWS Python SDK \(p. 52\)](#)

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.

Dataset type	Required fields	Reserved keywords
Users	USER_ID (<code>string</code>) 1 metadata field (categorical <code>string</code> or numerical)	
Items	ITEM_ID (<code>string</code>) 1 metadata field (categorical or textual <code>string</code> field or numerical field)	CREATION_TIMESTAMP (<code>long</code>)

Dataset type	Required fields	Reserved keywords
Interactions	USER_ID (string) ITEM_ID (string) TIMESTAMP (long)	EVENT_TYPE (string) EVENT_VALUE (float, null) IMPRESSION (string) RECOMMENDATION_ID (string, null)

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.

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

- You must define the schema in [Avro format](#). For information on the Avro data types we support, see [Schema data types \(p. 43\)](#).
- 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.
- Schemas must be flat JSON files without nested structures. For example, a field cannot be the parent of multiple sub-fields.
- Schema fields must have unique alphanumeric names. For example, you can't add both a GENRES_FIELD_1 field and a GENRESFIELD1 field.

Schema data types

You must define your schema in [Avro format](#). Amazon Personalize doesn't support complex types such as arrays and maps. For fields with multiple values, including categorical metadata and impressions data, use the data type *string*. When you format your data, separate each value using the vertical bar, '|', character.

We support only the following Avro types for fields ([Reserved keywords \(p. 44\)](#) have additional type requirements.):

- float
- double
- int
- long
- string
- boolean (values `true` and `false` must be lower case in your data)
- null

You can use `null` for `EVENT_VALUE` and `RECOMMENDATION_ID` reserved keywords, and interaction, user, and item metadata fields. Adding a `null` type to a field allows you to use imperfect data (for example, metadata with blank values), to generate personalized recommendations. The following example shows how to add a `null` type for a `GENRES` field.

```
{
  "name": "GENRES",
```

```
"type": [  
    "null",  
    "string"  
,  
    "categorical": true  
}
```

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.
- You can add at most 5 metadata fields for a Users schema and 50 metadata fields for an Items schema.
- If you add your own metadata field of type `string`, it must include the `categorical` attribute or the `textual` attribute (only Items schemas support fields with the `textual` attribute). Otherwise, Amazon Personalize won't use the field when training a model.

Reserved keywords

Reserved keywords are optional, non-metadata fields. These fields are considered reserved because you must define the fields as their required data type when you use them, and the keywords can't be used as values in your data. The following are reserved keywords:

- `EVENT_TYPE`: For Interactions datasets with one or more event types, such as both `click` and `download`, use an `EVENT_TYPE` field. You must define an `EVENT_TYPE` field as a `string`.
- `EVENT_VALUE`: For Interactions datasets that include value data for events, such as the percentage of a video a user watched, use an `EVENT_VALUE` field with type `float` and optionally `null`.
- `CREATION_TIMESTAMP`: For Items datasets with a timestamp for each item's creation date, use a `CREATION_TIMESTAMP` field with a type `long`. Amazon Personalize uses `CREATION_TIMESTAMP` data to calculate the age of an item and adjust recommendations accordingly. See [Creation timestamp data \(p. 50\)](#).
- `IMPRESSION`: For Interactions datasets with explicit impressions data, use an `IMPRESSION` field with type `String`. 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 \(p. 46\)](#).
- `RECOMMENDATION_ID`: For Interactions datasets that use previous recommendations as implicit impressions data, optionally use a `RECOMMENDATION_ID` field with type `String` and optionally type `null`.

You don't need to add a `RECOMMENDATION_ID` field for Amazon Personalize to use implicit impressions when generating recommendations. You can pass a `recommendationId` in a [PutEvents \(p. 330\)](#) operation without it. For more information see [Impressions data \(p. 46\)](#).

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 \(p. 207\)](#).

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 \(p. 47\)](#). For information on schema requirements see [Dataset and schema requirements \(p. 42\)](#).

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 \(p. 47\)](#). For information about importing historical interactions data, see [Preparing and importing data \(p. 54\)](#). For information about recording events in real-time using the [PutEvents \(p. 330\)](#) API, see [Recording events \(p. 81\)](#).

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 \(p. 170\)](#).

Topics

- [Required interaction data \(p. 45\)](#)
- [Contextual metadata \(p. 45\)](#)
- [Impressions data \(p. 46\)](#)
- [Interactions schema example \(p. 47\)](#)

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. The maximum number of metadata fields excluding reserved fields, such as IMPRESSION, is 5. Categorical values can have at most 1000 characters. Any interaction with a categorical value with more than 1,000 characters is dropped during a dataset import job and is not used in training.

For more information on minimum requirements and maximum data limits for an Interactions dataset, see [Service quotas \(p. 207\)](#).

Contextual metadata

If you use the [User-Personalization \(p. 98\)](#) or the [Personalized-Ranking \(p. 118\)](#) recipes, 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](#).

Impressions data

If you use the [User-Personalization \(p. 98\)](#) recipe, Amazon Personalize can model impressions data that you upload to an Interactions dataset. 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 uses impressions data to determine what items to include in exploration. *Exploration* is where recommendations include new items with less interactions data or relevance. The more frequently an item occurs in impressions data, the less likely it is that Amazon Personalize includes the item in exploration.

For information about the benefits of exploration see [User-Personalization \(p. 98\)](#). Amazon Personalize can model two types of impressions: [Implicit impressions \(p. 46\)](#) and [Explicit impressions \(p. 46\)](#).

Implicit 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 \(p. 340\)](#) and [GetPersonalizedRanking \(p. 336\)](#) operations) as input for future [PutEvents \(p. 330\)](#) requests. Amazon Personalize derives the implicit impressions based on your recommendation data.

For example, you might have an application that provides recommendations for streaming video. Your recommendation workflow using implicit impressions might be as follows:

1. You request video recommendations for one of your users using the Amazon Personalize [the section called "GetRecommendations" \(p. 340\)](#) API operation.
2. Amazon Personalize generates recommendations for the user using your model (solution version) and returns them with a `recommendationId` in the API response.
3. You show the video recommendations to your user in your application.
4. When your user interacts with (for example, clicks) a video, record the choice in a call to the [PutEvents \(p. 330\)](#) API and include the `recommendationId` as a parameter. For a code sample see [Recording impressions data \(p. 87\)](#).
5. Amazon Personalize uses the `recommendationId` to derive the impression data from the previous video recommendations, and then uses the impression data to guide exploration, where future recommendations include new videos with less interactions data or relevance.

For more information on recording events with implicit impression data, see [Recording impressions data \(p. 87\)](#).

Explicit impressions

Explicit impressions are impressions that you manually record and send to Amazon Personalize. Use explicit impressions to manipulate results from Amazon Personalize. The order of the items has no impact.

For example, you might have a shopping application that provides recommendations for shoes. If you only recommend shoes that are currently in stock, you can specify these items using explicit impressions. Your recommendation workflow using explicit impressions might be as follows:

1. You request recommendations for one of your users using the Amazon Personalize [the section called "GetRecommendations" \(p. 340\)](#) API.
2. Amazon Personalize generates recommendations for the user using your model (solution version) and returns them in the API response.
3. You show the user only the recommended shoes that are in stock.
4. For real-time incremental data import, when your user interacts with (for example, clicks) a pair of shoes, you record the choice in a call to the [PutEvents \(p. 330\)](#) API and list the recommended items that are in stock in the `impression` parameter. For a code sample see [Recording impressions data \(p. 87\)](#).

For importing impressions in historical interactions data, you can list explicit impressions in your csv file and separate each item with a ';' character. See [Formatting explicit impressions \(p. 64\)](#).

5. Amazon Personalize uses the impression data to guide exploration, where future recommendations include new shoes with less interactions data or relevance.

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`, and `IMPRESSION` 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 \(p. 42\)](#).

```
{  
    "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",  
                "null"  
            ]  
        },  
        {  
            "name": "LOCATION",  
            "type": "string",  
            "categorical": true  
        },  
        {  
            "name": "DEVICE",  
            "type": "string",  
            "categorical": true  
        }  
    ]  
}
```

```
"type": [
    "string",
    "null"
],
"categorical": true
},
{
    "name": "TIMESTAMP",
    "type": "long"
},
{
    "name": "IMPRESSION",
    "type": "string"
}
],
"version": "1.0"
}
```

For this schema, the first few lines of historical data in a CSV file might look like the following. Note that some values for EVENT_VALUE are null.

```
USER_ID,ITEM_ID,EVENT_TYPE,EVENT_VALUE,LOCATION,DEVICE,TIMESTAMP,IMPRESSION
35,73,click,,Ohio,Tablet,1586731606,73|70|17|95|96|92|55|45|16|97|56|54|33|94|36|10|5|43|
19|13|51|90|65|59|38
54,35,watch,0.75,Indiana,Cellphone,1586735164,35|82|78|57|20|63|1|90|76|75|49|71|26|24|25|
6|37|85|40|98|32|13|11|54|48
9,33,click,,Oregon,Cellphone,1586735158,68|33|62|6|15|57|45|24|78|89|90|40|26|91|66|31|47|
17|99|29|27|41|77|75|14
23,10,watch,0.25,California,Tablet,1586735697,92|89|36|10|39|77|4|27|79|18|83|16|28|68|78|
40|50|3|99|7|87|49|12|57|53
27,11,watch,0.55,Indiana,Tablet,1586735763,11|7|39|95|71|1|6|40|41|28|99|53|68|76|0|65|69|
36|22|42|34|67|24|20|66
...
...
```

Users dataset

A *Users dataset* stores metadata about your users. This might include information such as age, gender, or loyalty membership. A Users dataset is optional. You must at minimum create an [Interactions dataset \(p. 44\)](#).

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 \(p. 49\)](#). For information on schema requirements see [Dataset and schema requirements \(p. 42\)](#).

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 \(p. 49\)](#). For information about importing user data into a Users dataset, see [Preparing and importing data \(p. 54\)](#).

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 \(p. 170\)](#).

Note

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

Topics

- [Required user data \(p. 49\)](#)

- [Categorical metadata \(p. 49\)](#)
- [Users schema example \(p. 49\)](#)

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 (max length 256 characters). 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 \(p. 207\)](#).

Categorical metadata

With the [User-Personalization \(p. 98\)](#) or [Personalized-Ranking \(p. 118\)](#) recipes, Amazon Personalize uses categorical metadata, such as a user's gender or membership status, when identifying underlying patterns that reveal the most relevant items for your users.

With all recipes, you can import categorical metadata and use it to filter recommendations based on a user's attributes. For information about filtering recommendations see [Filtering recommendations \(p. 170\)](#).

To use categorical data, add a field of type `string` and set the field's `categorical` attribute to `true` in your schema. Then include the categorical data in your bulk CSV file and incremental item imports. For users with multiple categories, separate each value using the vertical bar, '|'. For an example of a schema with a categorical field see [Users schema example \(p. 49\)](#).

Categorical values can have at most 1,000 characters. If you have a user with a categorical value with more than 1,000 characters, your dataset import job will fail.

Users schema example

The following example shows how to structure a Users schema. The `USER_ID` field is required and the `AGE` and `GENDER` fields are metadata. At least one metadata field is required and you can add at most 5 metadata fields. For information about schema requirements see [Dataset and schema requirements \(p. 42\)](#).

```
{  
    "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"  
}
```

For this schema, the first few lines of historical data in a CSV file might look like the following.

```
USER_ID,AGE,GENDER
5,34,Male
6,56,Female
8,65,Male
...
...
```

Items dataset

An *Items dataset* stores metadata about your items. This might include information such as price, genre, or availability. An Items dataset is required for the [Similar-Items recipe \(p. 122\)](#) and optional for all other recipes.

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 an Items dataset schema see [Items schema example \(p. 51\)](#). For information on schema requirements see [Dataset and schema requirements \(p. 42\)](#).

This section provides information about required item data and the kinds of item data you can upload for training. For information about importing item data into an Items dataset, see [Preparing and importing data \(p. 54\)](#).

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 \(p. 170\)](#).

Topics

- [Required item data \(p. 50\)](#)
- [Creation timestamp data \(p. 50\)](#)
- [Categorical metadata \(p. 51\)](#)
- [Unstructured text metadata \(p. 51\)](#)
- [Items schema example \(p. 51\)](#)

Required item data

The data you provide for each item must match your Items dataset schema. At minimum, you must provide an Item ID for each item (max length 256 characters). Depending on your schema, item metadata can include empty/null values.

During model training, Amazon Personalize considers a maximum of 750,000 items. If you import more than 750,000 items, Amazon Personalize decides which items to include in training, with an emphasis on including new items (items you recently added 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 \(p. 207\)](#).

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 timestamp. If an item has no interaction data, its creation timestamp is set as the timestamp of the latest interaction in the training set and Amazon Personalize considers it a new item.

Categorical metadata

With the [User-Personalization \(p. 98\)](#), [Personalized-Ranking \(p. 118\)](#), or [Similar-Items \(p. 122\)](#) recipes, Amazon Personalize uses categorical metadata, such as an item's genre or color, when identifying underlying patterns that reveal the most relevant items for your users.

With all recipes, you can import categorical data and use it to filter recommendations based on an item's attributes. For information about filtering recommendations, see [Filtering recommendations \(p. 170\)](#).

To use categorical data, add a field of type `string` to your schema and set the field's categorical attribute to `true`. Then include the categorical data in your bulk CSV file and incremental item imports. For items with multiple categories, separate each value with the vertical bar, '|'. For an example of a schema with a categorical field see [Items schema example \(p. 51\)](#).

Categorical values can have a maximum of 1,000 characters. If you have an item with a categorical value with more than 1,000 characters, your dataset import job will fail.

Unstructured text metadata

With the [User-Personalization \(p. 98\)](#), [Personalized-Ranking \(p. 118\)](#), or [Similar-Items \(p. 122\)](#) recipes, Amazon Personalize can extract meaningful information from unstructured text metadata, such as product descriptions, product reviews, or movie synopses. Amazon Personalize uses unstructured text to identify relevant items for your users, particularly when items are new or have less interactions data. Include unstructured text data in your Items dataset to increase click-through rates and conversation rates for new items in your catalog.

To use unstructured data, add a field with type `string` to your Items schema and set the field's `textual` attribute to `true`. Then include the text data in your bulk CSV file and incremental item imports. For bulk CSV files, wrap the text in double quotes. Use the \ character to escape any double quotes or \ characters in your data. For an example of an Items schema with a field for unstructured text data, see [Items schema example \(p. 51\)](#). For information about importing data into Amazon Personalize, see [Preparing and importing data \(p. 54\)](#).

Unstructured text values can have at most 20,000 characters and text must be in English. Amazon Personalize truncates values that exceed the character limit to 20,000 characters.

Items schema example

The following example shows how to structure an Items schema. The `ITEM_ID` field is required. The `GENRE` field is categorical metadata and the `DESCRIPTION` field is textual metadata. At least one metadata field is required. You can add a maximum of 50 metadata fields. The `CREATION_TIMESTAMP` field is a reserved keyword. For information about schema requirements, see [Dataset and schema requirements \(p. 42\)](#).

```
{  
    "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"
},
{
  "name": "DESCRIPTION",
  "type": [
    "null",
    "string"
  ],
  "textual": true
},
],
"version": "1.0"
}
```

For this schema, the first few lines of historical data in a CSV file might look like the following.

```
ITEM_ID,GENRES,CREATION_TIMESTAMP,DESCRIPTION
1,Adventure|Animation|Children|Comedy|Fantasy,1570003267,"This is an animated movie that features action, comedy, and fantasy. Audience is children. This movie was released in 2004."
2,Adventure|Children|Fantasy,1571730101,"Critics say \"this film is perfect for children and grown ups too!\". It's an adventure movie with elements of fantasy. Audience is children. This movie was release in 2010."
3,Comedy|Romance,1560515629,"This is a romantic comedy. The movie was released in 1999. Audience is young women."
4,Comedy|Drama|Romance,1581670067,"This movie includes elements of both comedy and drama as well as romance. This movie was released in 2020."
...
...
```

Schema examples

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

- [Interactions schema example \(p. 47\)](#)
- [Users schema example \(p. 49\)](#)
- [Items schema example \(p. 51\)](#)

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.

```
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 \(p. 317\)](#) 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 \(p. 62\)](#).

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.

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.

This section provides information about importing historical data into Amazon Personalize. For information about recording live interactions data, see [Recording events \(p. 81\)](#).

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 \(p. 54\)](#).
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 \(p. 57\)](#).
3. Import your data:
 - Import bulk records stored in an Amazon S3 bucket using a dataset import job. See [Importing bulk records \(p. 62\)](#).
 - Import records incrementally using the AWS python SDK or AWS Command Line Interface (AWS CLI). See [Importing records incrementally \(p. 73\)](#).

Step 1: Creating a dataset group

A *dataset group* is container for Amazon Personalize components, including datasets, event trackers, solutions, filters, campaigns, and batch inference jobs. A dataset group organizes your resources into independent collections, so resources from one dataset group cannot influence resources in any other dataset group.

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 with the Amazon Personalize console, AWS Command Line Interface (AWS CLI) or AWS SDKs.

Topics

- [Creating a dataset group \(console\) \(p. 54\)](#)
- [Creating a dataset group \(AWS CLI\) \(p. 55\)](#)
- [Creating a dataset group \(AWS SDKs\) \(p. 55\)](#)

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/home> 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 \(console\) \(p. 57\)](#).

Creating a dataset group (AWS CLI)

Create a dataset group with the following command. For more information about the `CreateDatasetGroup` API operation, see [CreateDatasetGroup \(p. 227\)](#) in the API reference section.

```
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.

```
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",  
        "datasetGroupArn": "arn:aws:personalize:us-west-2:acct-id:dataset-group/  
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 \(AWS CLI\) \(p. 58\)](#).

Creating a dataset group (AWS SDKs)

The following code shows how to create a dataset group with the AWS SDK for Python (Boto3) or the SDK for Java 2.x. For more information about the API operation, see [CreateDatasetGroup \(p. 227\)](#) in the API reference section.

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_dataset_group(name = 'dataset group name')
dsg_arn = response['datasetGroupArn']

description = personalize.describe_dataset_group(datasetGroupArn = dsg_arn)
['datasetGroup']

print('Name: ' + description['name'])
print('ARN: ' + description['datasetGroupArn'])
print('Status: ' + description['status'])
```

SDK for Java 2.x

```
public static void createDatasetGroup(PersonalizeClient personalizeClient, String
datasetGroupName) {

    long waitInMilliseconds = 60 * 1000;

    try {
        CreateDatasetGroupRequest createDatasetGroupRequest =
CreateDatasetGroupRequest.builder()
            .name(datasetGroupName)
            .build();

        String datasetGroupArn =
personalizeClient.createDatasetGroup(createDatasetGroupRequest)
            .datasetGroupArn();

        long maxTime = Instant.now().getEpochSecond() + (15 * 60); // 15 minutes

        DescribeDatasetGroupRequest describeRequest =
DescribeDatasetGroupRequest.builder()
            .datasetGroupArn(datasetGroupArn)
            .build();

        String status = null;

        while (Instant.now().getEpochSecond() < maxTime) {

            status = personalizeClient.describeDatasetGroup(describeRequest)
                .datasetGroup()
                .status();

            System.out.println("DatasetGroup status:" + status);

            if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
                break;
            }

            try {
                Thread.sleep(waitInMilliseconds);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    } catch(PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}
```

}

The [DescribeDatasetGroup \(p. 272\)](#) operation returns the `datasetGroupArn` and the status of the operation. When the dataset group's status is ACTIVE, proceed to [Creating a dataset and a schema \(AWS SDKs\) \(p. 59\)](#).

Step 2: Creating a dataset and a schema

After you have completed [Step 1: Creating a dataset group \(p. 54\)](#), you are ready to create a dataset. *Datasets* are Amazon Personalize containers for data. 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 create datasets with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs. For information about the different types of datasets, and dataset and schema requirements, see [Datasets and schemas \(p. 42\)](#).

Topics

- [Creating a dataset and a schema \(console\) \(p. 57\)](#)
- [Creating a dataset and a schema \(AWS CLI\) \(p. 58\)](#)
- [Creating a dataset and a schema \(AWS SDKs\) \(p. 59\)](#)

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 schemas \(p. 42\)](#).

Note

If you just completed [Step 1: Creating a dataset group \(p. 54\)](#) 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/home> 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 \(p. 54\)](#). 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 \(p. 42\)](#) 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 \(p. 62\)](#) 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 \(p. 238\)](#) operation. Then create a dataset using the [CreateDataset \(p. 221\)](#) operation. For information on Amazon Personalize datasets and schema requirements, see [Datasets and schemas \(p. 42\)](#).

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 schemas \(p. 42\)](#).

```
{  
    "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 \(p. 238\)](#).

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

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

```
{  
    "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 \(AWS CLI\) \(p. 55\)](#) 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 \(p. 221\)](#).

```
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:acct-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 \(p. 62\)](#).

Creating a dataset and a schema (AWS SDKs)

To create a dataset and a schema using the AWS SDKs, you first define a schema in [Avro format](#) and add it to Amazon Personalize using the [CreateSchema \(p. 238\)](#) operation. Then create a dataset using the [CreateDataset \(p. 221\)](#) operation. For information on Amazon Personalize datasets and schema requirements, see [Datasets and schemas \(p. 42\)](#).

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 schemas \(p. 42\)](#).

```
{
  "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 \(p. 238\)](#) API operation.

SDK for Python (Boto3)

Use the following `create_schema` method to create a schema. 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 )
```

SDK for Java 2.x

Use the following `createSchema` method to create a schema. Pass the following as parameters: a `PersonalizeClient`, the name for your schema, and the file path for your schema JSON file.

```
public static String createSchema(PersonalizeClient personalizeClient, String
schemaName, String filePath) {

    String schema = null;

    try {
        schema = new String(Files.readAllBytes(Paths.get(filePath)));
    } catch (IOException e) {
        System.out.println(e.getMessage());
    }

    try {
        CreateSchemaRequest createSchemaRequest = CreateSchemaRequest.builder()
            .name(schemaName)
            .schema(schema)
            .build();

        String schemaArn =
personalizeClient.createSchema(createSchemaRequest).schemaArn();
        System.out.println("Schema arn: " + schemaArn);

        return schemaArn;
    } catch(PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

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 \(p. 221\)](#) operation. For information about the different types of datasets, see [Datasets and schemas \(p. 42\)](#).

SDK for Python (Boto3)

Use the following `create_dataset` method to create an Amazon Personalize dataset. Specify the `datasetGroupArn` returned in [Creating a dataset group \(AWS SDKs\) \(p. 55\)](#). Use the `schemaArn` created in the previous step. Replace `dataset_type` with the type of dataset you are uploading (Interactions, Users, or Items).

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_dataset(
    name = 'datase_name',
    schemaArn = 'schema_arn',
    datasetGroupArn = 'dataset_group_arn',
    datasetType = 'dataset_type'
)

print ('Dataset Arn: ' + response['datasetArn'])
```

SDK for Java 2.x

Use the following `createDataset` method to create an Amazon Personalize dataset. Pass the following as parameters: a `PersonalizeClient`, the name for your dataset, the `schemaArn` created in the previous step, your dataset group ARN, and the dataset type (Interactions, Users, or Items).

```
public static String createDataset(PersonalizeClient personalizeClient,
                                    String datasetName,
                                    String datasetGroupArn,
                                    String datasetType,
                                    String schemaArn) {
    try {
        CreateDatasetRequest request = CreateDatasetRequest.builder()
            .name(datasetName)
            .datasetGroupArn(datasetGroupArn)
            .datasetType(datasetType)
            .schemaArn(schemaArn).build();

        String datasetArn = personalizeClient.createDataset(request).datasetArn();
        System.out.println("Dataset " + datasetName + " created. Dataset ARN: " +
                           datasetArn);

        return datasetArn;
    } catch(PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

After you have created a dataset, you are ready to import your training data. See [Step 3: Importing your data \(p. 62\)](#).

Step 3: Importing your data

When you have completed [Step 1: Creating a dataset group \(p. 54\)](#) and [Step 2: Creating a dataset and a schema \(p. 57\)](#), 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 \(p. 81\)](#)

Topics

- [Importing bulk records \(p. 62\)](#)
- [Importing records incrementally \(p. 73\)](#)

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 \(p. 54\)](#) and [Step 2: Creating a dataset and a schema \(p. 57\)](#), 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 \(p. 62\)](#)
- [Uploading to an Amazon S3 bucket \(p. 64\)](#)
- [Importing bulk records with a dataset import job \(p. 68\)](#)

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 \(p. 64\)](#).

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.

```
{
  "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 explicit impressions

If you use the [User-Personalization \(p. 98\)](#) recipe, you can record and upload 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. To upload impressions data in a bulk data import, you manually record each item ID, separating the values with a vertical bar, '|', character as part of your historical interactions data. For more information on impressions data, see [Impressions data \(p. 46\)](#).

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

EVENT_TYPE	IMPRESSION	ITEM_ID	TIMESTAMP	USER_ID
click	73 70 17 95 96	73	1586731606	USER_1
click	35 82 78 57 20 63 1 90 76 75 49 71 26 24 25 6	35	1586735164	USER_2
...

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 \(p. 64\)](#).

Uploading to an Amazon S3 bucket

After you format your historical input data (see [Formatting your input data \(p. 62\)](#)), 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 \(p. 9\)](#) 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 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 \(p. 65\)](#).

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 service-linked role permission to use your key. You must also add Amazon Personalize as a Principle in your AWS KMS key policy. 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 \(p. 62\)](#).

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 \(p. 9\)](#) 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 service role for Amazon Personalize \(p. 10\)](#)) that allows access to your Amazon S3 bucket. For more information, see [Attaching an Amazon S3 policy to your Amazon Personalize service role \(p. 65\)](#).
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 Amazon S3 bucket \(p. 67\)](#).
4. If you are using AWS Key Management Service (AWS KMS) for encryption, you must give your IAM user and Amazon Personalize IAM service-linked role permission to use your key. You must also add Amazon Personalize as a Principle in your AWS KMS key policy. 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.

Topics

- [Attaching an Amazon S3 policy to your Amazon Personalize service role \(p. 65\)](#)
- [Attaching an Amazon Personalize access policy to your Amazon S3 bucket \(p. 67\)](#)

Attaching an Amazon S3 policy to your Amazon Personalize service role

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

1. Sign in to the IAM console (<https://console.aws.amazon.com/iam/>).
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 [Service-linked role policy for batch workflows \(p. 66\)](#).

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

```

        "Action": [
            "s3:GetObject",
            "s3>ListBucket"
        ],
        "Resource": [
            "arn:aws:s3:::bucket-name",
            "arn:aws:s3:::bucket-name/*"
        ]
    }
}

```

4. Choose **Next: Tags**. Optionally add any tags and choose **Review**.
5. Give the policy a name.
6. (Optional) For **Description**, enter a short sentence describing this policy, for example, **Allow Amazon Personalize to access its Amazon 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 service role for Amazon Personalize \(p. 10\)](#).
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 Amazon S3 bucket \(p. 67\)](#).

[Service-linked role policy for batch workflows](#)

To complete a batch workflow, 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 \(p. 163\)](#).

```

{
    "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/*"
            ]
        }
    ]
}

```

[Service-linked role policy for exporting a dataset](#)

To export a dataset, your Amazon Personalize service-linked role needs permission to use the **PutObject** and **ListBucket** Actions on your Amazon S3 bucket. The following example policy grants

Amazon Personalize `PutObject` and `ListBucket` permissions. Replace `bucket-name` with the name of your bucket and attach the policy to your service-linked role. For information about attaching policies to a service-linked IAM role see [Attaching an Amazon S3 policy to your Amazon Personalize service role \(p. 65\)](#).

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

Attaching an Amazon Personalize access policy to your Amazon 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 [Amazon S3 bucket policy for batch workflows \(p. 67\)](#).

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

Amazon 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 \(p. 163\)](#).

```
{
    "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/*"
            ]
        }
    ]
}
```

Amazon S3 bucket policy for exporting a dataset

To export a dataset, Amazon Personalize needs permission to use the `PutObject` and `ListBucket` Actions on your Amazon S3 bucket. The following example policy grants the Amazon Personalize principle `PutObject` and `ListBucket` permissions. Replace `bucket-name` with the name of your bucket and attach the policy to your bucket. For information on adding an Amazon S3 bucket policy to a bucket, see [How Do I Add an S3 Bucket Policy?](#) in the Amazon Simple Storage Service User Guide.

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

Importing bulk records with a dataset import job

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 formatted your input data (see [Formatting your input data \(p. 62\)](#)) and uploaded it to an Amazon Simple Storage Service (Amazon S3) bucket (see [Uploading to an Amazon S3 bucket \(p. 64\)](#)), 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 SDKs.

Topics

- [Importing bulk records \(console\) \(p. 69\)](#)
- [Importing bulk records \(AWS CLI\) \(p. 69\)](#)
- [Importing bulk records \(AWS SDKs\) \(p. 71\)](#)

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 \(p. 57\)](#), skip to step 5.

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> 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 service role for Amazon Personalize \(p. 10\)](#).
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 \(p. 96\)](#).

Importing bulk records (AWS CLI)

To import bulk records using the AWS CLI, create a dataset import job using the [CreateDatasetImportJob \(p. 230\)](#) command.

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 \(p. 57\)](#) 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 service role for Amazon Personalize \(p. 10\)](#). For more information about the operation, see [CreateDatasetImportJob \(p. 230\)](#).

```
aws personalize create-dataset-import-job \
--job-name dataset import job name \
--dataset-arn dataset arn \
--data-source dataLocation=s3://<bucketname>/filename \
--role-arn roleArn
```

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

```
{  
    "datasetImportJobArn": "arn:aws:personalize:us-west-2:acct-id:dataset-import-job/  
DatasetImportJobName"  
}
```

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 \(p. 274\)](#).

```
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",  
        "datasetImportJobArn": "arn:aws:personalize:us-west-2:acct-id:dataset-import-job/  
DatasetImportJobArn",  
        "datasetArn": "arn:aws:personalize:us-west-2:acct-id:dataset/DatasetGroupName/  
INTERACTIONS",  
        "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 \(p. 96\)](#).

Importing bulk records (AWS SDKs)

To add data to your dataset, create and run a dataset import job using the [CreateDatasetImportJob \(p. 230\)](#) operation. The following code shows how to create a dataset import job using the SDK for Python (Boto3) or SDK for Java 2.x.

SDK for Python (Boto3)

Specify the `datasetGroupArn` and set the `dataLocation` to the path to your Amazon S3 bucket where you stored the training data.

For the `roleArn`, specify the AWS Identity and Access Management (IAM) role that gives Amazon Personalize permissions to access your S3 bucket. See [Creating an IAM service role for Amazon Personalize \(p. 10\)](#).

```
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'])
```

SDK for Java 2.x

Use the following `createPersonalizeDatasetImportJob` method to create a dataset import job. Pass the following as parameters: an Amazon Personalize service client, the dataset group's ARN (Amazon Resource Name), a name for the job, the dataset ARN, the Amazon S3 bucket path (`s3://<bucketname>/<file_name.csv>`) where you stored your training data, and your service-linked role's ARN (see [Creating an IAM service role for Amazon Personalize \(p. 10\)](#)).

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

```
public static String createPersonalizeDatasetImportJob(PersonalizeClient  
personalizeClient,  
                                                    String jobName,  
                                                    String datasetArn,  
                                                    String s3BucketPath,  
                                                    String roleArn) {  
  
    long waitInMilliseconds = 60 * 1000;  
    String status;  
    String datasetImportJobArn;  
  
    try {
```

```

DataSource importDataSource = DataSource.builder()
    .dataLocation(s3BucketPath)
    .build();

CreateDatasetImportJobRequest createDatasetImportJobRequest =
CreateDatasetImportJobRequest.builder()
    .datasetArn(datasetArn)
    .dataSource(importDataSource)
    .jobName(jobName)
    .roleArn(roleArn)
    .build();

datasetImportJobArn =
personalizeClient.createDatasetImportJob(createDatasetImportJobRequest)
    .datasetImportJobArn();

DescribeDatasetImportJobRequest describeDatasetImportJobRequest =
DescribeDatasetImportJobRequest.builder()
    .datasetImportJobArn(datasetImportJobArn)
    .build();

long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

while (Instant.now().getEpochSecond() < maxTime) {

    DatasetImportJob datasetImportJob = personalizeClient
        .describeDatasetImportJob(describeDatasetImportJobRequest)
        .datasetImportJob();

    status = datasetImportJob.status();
    System.out.println("Dataset import job status: " + status);

    if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
        break;
    }
    try {
        Thread.sleep(waitInMilliseconds);
    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
    }
}
return datasetImportJobArn;

} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return "";
}
}

```

The response from the [DescribeDatasetImportJob \(p. 274\)](#) 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 \(p. 308\)](#) to list the datasets in a dataset group and [DeleteDataset \(p. 250\)](#) 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 \(p. 96\)](#).

Importing records incrementally

After you have completed [Step 1: Creating a dataset group \(p. 54\)](#) and [Step 2: Creating a dataset and a schema \(p. 57\)](#), you can incrementally import one or more new records, including interaction *events*, users, or items, to an existing dataset. Incrementally importing records allows you to import one or more records into your Amazon Personalize datasets as your catalog grows. If you have a large amount of historical records, we recommend that you first import data in bulk and then import data incrementally as necessary. See [Importing bulk records \(p. 62\)](#).

Filter updates for incremental record imports

Amazon Personalize updates any filters you created in the dataset group with your new interaction, 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 records influence recommendations as follows:

- For *new events*, Amazon Personalize immediately uses historical and real-time interaction events between a user and existing items (items you included in the data you used to train the latest model) when generating recommendations for the same user. Historical events that you import using the Amazon Personalize console and events that you record in real-time influence recommendations in the same way. For more information, see [How real-time events influence recommendations \(p. 82\)](#).
- For *new items*, if you trained the solution version with User-Personalization, Amazon Personalize automatically updates the model every two hours. After each update, the new items can be included in recommendations with exploration. For information about exploration see [User-Personalization recipe \(p. 98\)](#).

For any other recipe, you must retrain the model for the new items to be included in recommendations.

- For *new users*, recommendations will initially be only for popular items. Starting with the first event, user recommendations will be more relevant as you record events. For more information, see [Recording events \(p. 81\)](#).

Topics

- [Importing interactions incrementally \(p. 73\)](#)
- [Importing users incrementally \(p. 75\)](#)
- [Importing items incrementally \(p. 77\)](#)

Importing interactions incrementally

To import interaction *events* incrementally, you create an *event tracker* and then import one or more events into your Interactions dataset. You can incrementally import historical interaction events using the Amazon Personalize console, or import historical or real-time events using the AWS Command Line Interface (AWS CLI), or the AWS SDKs.

This section includes information about importing events using the Amazon Personalize console. We recommend using the Amazon Personalize console to incrementally import *only* historical events. For information about using the AWS CLI or the AWS SDKs to record events in real-time, see [Recording events \(p. 81\)](#).

For information about how Amazon Personalize updates filters for new records and how new records influence recommendations, see [Importing records incrementally \(p. 73\)](#).

Topics

- [Creating an event tracker \(console\) \(p. 74\)](#)
- [Importing events incrementally \(console\) \(p. 74\)](#)

Creating an event tracker (console)

Note

If you've created an event tracker, you can skip to [Importing events incrementally \(console\) \(p. 74\)](#).

Before you can incrementally import an event to an Interactions dataset, you must create an *event tracker* for the dataset group.

To create an event tracker (console)

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> and sign in to your account.
2. On the **Dataset groups** page, choose the dataset group with the Interactions dataset that you want to import events to.
3. On the **Dashboard** for the dataset group, in **Install event ingestion SDK**, choose **Start**.
4. On the **Configure tracker** page, in **Tracker configurations**, for **Tracker name**, provide a name for the event tracker, and choose **Next**.
5. The **Install the SDK** page shows the **Tracking ID** for the new event tracker and instructions for using AWS Amplify or AWS Lambda to stream event data.

You can ignore this information because you're using the Amazon Personalize console to upload event data. If you want to stream event data using AWS Amplify or AWS Lambda in the future, you can view this information by choosing the event tracker on the **Event trackers** page.

6. Choose **Finish**. You can now incrementally import events using the console (see [Importing events incrementally \(console\) \(p. 74\)](#)) or record events in real time using the **PutEvents** operation (see [Recording events \(p. 81\)](#)).

Importing events incrementally (console)

After you create an event tracker, you can incrementally import events to an Interactions dataset. This procedure assumes you have already created an Interactions dataset. For information about creating datasets, see [Step 2: Creating a dataset and a schema \(p. 57\)](#).

To import events incrementally (console)

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> and sign in to your account.
2. On the **Dataset groups** page, choose the dataset group with the Interactions dataset that you want to import events to.
3. In the navigation pane, choose **Datasets**.
4. On the **Datasets** page, choose the Interactions dataset.
5. At the top right of the dataset details page, choose **Modify dataset**, and choose **Create record**.
6. In **Create user-item interaction record(s)** page, for **Record input**, enter the event details in JSON format. The event's field names and values must match the schema that you used when you created the Interactions dataset. Amazon Personalize provides a JSON template with field names and data types from this schema. You can import up to 10 events at a time.
7. Choose **Create record(s)**. In **Response**, the result of the import is listed and a success or failure message is displayed.

Importing users incrementally

After you create a Users dataset, you can incrementally import one or more new users into the dataset. Incrementally importing users allows you to keep your Users dataset current as your business grows. If you have a large amount of new users, we recommend that you first import data in bulk and then import user data incrementally as necessary. See [Importing bulk records \(p. 62\)](#).

You can use the Amazon Personalize console, the AWS Command Line Interface (AWS CLI), or AWS SDKs to import users. If you import a user with the same `userId` as a user that's already in your Users dataset, Amazon Personalize replaces the user with the new one. You can import up to 10 users at a time.

For information about how Amazon Personalize updates filters for new records and how new records influence recommendations, see [Importing records incrementally \(p. 73\)](#).

Topics

- [Importing users incrementally \(console\) \(p. 75\)](#)
- [Importing users incrementally \(AWS CLI\) \(p. 75\)](#)
- [Importing users incrementally \(AWS SDKs\) \(p. 76\)](#)

Importing users incrementally (console)

You can import up to 10 users at a time. This procedure assumes you have already created a Users dataset. For information about creating datasets, see [Step 2: Creating a dataset and a schema \(p. 57\)](#).

To import users incrementally (console)

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> and sign in to your account.
2. On the **Dataset groups** page, choose the dataset group with the Users dataset that you want to import the user to.
3. In the navigation pane, choose **Datasets**.
4. On the **Datasets** page, choose the Users dataset.
5. On the dataset details page, at the top right, choose **Modify dataset** and choose **Create record**.
6. On the **Create user record(s)** page, for record input, enter the user details in JSON format. The user's field names and values must match the schema you used when you created the Users dataset. Amazon Personalize provides a JSON template with field names and data types from this schema.
7. Choose **Create record(s)**. In **Response**, the result of the import is listed and a success or failure message is displayed.

Importing users incrementally (AWS CLI)

Add one or more users to your Users dataset with the [PutUsers \(p. 334\)](#) operation. You can import up to 10 users with a single `PutUsers` call. This section assumes that you have already created an Users dataset. For information about creating datasets, see [Step 2: Creating a dataset and a schema \(p. 57\)](#).

Use the following `put-users` command to add one or more users with the AWS CLI. Replace `dataset_arn` with the Amazon Resource Name (ARN) of your dataset and `user_id` with the ID of the user. If an user with the same `userId` is already in your Users dataset, Amazon Personalize replaces it with the new one.

For `properties`, for each field in your Users dataset, replace the `propertyName` with the field name from your schema in camel case. For example, `GENDER` would be `gender` and `MEMBERSHIP_TYPE` would be `membershipType`. Replace `user_data` with the data for the user. For categorical string data, to

include multiple categories for a single property, separate each category with a pipe (|). For example `\\"Premium Class|Legacy Member\\".`

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

Importing users incrementally (AWS SDKs)

Add one or more users to your Users dataset with the [PutUsers \(p. 334\)](#) operation. You can import up to 10 users with a single PutUsers call. This section assumes that you have already created a Users dataset. For information about creating datasets, see [Step 2: Creating a dataset and a schema \(p. 57\)](#).

The following code shows how to add one or more users to your Users dataset with the AWS SDK for Python (Boto3) or the AWS SDK for Java 2.x.

SDK for Python (Boto3)

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

For `properties`, for each field in your Users dataset, replace the `propertyName` with the field name from your schema in camel case. For example, `GENDER` would be `gender` and `MEMBERSHIP_TYPE` would be `membershipType`. Replace `user data` with the data for the user. For categorical string data, to include multiple categories for a single property separate each category with a pipe (|). For example `\\"Premium Class|Legacy Member\\".`

```
import boto3

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

personalize_events.put_users(
    datasetArn = 'dataset arn',
    users = [
        {
            'userId': 'user ID',
            'properties': "{\"propertyName\": \"\user data\"}"
        },
        {
            'userId': 'user ID',
            'properties': "{\"propertyName\": \"\user data\"}"
        }
    ]
)
```

SDK for Java 2.x

The following `putUsers` method shows how to add two users to a Users dataset with the SDK for Java 2.x. If a user with the same `userId` is already in your Users dataset, Amazon Personalize replaces it with the new one. In this example, each user has a single property. If your Users dataset schema has additional fields, modify the code to use additional property and value parameters.

For each property name parameter, pass the field name from your schema in camel case. For example, `GENDER` would be `gender` and `MEMBERSHIP_TYPE` would be `membershipType`. For each

property value parameter, pass the data for the user. For categorical string data, to include multiple categories for a single property separate each category with a pipe (|). For example "Premium class|Legacy Member".

```
public static int putUsers(PersonalizeEventsClient personalizeEventsClient,
                           String datasetArn,
                           String user1Id,
                           String user1PropertyName,
                           String user1PropertyValue,
                           String user2Id,
                           String user2PropertyName,
                           String user2PropertyValue) {

    int responseCode = 0;
    ArrayList<User> users = new ArrayList<>();

    try {
        User user1 = User.builder()
            .userId(user1Id)
            .properties(String.format("{\"%1$s\": \"%2$s\"}", user1PropertyName,
user1PropertyValue))
            .build();

        users.add(user1);

        User user2 = User.builder()
            .userId(user2Id)
            .properties(String.format("{\"%1$s\": \"%2$s\"}", user2PropertyName,
user2PropertyValue))
            .build();

        users.add(user2);

        PutUsersRequest putUsersRequest = PutUsersRequest.builder()
            .datasetArn(datasetArn)
            .build();

        responseCode =
personalizeEventsClient.putUsers(putUsersRequest).sdkHttpResponse().statusCode();
        System.out.println("Response code: " + responseCode);
        return responseCode;
    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return responseCode;
}
```

Importing items incrementally

After you create an Items dataset, you can incrementally import one or more new items into the dataset. Incrementally importing items allows you to keep your Items dataset current as your catalog grows. If you have a large amount of new items, we recommend that you first import data in bulk and then import item data incrementally as necessary. See [Importing bulk records \(p. 62\)](#).

You can use the Amazon Personalize console, the AWS Command Line Interface (AWS CLI), or AWS SDKs to import items. If you import an item with the same `itemId` as an item that's already in your Items dataset, Amazon Personalize replaces it with the new item. You can import up to 10 items at a time.

For information about how Amazon Personalize updates filters for new records and how new records influence recommendations, see [Importing records incrementally \(p. 73\)](#).

Topics

- [Importing items incrementally \(console\) \(p. 78\)](#)
- [Importing items incrementally \(AWS CLI\) \(p. 78\)](#)
- [Importing items incrementally \(AWS SDKs\) \(p. 79\)](#)

Importing items incrementally (console)

You can import up to 10 items to an Items dataset at a time. This procedure assumes that you have already created an Items dataset. For information about creating datasets, see [Step 2: Creating a dataset and a schema \(p. 57\)](#).

To import items incrementally (console)

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> and sign in to your account.
2. On the **Dataset groups** page, choose the dataset group with the Items dataset that you want to import the items to.
3. In the navigation pane, choose **Datasets**.
4. On the **Datasets** page, choose the Items dataset.
5. At the top right of the dataset details page, choose **Modify dataset**, and then choose **Create record**.
6. In **Create item record(s)** page, for **Record input**, enter the item details in JSON format. The item's field names and values must match the schema you used when you created the Items dataset. Amazon Personalize provides a JSON template with field names and data types from this schema.
7. Choose **Create record(s)**. In **Response**, the result of the import is listed and a success or failure message is displayed.

Importing items incrementally (AWS CLI)

Add one or more items to your Items dataset using the [PutItems \(p. 332\)](#) operation. You can import up to 10 items with a single PutItems call. This section assumes that you have already created an Items dataset. For information about creating datasets, see [Step 2: Creating a dataset and a schema \(p. 57\)](#).

Use the following put-items command to add one or more items with the AWS CLI. Replace dataset arn with the Amazon Resource Name (ARN) of your dataset and item id with the ID of the item. If an item with the same itemId is already in your Items dataset, Amazon Personalize replaces it 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, GENRES would be genres and CREATION_TIMESTAMP would be creationTimestamp. Replace item data with the data for the item. CREATION_TIMESTAMP data must be in [Unix epoch time format \(p. 64\)](#) and in seconds. For categorical string data, to include multiple categories for a single property, separate each category with a pipe (|). For example \\"Horror|Action\\".

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

Importing items incrementally (AWS SDKs)

Add one or more items to your Items dataset using the [PutItems \(p. 332\)](#) operation. You can import up to 10 items with a single PutItems call. This section assumes that you have already created an Items dataset. For information about creating datasets, see [Step 2: Creating a dataset and a schema \(p. 57\)](#).

The following code shows how to add one or more items to your Items dataset with the AWS SDK for Python (Boto3) or the AWS SDK for Java 2.x.

SDK for Python (Boto3)

Replace `dataset_arn` with the Amazon Resource Name (ARN) of your dataset and `item_ID` with the ID of the item. If an item with the same `itemId` is already in your Items dataset, Amazon Personalize replaces it 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, `GENRES` would be `genres` and `CREATION_TIMESTAMP` would be `creationTimestamp`. Replace `item_data` with the data for the item. `CREATION_TIMESTAMP` data must be in [Unix epoch time format \(p. 64\)](#) and in seconds. For categorical string data, to include multiple categories for a single property, separate each category with a pipe (|). For example `\\"Horror|Action\\"`.

```
import boto3

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

personalize_events.put_items(
    datasetArn = 'dataset_arn',
    items = [
        {
            'itemId': 'item_ID',
            'properties': "{\"propertyName\": \"item_data\"}"
        },
        {
            'itemId': 'item_ID',
            'properties': "{\"propertyName\": \"item_data\"}"
        }
    ]
)
```

SDK for Java 2.x

The following `putItems` method shows how to add two items to an Items dataset with the SDK for Java 2.x. If an item with the same `itemId` is already in your Items dataset, Amazon Personalize replaces it with the new one. In this example, each item has a single property. If your Items dataset schema has additional fields, modify the code to use additional property and value parameters.

For each property name parameter, pass the field name from your schema in camel case. For example, `GENRES` would be `genres` and `CREATION_TIMESTAMP` would be `creationTimestamp`. For each property value parameter, pass the data for the item. `CREATION_TIMESTAMP` data must be in [Unix epoch time format \(p. 64\)](#) and in seconds. For categorical string data, to include multiple categories for a single property, separate each category with a pipe (|). For example `"Horror|Action"`.

```
public static int putItems(PersonalizeEventsClient personalizeEventsClient,
                           String datasetArn,
                           String itemId,
                           String item1PropertyName,
                           String item1PropertyValue,
                           String item2Id,
                           String item2PropertyName,
                           String item2PropertyValue) {
```

```
int responseCode = 0;
ArrayList<Item> items = new ArrayList<>();

try {
    Item item1 = Item.builder()
        .itemId(item1Id)
        .properties(String.format("{\"%1$s\": \"%2$s\"}",
            item1PropertyName, item1PropertyValue))
        .build();

    items.add(item1);

    Item item2 = Item.builder()
        .itemId(item2Id)
        .properties(String.format("{\"%1$s\": \"%2$s\"}",
            item2PropertyName, item2PropertyValue))
        .build();

    items.add(item2);

    PutItemsRequest putItemsRequest = PutItemsRequest.builder()
        .datasetArn(datasetArn)
        .items(items)
        .build();

    responseCode =
personalizeEventsClient.putItems(putItemsRequest).sdkHttpResponse().statusCode();
    System.out.println("Response code: " + responseCode);
    return responseCode;
}

} catch (PersonalizeEventsException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
return responseCode;
}
```

Recording events

Amazon Personalize can make recommendations based on real-time *event* data only, historical event data only (see [Importing bulk records \(p. 62\)](#)), or a mixture of both. Record events in real-time so Amazon Personalize can learn from your user's most recent activity and update recommendations as they use your application. This keeps your interactions data fresh and improves the relevance of Amazon Personalize recommendations.

You can record real-time events using the AWS SDKs, AWS Amplify or AWS Command Line Interface (AWS CLI). When you record events, Amazon Personalize appends the event data to the Interactions dataset in your dataset group.

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](#)

Topics

- Requirements for recording events and training a model ([p. 81](#))
- How real-time events influence recommendations ([p. 82](#))
- Creating an event tracker ([p. 82](#))
- PutEvents operation ([p. 84](#))
- Recording impressions data ([p. 87](#))
- Event metrics ([p. 89](#))
- Events and solutions ([p. 89](#))
- Sample Jupyter notebook ([p. 89](#))

Requirements for recording events and training a model

To record events, you need the following:

- A dataset group that includes an `Interactions` dataset, which can be empty. If you went through the [Getting started \(p. 14\)](#) guide, you can use the same dataset group and dataset that you created. For information on creating a dataset group and a dataset, see [Preparing and importing data \(p. 54\)](#).
- An event tracker.
- A call to the `PutEvents` ([p. 330](#)) 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 real-time 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, Amazon Personalize adds the new recorded event data 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 User-Personalization, Amazon Personalize automatically updates the model every two hours, and after each update the new items influence recommendations. See [User-Personalization recipe \(p. 98\)](#).

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.

For new, anonymous users (users without a userId), Amazon Personalize uses the sessionId you pass in the [PutEvents \(p. 330\)](#) operation to associate events with the user before they log in. This creates a continuous event history that includes events that occurred when the user was anonymous.

Creating an event tracker

An [event tracker](#) specifies a destination dataset group for new event data. To create an event tracker you call the [CreateEventTracker \(p. 233\)](#) API operation and pass as a parameter the Amazon Resource Name (ARN) of the dataset group that contains the target Interactions dataset. For instructions on creating an event tracker using the Amazon Personalize console, see [Creating an event tracker \(console\) \(p. 74\)](#).

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

You can create only one event tracker for a dataset group.

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'])
```

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

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

AWS CLI

```
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:

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

SDK for Java 2.x

```
public static String createEventTracker(PersonalizeClient personalizeClient,  
                                         String eventTrackerName,  
                                         String datasetGroupArn) {  
  
    String eventTrackerId = null;  
    String eventTrackerArn = null;  
    long maxTime = 3 * 60 * 60;  
    long waitInMilliseconds = 30 * 1000;  
    String status;  
  
    try {  
        CreateEventTrackerRequest createEventTrackerRequest =  
CreateEventTrackerRequest.builder()  
            .name(eventTrackerName)  
            .datasetGroupArn(datasetGroupArn)  
            .build();  
  
        CreateEventTrackerResponse createEventTrackerResponse =  
            personalizeClient.createEventTracker(createEventTrackerRequest);  
  
        eventTrackerArn = createEventTrackerResponse.eventTrackerArn();  
        eventTrackerId = createEventTrackerResponse.trackingId();  
  
        System.out.println("Event tracker ARN: " + eventTrackerArn);  
        System.out.println("Event tracker ID: " + eventTrackerId);  
  
        maxTime = Instant.now().getEpochSecond() + maxTime;  
  
        DescribeEventTrackerRequest describeRequest =  
DescribeEventTrackerRequest.builder()  
            .eventTrackerArn(eventTrackerArn)  
            .build();  
  
        while (Instant.now().getEpochSecond() < maxTime) {  
  
            status =  
personalizeClient.describeEventTracker(describeRequest).eventTracker().status();  
            System.out.println("EventTracker status: " + status);  
    }  
}
```

```

        if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return eventTrackerId;
}
catch (PersonalizeException e){
    System.out.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return eventTrackerId;
}

```

PutEvents operation

To record events, you call the [PutEvents \(p. 330\)](#) operation. The following example shows a PutEvents operation 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. For example, you could generate a universally unique identifier (UUID) and use this value for the `sessionId`. 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. We recommend having at most one user mapped to one `sessionId`.

The event list is an array of [Event \(p. 429\)](#) 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 `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 \(p. 42\)](#).

Corresponding interactions schema

```

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

```

Code example

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': 'eventTypePlaceholder',
        'itemId': 'ITEM_ID'
    }]
)

```

AWS CLI

```

aws personalize-events put-events \
--tracking-id tracking_id \
--user-id USER_ID \
--session-id session_id \
--event-list '[{
    "sentAt": "TIMESTAMP",
    "eventType": "eventTypePlaceholder",
    "itemId": "ITEM_ID"
}]'

```

SDK for Java 2.x

```

public static void putEvents(PersonalizeEventsClient personalizeEventsClient,
                             String trackingId,
                             String sessionId,
                             String userId,
                             String itemId) {

    try {
        Event event = Event.builder()
            .sentAt(Instant.ofEpochMilli(System.currentTimeMillis() + 10 * 60 * 1000))
            .itemId(itemId)
            .eventType("typePlaceholder")
            .build();

        PutEventsRequest putEventsRequest = PutEventsRequest.builder()
            .trackingId(trackingId)
            .userId(userId)
            .sessionId(sessionId)
            .eventList(event)
            .build();

        int responseCode = personalizeEventsClient.putEvents(putEventsRequest)
            .sdkHttpResponse()
            .statusCode();
        System.out.println("Response code: " + responseCode);

    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}

```

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. 240) 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

```
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\"}"
}]'
```

SDK for Java 2.x

```
public static void putMultipleEvents(PersonalizeEventsClient personalizeEventsClient,
                                      String trackingId,
                                      String sessionId,
                                      String userId,
                                      String eventType,
                                      Float event1Value,
                                      String event1ItemId,
```

```

        int event1NumRatings,
        String event2Type,
        Float event2Value,
        String event2ItemId,
        int event2NumRatings) {

    ArrayList<Event> eventList = new ArrayList<Event>();

    try {
        Event event1 = Event.builder()
            .eventType(event1Type)
            .sentAt(Instant.ofEpochMilli(System.currentTimeMillis() + 10 * 60 * 1000))
            .itemId(event1ItemId)
            .eventValue(event1Value)
            .properties("{\"numRatings\": " + event1NumRatings + "}")
            .build();

        eventList.add(event1);

        Event event2 = Event.builder()
            .eventType(event2Type)
            .sentAt(Instant.ofEpochMilli(System.currentTimeMillis() + 10 * 60 * 1000))
            .itemId(event2ItemId)
            .eventValue(event2Value)
            .properties("{\"numRatings\": " + event2NumRatings + "}")
            .build();

        eventList.add(event2);

        PutEventsRequest putEventsRequest = PutEventsRequest.builder()
            .trackingId(trackingId)
            .userId(userId)
            .sessionId(sessionId)
            .eventList(eventList)
            .build();

        int responseCode = personalizeEventsClient.putEvents(putEventsRequest)
            .sdkHttpResponse()
            .statusCode();

        System.out.println("Response code: " + responseCode);

    } catch (PersonalizeEventsException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}
}

```

Note

The properties keys use camel case names that match the fields in the Interactions schema. For example, if the field 'NUM_RATINGS' is defined in the Interactions schema, the property key should be numRatings.

Recording impressions data

If you use the [User-Personalization \(p. 98\)](#) recipe, you can record impressions data in your PutEvents operation. 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 uses impressions data to guide exploration, where recommendations include items with less interactions data or relevance. For information on the *implicit* and *explicit* impressions Amazon Personalize can model, see [Impressions data \(p. 46\)](#).

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 \(p. 330\)](#) request and Amazon Personalize derives the implicit impressions based on your recommendation data.

To manually record impressions data for an event, list the impressions in the [PutEvents \(p. 330\)](#) command's `impression` input parameter. The following code sample shows how to include a `recommendationId` and an `impression` in a `PutEvents` operation with either the SDK for Python (`Boto3`) or the SDK for Java 2.x. If you include both, Amazon Personalize uses the explicit impressions by default.

SDK for Python (Boto3)

```
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': 'item id',
            'recommendationId': 'recommendation id',
            'impression': ['itemId1', 'itemId2', 'itemId3']
        }
    ]
)
```

SDK for Java 2.x

Use the following `putEvents` method to record an event with impressions data and a `recommendationId`. For the `impressions` parameter, pass the list of `itemIds` as an `ArrayList`.

```
public static void putEvents(PersonalizeEventsClient personalizeEventsClient,
                             String trackingId,
                             String sessionId,
                             String userId,
                             String eventType,
                             Float eventValue,
                             String itemId,
                             ArrayList<String> impressions,
                             String recommendationId) {

    try {
        Event event = Event.builder()
            .eventType(eventType)
            .sentAt(Instant.ofEpochMilli(System.currentTimeMillis() + 10 * 60 * 1000))
            .itemId(itemId)
            .eventValue(eventValue)
            .impression(impressions)
            .recommendationId(recommendationId)
            .build();

        PutEventsRequest putEventsRequest = PutEventsRequest.builder()
            .trackingId(trackingId)
            .userId(userId)
```

```
.sessionId(sessionId)
.eventList(event)
.build();

int responseCode = personalizeEventsClient.putEvents(putEventsRequest)
.sdkHttpResponse()
.statusCode();
System.out.println("Response code: " + responseCode);

} catch (PersonalizeEventsException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}
}
```

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 \(p. 199\)](#).

Events and solutions

When training a model that uses event data, two parameters of the [CreateSolution \(p. 240\)](#) 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 \(p. 330\)](#) operation, see [2.View_Campaign_And_Interactions.ipynb](#) in the **getting_started** folder of the [amazon-personalize-samples](#) GitHub repository.

Exporting a dataset

After you've completed [Preparing and importing data \(p. 54\)](#), you can export the data in an Interactions, Items, or Users dataset to an Amazon S3 bucket. After exporting a dataset, you can verify and inspect the data that Amazon Personalize uses to generate recommendations, view the user interaction events that you previously recorded in real time, and perform offline analysis on your data.

You can choose to export only the data that you imported in bulk (imported using an Amazon Personalize dataset import job), only the data that you imported incrementally (historical and real-time records imported using the console or the `PutEvents`, `PutUsers`, or `PutItems` operations), or both.

To export a dataset, you create a dataset export job. A *dataset export job* is a record export tool that outputs the records in a dataset to one or more CSV files in an Amazon S3 bucket. The output CSV file includes a header row with column names that match the fields in the dataset's schema. Amazon Personalize combines duplicate records (records that match exactly for all fields) into one record.

You create a dataset export job using the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs.

Topics

- [Dataset export job permissions requirements \(p. 90\)](#)
- [Creating a dataset export job \(console\) \(p. 91\)](#)
- [Creating a dataset export job \(AWS CLI\) \(p. 92\)](#)
- [Creating a dataset export job \(AWS SDKs\) \(p. 93\)](#)

Dataset export job permissions requirements

To export a dataset, Amazon Personalize needs permission to add files to your Amazon S3 bucket. To grant permissions, attach a new AWS Identity and Access Management (IAM) policy to your Amazon Personalize service-linked role that grants the role permission to use the `PutObject` and `ListBucket` Actions on your bucket, and attach a bucket policy to your output Amazon S3 bucket that grants the Amazon Personalize principle permission to use the `PutObject` and `ListBucket` Actions.

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 service-linked role permission to use your key. You must also add Amazon Personalize as a Principle in your AWS KMS key policy. For more information, see [Using key policies in AWS KMS](#) in the [AWS Key Management Service Developer Guide](#).

Service-linked role policy for exporting a dataset

The following example policy grants your Amazon Personalize service-linked role permission to use the `PutObject` and `ListBucket` Actions. Replace `bucket-name` with the name of your output bucket. For information about attaching policies to a service-linked IAM role, see [Attaching an Amazon S3 policy to your Amazon Personalize service role \(p. 65\)](#).

```
{  
    "Version": "2012-10-17",  
    "Id": "PersonalizeS3BucketAccessPolicy",  
    "Statement": [
```

```
{  
    "Sid": "PersonalizeS3BucketAccessPolicy",  
    "Effect": "Allow",  
    "Action": [  
        "s3:PutObject",  
        "s3>ListBucket"  
    ],  
    "Resource": [  
        "arn:aws:s3:::bucket-name",  
        "arn:aws:s3:::bucket-name/*"  
    ]  
}  
]  
}
```

Amazon S3 bucket policy for exporting a dataset

The following example policy grants Amazon Personalize permission to use the PutObject and ListBucket Actions on an Amazon S3 bucket. Replace `bucket-name` with the name of your bucket. For information on adding an Amazon S3 bucket policy to a bucket, see [How Do I Add an S3 Bucket Policy?](#) in the *Amazon Simple Storage Service User Guide*.

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

Creating a dataset export job (console)

After you import your data into a dataset and create an output Amazon S3 bucket, you can export the data to the bucket for analysis. To export a dataset using the Amazon Personalize console, you create a dataset export job. For information about creating an Amazon S3 bucket, see [Creating a bucket](#) in the *Amazon Simple Storage Service User Guide*.

Before you export a dataset, make sure that your Amazon Personalize service-linked role can access and write to your output Amazon S3 bucket. See [Dataset export job permissions requirements \(p. 90\)](#).

To create a dataset export job (console)

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home>.
2. In the navigation pane, choose **Dataset groups**.

3. On the **Dataset groups** page, choose your dataset group.
4. In the navigation pane, choose **Datasets**.
5. Choose the dataset that you want to export to an Amazon S3 bucket.
6. In **Dataset export jobs**, choose **Create dataset export job**.
7. In **Dataset export job details**, for **Dataset export job name**, enter a name for the export job.
8. For **IAM service role**, choose the Amazon Personalize service-linked role that you created in [Creating an IAM service role for Amazon Personalize \(p. 10\)](#).
9. For **Amazon S3 data output path**, enter the destination Amazon S3 bucket. Use the following syntax:
s3://<name of your S3 bucket>/<folder path>
10. If you are using AWS KMS for encryption, for **KMS key ARN**, enter the Amazon Resource Name (ARN) for the AWS KMS key.
11. For **Export data type**, choose the type data to export based on how you originally imported the data.
 - Choose **Bulk** to export only data that you imported in bulk using a dataset import job.
 - Choose **Incremental** to export only data that you imported incrementally using the console or the `PutEvents`, `PutUsers`, or `PutItems` operations.
 - Choose **Both** to export all of the data in the dataset.
12. Choose **Create dataset export job**.

On the **Dataset overview** page, in **Dataset export jobs**, the job is listed with an **Export job status**. The dataset export job is complete when the status is **ACTIVE**. You can then download the data from the output Amazon S3 bucket. For information on downloading objects from an Amazon S3 bucket, see [Downloading an object in the Amazon Simple Storage Service User Guide](#).

Creating a dataset export job (AWS CLI)

After you import your data into the dataset and create an output Amazon S3 bucket, you can export the dataset to the bucket for analysis. To export a dataset using the AWS CLI, create a dataset export job using the `create-dataset-export-job` AWS CLI command. For information about creating an Amazon S3 bucket, see [Creating a bucket](#) in the *Amazon Simple Storage Service User Guide*.

Before you export a dataset, make sure that the Amazon Personalize service-linked role can access and write to your output Amazon S3 bucket. See [Dataset export job permissions requirements \(p. 90\)](#).

The following is an example of the `create-dataset-export-job` AWS CLI command. Give the job a name, replace `dataset_arn` with the Amazon Resource Name (ARN) of the dataset that you want to export, and replace `role_ARN` with the ARN of the Amazon Personalize service-linked role that you created in [Creating an IAM service role for Amazon Personalize \(p. 10\)](#). In `s3DataDestination`, for the `kmsKeyArn`, optionally provide the ARN for your AWS KMS key, and for the path provide the path to your output Amazon S3 bucket.

For `ingestion-mode`, specify the data to export from the following options:

- Specify **BULK** to export only data that you imported in bulk using a dataset import job.
- Specify **PUT** to export only data that you imported incrementally using the console or the `PutEvents`, `PutUsers`, or `PutItems` operations.
- Specify **ALL** to export all of the data in the dataset.

For more information see [CreateDatasetExportJob \(p. 224\)](#).

```
aws personalize create-dataset-export-job \
--job-name job name \
--dataset-arn dataset ARN \
--job-output "{\"s3DataDestination\":{\"kmsKeyArn\":\"kms key ARN\",\"path\": \
\"s3://bucket-name/folder-name\"}}" \
--role-arn role ARN \
--ingestion-mode PUT
```

The dataset export job ARN is displayed.

```
{  
  "datasetExportJobArn": "arn:aws:personalize:us-west-2:acct-id:dataset-export-job/  
DatasetExportJobName"  
}
```

Use the `DescribeDatasetExportJob` operation to check the status.

```
aws personalize describe-dataset-export-job \
--dataset-export-job-arn dataset export job ARN
```

Creating a dataset export job (AWS SDKs)

After you import your data into the dataset and create an output Amazon S3 bucket, you can export the dataset to the bucket for analysis. To export a dataset using the AWS SDKs, create a dataset export job using the [CreateDatasetExportJob](#) (p. 224) operation. For information about creating an Amazon S3 bucket, see [Creating a bucket](#) in the *Amazon Simple Storage Service User Guide*.

The following code shows how to create a dataset export job using the SDK for Python (Boto3) or the SDK for Java 2.x SDK.

Before you export a dataset, make sure that the Amazon Personalize service-linked role can access and write to your output Amazon S3 bucket. See [Dataset export job permissions requirements \(p. 90\)](#).

SDK for Python (Boto3)

Use the following `create_dataset_export_job` to export the data in a dataset to an Amazon S3 bucket. Give the job a name, replace `dataset_arn` with the Amazon Resource Name (ARN) of the dataset that you want to export, and replace `role_ARN` with the ARN of the Amazon Personalize service-linked role that you created in [Creating an IAM service role for Amazon Personalize \(p. 10\)](#). In `s3DataDestination`, for the `kmsKeyArn`, optionally provide the ARN for your AWS KMS key, and for the `path` provide the path to your output Amazon S3 bucket.

For `ingestionMode`, specify the data to export from the following options:

- Specify **BULK** to export only data that you imported in bulk using a dataset import job.
 - Specify **PUT** to export only data that you imported incrementally using the console or the `PutEvents`, `PutUsers`, or `PutItems` operations.
 - Specify **ALL** to export all of the data in the dataset.

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_dataset_export_job(
```

```

jobName = 'job name',
datasetArn = 'dataset ARN',
jobOutput = {
    "s3DataDestination": {
        "kmsKeyArn": "kms key ARN",
        "path": "s3://bucket-name/folder-name/"
    }
},
roleArn = 'role ARN',
ingestionMode = 'PUT'
)

dsej_arn = response['datasetExportJobArn']

print ('Dataset Export Job arn: ' + dsej_arn)

description = personalize.describe_dataset_export_job(
    datasetExportJobArn = dsej_arn)['datasetExportJob']

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

```

SDK for Java 2.x

Use the following `createDatasetExportJob` method to create a dataset export job. Pass the following as parameters: a `PersonalizeClient`, the name for your export job, the ARN of the dataset you want to export, the ingestion mode, the path for the output Amazon S3 bucket, and the ARN for your AWS KMS key.

The `ingestionMode` can be one of the following options:

- Use `IngestionMode.BULK` to export only data that you imported in bulk using a dataset import job.
- Use `IngestionMode.PUT` to export only data that you imported incrementally using the console or the `PutEvents`, `PutUsers`, or `PutItems` operations.
- Use `IngestionMode.ALL` to export all of the data in the dataset.

```

public static void createDatasetExportJob(PersonalizeClient personalizeClient,
                                         String jobName,
                                         String datasetArn,
                                         IngestionMode ingestionMode,
                                         String roleArn,
                                         String s3BucketPath,
                                         String kmsKeyArn) {

    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String status = null;

    try {
        S3DataConfig exportS3DataConfig = S3DataConfig.builder()
            .path(s3BucketPath)
            .kmsKeyArn(kmsKeyArn)
            .build();

        DatasetExportJobOutput jobOutput = DatasetExportJobOutput.builder()
            .s3DataDestination(exportS3DataConfig)
            .build();

        CreateDatasetExportJobRequest createRequest =
CreateDatasetExportJobRequest.builder()
            .jobName(jobName)
    }
}

```

```
.datasetArn(datasetArn)
.ingestionMode(ingestionMode)
.jobOutput(jobOutput)
.roleArn(roleArn)
.build();

String datasetExportJobArn =
personalizeClient.createDatasetExportJob(createRequest).datasetExportJobArn();

DescribeDatasetExportJobRequest describeDatasetExportJobRequest =
DescribeDatasetExportJobRequest.builder()
.datasetExportJobArn(datasetExportJobArn)
.build();

long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

while (Instant.now().getEpochSecond() < maxTime) {

    DatasetExportJob datasetExportJob =
personalizeClient.describeDatasetExportJob(describeDatasetExportJobRequest)
.datasetExportJob();

    status = datasetExportJob.status();
System.out.println("Export job status: " + status);

    if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
        break;
    }
    try {
        Thread.sleep(waitInMilliseconds);
    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
    }
}
} catch (PersonalizeException e) {
System.out.println(e.awsErrorDetails().errorMessage());
}
}
```

Creating a solution

After you have finished [Preparing and importing data \(p. 54\)](#), 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). After you create a solution with a solution version, you can [create a campaign \(p. 149\)](#) 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 \(p. 96\)](#).
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 \(p. 127\)](#).
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 \(p. 139\)](#).
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 a solution version with metrics \(p. 145\)](#).

Step 1: Choosing a recipe

Amazon Personalize provides recipes, based on common use cases, for training models. *Recipes* are Amazon Personalize algorithms that are prepared for specific use cases. 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 \(p. 134\)](#).

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.

Recipe type	Recipes	API	API requirements
USER_PERSONALIZATION	Personalization (p. 98)	GetRecommendations (p. 340)	userId: Required

Recipe type	Recipes	API	API requirements
	Popularity-Count (p. 106) HRNN recipe (legacy) (p. 107) HRNN-Metadata recipe (legacy) (p. 110) HRNN-Coldstart recipe (legacy) (p. 113)		itemId: Not used inputList: NA
PERSONALIZED_RANKING	Personalized-Ranking (p. 118)	GetPersonalizedRanking (p. 336)	userId: Required itemId: NA inputList: list of itemId's
RELATED_ITEMS	Similar-Items (p. 122) SIMS (p. 124)	GetRecommendations (p. 340)	userId: Not used itemId: Required inputList: NA

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 \(p. 315\)](#) API.
- With the AWS CLI, use the following command.

```
aws personalize list-recipes
```

To get information about a recipe using the SDK for Python (Boto3), call the [DescribeRecipe \(p. 282\)](#) API. To get information about a recipe using the AWS CLI, use the following command.

```
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 \(p. 98\)](#)
- [Popularity-Count recipe \(p. 106\)](#)
- [Legacy user personalization recipes \(p. 107\)](#)

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.

Topics

- [Automatic updates \(p. 98\)](#)
- [Working with impressions data \(p. 99\)](#)
- [Properties and hyperparameters \(p. 99\)](#)
- [Training with the User-Personalization recipe \(console\) \(p. 102\)](#)
- [Training with the User-Personalization recipe \(Python SDK\) \(p. 104\)](#)
- [Getting recommendations and recording impressions \(SDK for Python \(Boto3\)\) \(p. 105\)](#)
- [Sample Jupyter notebook \(p. 106\)](#)

Automatic updates

With User-Personalization, Amazon Personalize automatically updates the latest model (solution version) every two hours behind the scenes to include new data without creating a new solution version. 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. This is not a full retraining; you should still train a new solution version weekly with `trainingMode` set to `FULL` so the model can learn from your users' behavior.

If every two hours is not frequent enough, you can manually create a solution version with `trainingMode` set to `UPDATE` to include those new items in recommendations. Just remember that Amazon Personalize automatically updates only your latest fully trained solution version, so the manually updated solution version won't be automatically updated in the future.

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 versions you 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 a user when they interacted with (clicked, watched, purchased, and so on) a particular item.

Using this information, a solution created with the User-Personalization recipe can calculate the suitability of new items based on how frequently an item has been ignored, and change recommendations accordingly. For more information see [Impressions data \(p. 46\)](#).

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 \(p. 96\)](#).

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 \(p. 134\)](#).

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?

Name	Description
Algorithm hyperparameters	
hidden_dimension	<p>The number of hidden variables used in the model. <i>Hidden variables</i> 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 best value, use HPO. To use HPO, set <code>performHPO</code> to <code>true</code> when you call CreateSolution (p. 240) and CreateSolutionVersion (p. 245) operations.</p> <p>Default value: 149</p> <p>Range: [32, 256]</p> <p>Value type: Integer</p>

Name	Description
	HPO tunable: Yes
bptt	<p>Determines whether to use the back-propagation through time technique. <i>Back-propagation through time</i> 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.</p> <p>Default value: 32</p> <p>Range: [2, 32]</p> <p>Value type: Integer</p> <p>HPO tunable: Yes</p>
recency_mask	<p>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.</p> <p>Default value: <code>True</code></p> <p>Range: <code>True or False</code></p> <p>Value type: Boolean</p> <p>HPO tunable: Yes</p>
Featurization hyperparameters	

Name	Description
<code>min_user_history_length_percentile</code>	<p>The minimum percentile of user history lengths to include in model training. <i>History length</i> is the total amount of data about a user. Use <code>min_user_history_length_percentile</code> 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.</p> <p>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%.</p> <p>Default value: 0.0</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>
<code>max_user_history_length_percentile</code>	<p>The maximum percentile of user history lengths to include in model training. <i>History length</i> 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.</p> <p>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%.</p> <p>Default value: 0.99</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>
Item exploration campaign configuration hyperparameters	

Name	Description
<code>exploration_weight</code>	<p>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. 362).</p> <p>Default value: 0.3</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>
<code>exploration_item_age_cut_off</code>	<p>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. 362).</p> <p>Default value: 30.0</p> <p>Range: Positive floats</p> <p>Value type: Float</p> <p>HPO tunable: No</p>

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/home> 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](#) and [Unstructured text metadata \(p. 51\)](#) 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 importing data, see [Preparing and importing data \(p. 54\)](#).
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. In **Solution configuration**, if your Interactions dataset has EVENT_TYPE or both EVENT_TYPE and EVENT_VALUE columns, optionally use the **Event type** and **Event value threshold** fields to choose the interactions data that Amazon Personalize uses when training the model.

For more information see [Choosing the interactions data used for training \(p. 136\)](#).

8. Optionally configure hyperparameters for your solution. For a list of User-Personalization recipe properties and hyperparameters, see [Properties and hyperparameters \(p. 99\)](#).
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 \(p. 218\)](#) 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 \(p. 23\)](#) 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 \(p. 98\)](#).

To manually update the campaign, you first create and train a new solution version using the console or the [CreateSolutionVersion \(p. 245\)](#) operation, with `trainingMode` set to `update`.

You then manually update the campaign on the **Campaign** page of the console or by using the [UpdateCampaign \(p. 327\)](#) 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](#) and [Unstructured text metadata \(p. 51\)](#) 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 [Datasets and schemas \(p. 42\)](#).

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.

```
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 \(p. 99\)](#).

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.

```
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')

create_campaign_response = personalize.create_campaign(
    name = 'campaign name',
    solutionVersionArn = 'solution version arn',
    minProvisionedTPS = 1,
    campaignConfig = {"itemExplorationConfig": {"explorationWeight": "0.3",
"explorationItemAgeCutOff": "30"}}
)

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 \(p. 98\)](#).

To manually update the campaign, you first create and train a new solution version using the console or the [CreateSolutionVersion \(p. 245\)](#) operation, with `trainingMode` set to `update`. You then manually update the campaign on the **Campaign** page of the console or by using the [UpdateCampaign \(p. 327\)](#) operation.

Note

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

Getting recommendations and recording impressions (SDK for Python (Boto3))

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 SDKs see [Creating a batch inference job \(AWS SDKs\) \(p. 167\)](#).

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.

```
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.

```
import boto3
```

```

personalize = boto3.client('personalize')

event_tracker_response = personalize.create_event_tracker(
    name = 'event tracker name',
    datasetGroupArn = 'dataset group arn'
)
event_tracker_arn = event_tracker_response['eventTrackerArn']
event_tracking_id = event_tracker_response['trackingId']
print('eventTrackerArn:{}\n 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.

```

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 recommends the most popular item items based on all of your user behavioral data. The most popular items have the most interactions with unique users. The recipe returns the same popular items for all users. Popularity-Count is a good baseline for comparing with other recipes using the evaluation metrics Amazon Personalize generates when you create a solution version. For more information see [Step 4: Evaluating a solution version with metrics \(p. 145\)](#).

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-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 \(p. 98\)](#).

The following are legacy USER_PERSONALIZATION recipes.

- [HRNN recipe \(legacy\) \(p. 107\)](#)
- [HRNN-Coldstart recipe \(legacy\) \(p. 113\)](#)
- [HRNN-Metadata recipe \(legacy\) \(p. 110\)](#)

HRNN 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 \(p. 98\)](#).

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 \(p. 134\)](#).

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?

Name	Description
Algorithm hyperparameters	
hidden_dimension	<p>The number of hidden variables used in the model. <i>Hidden variables</i> 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. 240) and CreateSolutionVersion (p. 245) operations.</p> <p>Default value: 43</p> <p>Range: [32, 256]</p> <p>Value type: Integer</p> <p>HPO tunable: Yes</p>
bptt	<p>Determines whether to use the back-propagation through time technique. <i>Back-propagation through time</i> 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.</p> <p>Default value: 32</p> <p>Range: [2, 32]</p> <p>Value type: Integer</p> <p>HPO tunable: Yes</p>
recency_mask	<p>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.</p> <p>Default value: <code>True</code></p> <p>Range: <code>True</code> or <code>False</code></p> <p>Value type: Boolean</p>

Name	Description
	HPO tunable: Yes
Featurization hyperparameters	
<code>min_user_history_length_percentile</code>	<p>The minimum percentile of user history lengths to include in model training. <i>History length</i> is the total amount of data about a user. Use <code>min_user_history_length_percentile</code> 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.</p> <p>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%.</p> <p>Default value: 0.0</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>

Name	Description
<code>max_user_history_length_percentile</code>	<p>The maximum percentile of user history lengths to include in model training. <i>History length</i> 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.</p> <p>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%.</p> <p>Default value: 0.99</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>

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 \(p. 98\)](#).

The HRNN-Metadata recipe predicts the items that a user will interact with. It is similar to the [HRNN \(p. 107\)](#) 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 \(p. 134\)](#).

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)?

Name	Description
Algorithm Hyperparameters	
hidden_dimension	<p>The number of hidden variables used in the model. <i>Hidden variables</i> 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. 240) and CreateSolutionVersion (p. 245) operations.</p> <p>Default value: 43</p> <p>Range: [32, 256]</p> <p>Value type: Integer</p> <p>HPO tunable: Yes</p>
bptt	<p>Determines whether to use the back-propagation through time technique. <i>Back-propagation through time</i> 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.</p> <p>Default value: 32</p> <p>Range: [2, 32]</p> <p>Value type: Integer</p> <p>HPO tunable: Yes</p>
recency_mask	<p>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</p>

Name	Description
	<p>false. To get good recommendations using an equal weight, you might need a larger training dataset.</p> <p>Default value: True</p> <p>Range: True or False</p> <p>Value type: Boolean</p> <p>HPO tunable: Yes</p>
Featurization hyperparameters	
min_user_history_length_percentile	<p>The minimum percentile of user history lengths to include in model training. <i>History length</i> is the total amount of data about a user. Use <code>min_user_history_length_percentile</code> 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.</p> <p>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%.</p> <p>Default value: 0.0</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>

Name	Description
<code>max_user_history_length_percentile</code>	<p>The maximum percentile of user history lengths to include in model training. <i>History length</i> 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.</p> <p>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%.</p> <p>Default value: 0.99</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>

HRNN-Coldstart 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 \(p. 98\)](#).

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 \(p. 110\)](#) 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 \(p. 96\)](#).

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 \(p. 134\)](#).

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?

Name	Description
Algorithm hyperparameters	
hidden_dimension	<p>The number of hidden variables used in the model. <i>Hidden variables</i> 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. 240) and CreateSolutionVersion (p. 245) operations.</p> <p>Default value: 149</p> <p>Range: [32, 256]</p> <p>Value type: Integer</p> <p>HPO tunable: Yes</p>

Name	Description
bptt	<p>Determines whether to use the back-propagation through time technique. <i>Back-propagation through time</i> 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.</p> <p>Default value: 32</p> <p>Range: [2, 32]</p> <p>Value type: Integer</p> <p>HPO tunable: Yes</p>
recency_mask	<p>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.</p> <p>Default value: <code>True</code></p> <p>Range: <code>True</code> or <code>False</code></p> <p>Value type: Boolean</p> <p>HPO tunable: Yes</p>
Featurization hyperparameters	
cold_start_max_interactions	<p>The maximum number of user-item interactions an item can have to be considered a cold item.</p> <p>Default value: 15</p> <p>Range: Positive integers</p> <p>Value type: Integer</p> <p>HPO tunable: No</p>

Name	Description
<code>cold_start_max_duration</code>	<p>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 <code>cold_start_relative_from</code> hyperparameter.</p> <p>Default value: 5.0</p> <p>Range: Positive floats</p> <p>Value type: Float</p> <p>HPO tunable: No</p>
<code>cold_start_relative_from</code>	<p>Determines the starting point for the HRNN-Coldstart recipe to calculate <code>cold_start_max_duration</code>. To calculate from the current time, choose <code>currentTime</code>.</p> <p>To calculate <code>cold_start_max_duration</code> from the timestamp of the latest item in the Interactions dataset, choose <code>latestItem</code>. This setting is useful if you frequently add new items.</p> <p>Default value: <code>latestItem</code></p> <p>Range: <code>currentTime</code>, <code>latestItem</code></p> <p>Value type: String</p> <p>HPO tunable: No</p>

Name	Description
<code>min_user_history_length_percentile</code>	<p>The minimum percentile of user history lengths to include in model training. <i>History length</i> is the total amount of data about a user. Use <code>min_user_history_length_percentile</code> 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.</p> <p>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%.</p> <p>Default value: 0.0</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>
<code>max_user_history_length_percentile</code>	<p>The maximum percentile of user history lengths to include in model training. <i>History length</i> 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.</p> <p>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%.</p> <p>Default value: 0.99</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>

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 \(p. 240\)](#) 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.

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

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. 118)

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 data in your Interactions dataset, and if you created them, the Items dataset and Users dataset in your dataset group.

Personalized-Ranking recipe

The Personalized-Ranking recipe generates personalized rankings of items. A *personalized ranking* is a list of recommended items that are re-ranked for a specific user. This is useful if you have a collection of ordered items, such as search results, promotions, or curated lists, and you want to provide a personalized re-ranking for each of your users.

To train a model, the Personalized-Ranking recipe uses the data in your Interactions dataset, and if you created them, the Items dataset and Users dataset in your dataset group. With Personalized-Ranking,

your Items dataset can include [Unstructured text metadata \(p. 51\)](#) and your Interactions dataset can include [Contextual metadata \(p. 45\)](#). To get a personalized ranking, use the [GetPersonalizedRanking \(p. 336\)](#) API.

Note

If you provide items without interactions data for ranking, Amazon Personalize will return these items without a recommendation score in the GetPersonalizedRanking API response.

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 \(p. 134\)](#).

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)?

Name	Description
Algorithm hyperparameters	
hidden_dimension	<p>The number of hidden variables used in the model. <i>Hidden variables</i> 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. 240) and CreateSolutionVersion (p. 245) operations.</p> <p>Default value: 149</p> <p>Range: [32, 256]</p> <p>Value type: Integer</p> <p>HPO tunable: Yes</p>

Name	Description
bptt	<p>Determines whether to use the back-propagation through time technique. <i>Back-propagation through time</i> 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.</p> <p>Default value: 32</p> <p>Range: [2, 32]</p> <p>Value type: Integer</p> <p>HPO tunable: Yes</p>
recency_mask	<p>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.</p> <p>Default value: <code>True</code></p> <p>Range: <code>True or False</code></p> <p>Value type: Boolean</p> <p>HPO tunable: Yes</p>
Featurization hyperparameters	

Name	Description
<code>min_user_history_length_percentile</code>	<p>The minimum percentile of user history lengths to include in model training. <i>History length</i> is the total amount of data about a user. Use <code>min_user_history_length_percentile</code> 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.</p> <p>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%.</p> <p>Default value: 0.0</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>
<code>max_user_history_length_percentile</code>	<p>The maximum percentile of user history lengths to include in model training. <i>History length</i> 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.</p> <p>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%.</p> <p>Default value: 0.99</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>

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 recipes return items similar to an item that you specify when you get recommendations. The RELATED_ITEMS recipes are as follows:

- [Similar-Items recipe \(p. 122\)](#)
- [SIMS recipe \(p. 124\)](#)

Similar-Items (p. 122)

The Similar-Items recipe generates recommendations for items that are similar to an item you specify. Similarity is calculated based on both interactions data and item metadata. Use Similar-Items when you have a catalog that includes many new items with item metadata but little to no interactions data.

To use Similar-Items, you must create an Interactions dataset and an Items dataset. If you don't have item metadata and want to recommend similar items, use the [SIMS recipe \(p. 124\)](#) recipe. If Amazon Personalize can't find the item ID that you specify in your recommendation request, the recipe returns popular items as recommendations.

SIMS (p. 124)

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 your Interactions 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.

Similar-Items recipe

The Similar-Items (aws-similar-items) generates recommendations for items that are similar to an item you specify. Similar-Items is optimized for similar item recommendation scenarios with item metadata. To use Similar-Items, you must create an Interactions dataset and an Items dataset. Use Similar-Items when your catalog has item metadata and items with little to no interactions, but your Interactions dataset still has at minimum 1000 unique historical and event interactions (combined).

Similar-Items calculates similarity based on both the co-occurrence of the item in user histories in your Interaction dataset, and the item metadata, including categorical and unstructured text metadata, in your Items dataset. For example, with Similar-Items Amazon Personalize could recommend items customers frequently bought together with a similar style, or movies that different users also watched with a similar description.

With Similar-Items, you provide an item ID in a [GetRecommendations \(p. 340\)](#) operation (or the Amazon Personalize console) and Amazon Personalize returns a list of similar items. Or you can use a batch workflow to get similar items for all of the items in your inventory (see [Getting batch recommendations \(p. 163\)](#)). You can get recommendations for items that are similar to a cold item (an item with fewer than five interactions). If Amazon Personalize can't find the item ID that you specify in your recommendation request or batch input file, the recipe returns popular items as recommendations.

For information on formatting categorical and unstructured text metadata in your Items dataset, see [Items dataset \(p. 50\)](#). If you don't have item metadata and want to recommend similar items, use the [SIMS recipe \(p. 124\)](#).

Properties and hyperparameters

The Similar-Items recipe has the following properties:

- **Name** – aws-similar-items
- **Recipe Amazon Resource Name (ARN)** – arn:aws:personalize:::recipe/aws-similar-items
- **Algorithm ARN** – arn:aws:personalize:::algorithm/aws-similar-items

For more information, see [Step 1: Choosing a recipe \(p. 96\)](#).

The following table describes the hyperparameters for the Similar-Items recipe. A *hyperparameter* is an algorithm parameter that you can adjust to improve model performance. Algorithm hyperparameters control how the model performs. The process of choosing the best value for a hyperparameter is called hyperparameter optimization (HPO). For more information, see [Hyperparameters and HPO \(p. 134\)](#).

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?

Name	Description
Algorithm hyperparameters (HPO only)	
item_id_hidden_dimension	<p>The number of hidden variables Amazon Personalize uses to model item ID embeddings based on interactions data. <i>Hidden variables</i> recreate users' purchase history and item statistics to generate ranking scores. To use <code>item_id_hidden_dimension</code>, you must use HPO and provide minimum and maximum range values. Amazon Personalize uses HPO to find the best value within the range you specify. Specify a greater maximum value when you have a large Interactions dataset. Using a greater maximum value requires more time to process.</p> <p>To use HPO, set <code>performHPO</code> to <code>true</code> when you call the CreateSolution (p. 240) operation.</p> <p>Default value: 100</p> <p>Range: [30, 200]</p> <p>Value type: Integer</p> <p>HPO tunable: HPO only</p>
item_metadata_hidden_dimension	<p>The number of hidden variables Amazon Personalize uses to model item metadata. To use <code>item_id_hidden_dimension</code>, you must use HPO and provide minimum and maximum range values. Amazon Personalize uses HPO to find the best value within the range you specify. Specify a greater maximum value when you have a large Interactions dataset. Using a greater maximum requires more time to process.</p>

Name	Description
	<p>To use HPO, set <code>performHPO</code> to <code>true</code> when you call the CreateSolution (p. 240) operation.</p> <p>Default value: 100</p> <p>Range: [30, 200]</p> <p>Value type: Integer</p> <p>HPO tunable: HPO only</p>

SIMS recipe

The Item-to-item similarities (SIMS) recipe uses collaborative filtering to recommend items that are most similar to an item you specify when you get recommendations. SIMS uses your Interactions dataset, not item metadata such as color or price, to determine similarity. SIMS identifies the co-occurrence of the item in user histories in your Interaction dataset to recommend similar items. For example, with SIMS Amazon Personalize could recommend coffee shop items customers frequently bought together or movies that different users also watched.

Training is faster with the SIMS recipe compared to other recipes. If there isn't sufficient user behavior data for an item or the item ID you provide isn't found, SIMS recommends popular items.

The SIMS 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 \(p. 134\)](#).

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)?

Name	Description
Algorithm hyperparameters	
<code>popularity_discount_factor</code>	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 correlation. A value of 1 makes most items that have co-interactions (shared interaction) with the specific

Name	Description
	<p>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.</p> <p>Default value: 0.5</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: Yes</p>
min_cointeraction_count	<p>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.</p> <p>Default value: 3</p> <p>Range: [0, 10]</p> <p>Value type: Integer</p> <p>HPO tunable: Yes</p>
Featurization hyperparameters	
min_user_history_length_percentile	<p>The minimum percentile of user history lengths to include in model training. <i>History length</i> is the total amount of available data on a user. Use <code>min_user_history_length_percentile</code> 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.</p> <p>Default value: 0.005</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>

Name	Description
<code>max_user_history_length_percentile</code>	<p>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 <code>max_user_history_length_percentile</code> 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.</p> <p>For example, <code>min_hist_length_percentile = 0.05</code> and <code>max_hist_length_percentile = 0.95</code> includes all users except ones with history lengths at the bottom or top 5%.</p> <p>Default value: 0.995</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>
<code>min_item_interaction_count_percentile</code>	<p>The minimum percentile of item interaction counts to include in model training. Use <code>min_item_interaction_count_percentile</code> 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.</p> <p>Default value: 0.01</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>

Name	Description
max_item_interaction_count_percentile	<p>The maximum percentile of item interaction counts to include in model training. Use <code>max_item_interaction_count_percentile</code> to exclude a percentage of items with a long history of interactions. Items with a long history tend to be older and might be out of date. For example, a movie release that is out of print. Removing these items can focus on more relevant items. 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.</p> <p>For example, <code>min_item_interaction_count_percentile = 0.05</code> and <code>max_item_interaction_count_percentile = 0.95</code> includes all items except ones with an interaction count at the bottom or top 5%.</p> <p>Default value: 0.9</p> <p>Range: [0.0, 1.0]</p> <p>Value type: Float</p> <p>HPO tunable: No</p>

SIMS sample notebook

For a sample Jupyter notebook that shows you how to use the SIMS recipe, see [Finding similar items + HPO](#).

Step 2: Configuring a solution

Once you complete [Step 1: Choosing a recipe \(p. 96\)](#), you are ready to configure a solution for training a model.

Configuring a solution allows you 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 can 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 \(console\) \(p. 128\)](#)
- [Configuring a solution \(AWS CLI\) \(p. 128\)](#)
- [Configuring a solution \(AWS SDKs\) \(p. 129\)](#)
- [Optimizing a solution for an additional objective \(p. 130\)](#)
- [Hyperparameters and HPO \(p. 134\)](#)

- Choosing the interactions data used for training (p. 136)

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.

Configure a solution (console)

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> 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 \(p. 96\)](#)).
6. In **Solution configuration**, if your Interactions dataset has EVENT_TYPE or both EVENT_TYPE and EVENT_VALUE columns, optionally use the **Event type** and **Event value threshold** fields to choose the interactions data that Amazon Personalize uses when training the model.
For more information see [Choosing the interactions data used for training \(p. 136\)](#).
7. If you use either the [User-Personalization recipe \(p. 98\)](#) or [Personalized-Ranking recipe \(p. 118\)](#) recipe, optionally specify an **Objective** and choose an **Objective sensitivity** to optimize your solution for an objective in addition to relevance. For more information see [Optimizing a solution for an additional objective \(p. 130\)](#).
8. 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 \(p. 96\)](#).
9. Choose **Next**. The Create Solution Version page displays. Proceed to [Creating a solution version \(console\) \(p. 139\)](#).

Configuring a solution (AWS CLI)

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

```
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:

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

You can modify the above code to optimize recipe properties and hyperparameters (see [Hyperparameters and HPO \(p. 134\)](#)) or filter the Interactions data used for training (see [Choosing the interactions data used for training \(p. 136\)](#)).

If you use either the [User-Personalization recipe \(p. 98\)](#) or [Personalized-Ranking recipe \(p. 118\)](#) recipe, you can optimize your solution for an objective in addition to relevance. For more information see [Optimizing a solution for an additional objective \(p. 130\)](#).

Record the solution ARN for future use and proceed to [Creating a solution version \(AWS CLI\) \(p. 140\)](#).

Configuring a solution (AWS SDKs)

The following code shows how to create an Amazon Personalize solution using the SDK for Python (Boto3) or SDK for Java 2.x.

You can modify the following code to optimize recipe properties and hyperparameters (see [Hyperparameters and HPO \(p. 134\)](#)) or filter the Interactions data used for training (see [Choosing the interactions data used for training \(p. 136\)](#)). If you use either the [User-Personalization recipe \(p. 98\)](#) or [Personalized-Ranking recipe \(p. 118\)](#) recipe, you can optimize your solution for an objective in addition to relevance. For more information see [Optimizing a solution for an additional objective \(p. 130\)](#).

Record the solution ARN for future use and proceed to [Creating a solution version \(AWS SDKs\) \(p. 140\)](#).

SDK for Python (Boto3)

Create a new solution using the following `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 \(p. 96\)](#), and `dataset_group_arn` with the ARN of your dataset group.

```
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)
```

SDK for Java 2.x

Create a new solution using the following `createPersonalizeSolution` method. Pass the following as parameters: A `PersonalizeClient`, the Amazon Resource Name (ARN) of your dataset group, a name for the solution, and the ARN for the recipe from [Step 1: Choosing a recipe \(p. 96\)](#).

```
public static String createPersonalizeSolution(PersonalizeClient personalizeClient,
                                                String datasetGroupArn,
                                                String solutionName,
                                                String recipeArn) {

    try {
        CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
            .name(solutionName)
            .datasetGroupArn(datasetGroupArn)
            .recipeArn(recipeArn)
            .build();

        CreateSolutionResponse solutionResponse =
            personalizeClient.createSolution(solutionRequest);
```

```
        return solutionResponse.solutionArn();

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

Optimizing a solution for an additional objective

The primary objective of Amazon Personalize is to predict the most relevant items for your users based on historical and real-time interactions data. These are the items your users will most likely interact with (for example, the items they will most likely click). If you have an additional objective, such as maximizing streaming minutes or increasing revenue, you can create a solution that generates recommendations based on both relevance and your objective.

After you create an Interactions dataset and an Items dataset with a non-null, numerical column metadata attribute, you can create a solution that is optimized for an additional objective based on your item metadata. You can use the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs.

To optimize a solution for an additional objective, create a new solution with the User-Personalization recipe or Personalized-Ranking recipe and choose the numerical metadata column in your Items dataset that is related to your objective. When generating recommendations, Amazon Personalize gives more importance to items with higher values for this column of data. For example, you might choose a VIDEO_LENGTH column to maximize streaming minutes or a PRICE column to maximize revenue.

Objective requirements are as follows:

- You can choose only one column for your objective.
- The column must have a numerical type in your schema.
- The column can't have a null type in your schema.

For more information about schemas and data types, see [Datasets and schemas \(p. 42\)](#).

Topics

- [Balancing objective emphasis and relevance \(p. 130\)](#)
- [Measuring optimization performance \(p. 131\)](#)
- [Optimizing a solution \(console\) \(p. 131\)](#)
- [Optimizing a solution \(AWS CLI\) \(p. 132\)](#)
- [Optimizing a solution \(AWS SDKs\) \(p. 132\)](#)
- [Sample Jupyter notebook \(p. 134\)](#)

Balancing objective emphasis and relevance

There can be a trade-off when recommending items based more on your objective than relevance. For example, if you want to increase revenue through recommendations, recommendations for only expensive items might make items less relevant for your users and decrease user engagement and conversion.

To configure the balance between relevance and your objective, choose one of the following objective sensitivity levels when you create the solution:

- Off: Amazon Personalize uses primarily interactions data to predict the most relevant items for your user.
- Low: Amazon Personalize places less emphasis on your objective. Relevance through interactions data is more important.
- Medium: Amazon Personalize places equal emphasis on your objective and relevance through interactions data.
- High: Amazon Personalize places more emphasis on your objective. Relevance through interactions data is less important.

Measuring optimization performance

When you create a solution version (train a model) for a solution with an optimization objective, Amazon Personalize generates an `average_rewards_at_k` metric. The score for `average_rewards_at_k` tells you how well the solution version performs in achieving your objective. To calculate this metric, Amazon Personalize calculates the rewards for each user as follows:

```
rewards_per_user = total rewards from the user's interactions with their top 25 reward generating recommendations / total rewards from the user's interactions with recommendations
```

The final `average_rewards_at_k` is the average of all `rewards_per_user` normalized to be a decimal value less than or equal to 1 and greater than 0. The closer the value is to 1, the more gains on average per user you can expect from recommendations.

For example, if your objective is to maximize revenue from clicks, Amazon Personalize calculates each user score by dividing total revenue generated by the items the user clicked from their top 25 most expensive recommendations by the revenue from all of the recommended items the user clicked. Amazon Personalize then returns a normalized average of all user scores. The closer the `average_rewards_at_k` is to 1, the more revenue on average you can expect to gain per user from recommendations.

For more information about generating metrics, see [Step 4: Evaluating a solution version with metrics \(p. 145\)](#).

Optimizing a solution (console)

To optimize a solution for an additional objective with the Amazon Personalize console, create a new solution and choose the column of numerical item metadata that is related to your objective.

To optimize a solution for an additional objective (console)

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> and sign in to your account.
2. Choose the dataset group that you want to use for training.
3. In the **Create solutions** section of the dashboard, choose the **Start** button. If you have already created a solution, choose the **Create solution** button.
4. For **Solution name**, specify a name for the solution.
5. For **Recipe**, choose either [User-Personalization recipe \(p. 98\)](#) or [Personalized-Ranking recipe \(p. 118\)](#).
6. In **Solution configuration**, if your Interactions dataset has `EVENT_TYPE` or both `EVENT_TYPE` and `EVENT_VALUE` columns, optionally use the **Event type** and **Event value threshold** fields to choose the interactions data that Amazon Personalize uses when training the model.

For more information see [Choosing the interactions data used for training \(p. 136\)](#).

7. For **Objective**, choose the numerical column from the Items dataset that is related to your objective. You can choose only a numerical metadata column.
8. For **Objective sensitivity**, choose the level of emphasis Amazon Personalize places on the additional objective when generating recommendations. The objective sensitivity configures how Amazon Personalize balances recommending items based on your objective versus relevance through interactions data. For more information, see [Balancing objective emphasis and relevance \(p. 130\)](#).
9. Configure any hyperparameter options based on your recipe and business needs. Different recipes use different hyperparameters. For available hyperparameters, see the documentation for individual recipes in [Step 1: Choosing a recipe \(p. 96\)](#).
10. Choose **Next**.
11. On the **Create solution version** page, create a new solution version for the solution. For details, see [Creating a solution version \(console\) \(p. 139\)](#). Once you create a solution version, you can view the optimization performance in the solution version metrics. See [Measuring optimization performance \(p. 131\)](#).

Optimizing a solution (AWS CLI)

You can optimize for an objective only with the User-Personalization or Personalized-Ranking recipe. To optimize a solution for an additional objective using the AWS CLI, create a new solution and specify your objective details using the `optimizationObjective` key in the `solutionConfig` object. The `optimizationObjective` has the following fields:

- `itemAttribute`: Specify the name of the numerical metadata column from the Items dataset that relates to your objective.
- `objectiveSensitivity`: Specify the level of emphasis that the solution places on your objective when generating recommendations. The objective sensitivity level configures how Amazon Personalize balances recommending items based on your objective versus relevance through interactions data. The `objectiveSensitivity` can be OFF, LOW, MEDIUM or HIGH. For more information, see [Balancing objective emphasis and relevance \(p. 130\)](#).

The following is an example of the `create-solution` AWS CLI command. Replace the `solution name`, `dataset group arn`, and `recipe arn` values with your own.

For `optimizationObjective`, replace `COLUMN_NAME` with the numerical metadata column name from the Items dataset that is related to your objective. For `objectiveSensitivity`, specify OFF, LOW, MEDIUM, or HIGH.

```
aws personalize create-solution \
--name solution name \
--dataset-group-arn dataset group arn \
--recipe-arn recipe arn \
--solution-config "{\"optimizationObjective\":{\"itemAttribute\":\"COLUMN_NAME\",\"objectiveSensitivity\":\"MEDIUM\"]}"
```

When your solution is ready, create a new solution version (for an example command see [Configuring a solution \(AWS CLI\) \(p. 128\)](#)). Once you create a solution version, you can view the optimization performance with the solution version metrics. See [Measuring optimization performance \(p. 131\)](#).

Optimizing a solution (AWS SDKs)

You can optimize for an objective only with the User-Personalization or Personalized-Ranking recipe.

To optimize a solution for an additional objective using the AWS SDKs, create a new solution and specify your objective details using the `optimizationObjective` key in the `solutionConfig` object for the solution. The `optimizationObjective` has the following fields:

- **itemAttribute:** Specify the name of the numerical metadata column from the dataset group's Items dataset that relates to your objective.
- **objectiveSensitivity:** Specify the level of emphasis that the solution places on your objective when generating recommendations. The objective sensitivity level configures how Amazon Personalize balances recommending items based on your objective versus relevance through interactions data. The **objectiveSensitivity** can be OFF, LOW, MEDIUM or HIGH. For more information, see [Balancing objective emphasis and relevance \(p. 130\)](#).

Use the following code to create a solution with an additional objective with the AWS SDK for Python (Boto3) or the AWS SDK for Java 2.x.

When your solution is ready, create a new solution version (for example code see [Creating a solution version \(AWS SDKs\) \(p. 140\)](#)). Once you create a solution version, you can view the optimization performance with the solution version metrics. See [Measuring optimization performance \(p. 131\)](#).

SDK for Python (Boto3)

To create a solution that is optimized for an additional objective, use the following `create_solution` method. Replace the `solution_name`, `dataset_group_arn`, and `recipe_arn` values with your own.

For `optimizationObjective`, replace `COLUMN_NAME` with the numerical metadata column name from the Items dataset that is related to your objective. For `objectiveSensitivity`, specify OFF, LOW, MEDIUM, or HIGH.

```
import boto3

personalize = boto3.client('personalize')

create_solution_response = personalize.create_solution(
    name= 'solution name',
    recipeArn= 'recipe arn',
    datasetGroupArn = 'dataset group arn',
    solutionConfig = {
        "optimizationObjective": {
            "itemAttribute": "COLUMN_NAME",
            "objectiveSensitivity": "MEDIUM"
        }
    }
)
solution_arn = create_solution_response['solutionArn']
print('solution_arn: ', solution_arn)
```

SDK for Java 2.x

To create a solution that is optimized for an additional objective, use the following `createPersonalizeSolution` method and pass the following as parameters: an Amazon Personalize service client, the dataset group's Amazon Resource Name (ARN), a solution name, the recipe ARN, the item attribute, and the objective sensitivity level.

```
public static String createPersonalizeSolution(PersonalizeClient personalizeClient,
                                                String datasetGroupArn,
                                                String solutionName,
                                                String recipeArn,
                                                String itemAttribute,
                                                String objectiveSensitivity) {

    try {
        OptimizationObjective optimizationObjective = OptimizationObjective.builder()
            .itemAttribute(itemAttribute)
```

```

        .objectiveSensitivity(objectiveSensitivity)
        .build();

    SolutionConfig solutionConfig = SolutionConfig.builder()
        .optimizationObjective(optimizationObjective)
        .build();

    CreateSolutionRequest solutionRequest = CreateSolutionRequest.builder()
        .name(solutionName)
        .datasetGroupArn(datasetGroupArn)
        .recipeArn(recipeArn)
        .solutionConfig(solutionConfig)
        .build();

    CreateSolutionResponse solutionResponse =
personalizeClient.createSolution(solutionRequest);

    return solutionResponse.solutionArn();

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";

```

Sample Jupyter notebook

For a sample Jupyter notebook that shows how to create a solution that is optimized for an additional objective based item metadata, see the [objective_optimization](#) folder of the [Amazon Personalize samples](#) GitHub repository

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 \(p. 419\)](#) object that is passed to the [CreateSolution \(p. 240\)](#) 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`.

```
{
  "name": "string",
  "performAutoML": boolean,
  "recipeArn": "string",
  "performHPO": boolean,
  "eventType": "string",
  "solutionConfig": {
    "optimizationObjective": {
      "itemAttribute": "string",
      "objectiveSensitivity": "string"
    }
    "autoMLConfig": {
      "metricName": "string",
      "recipeList": [ "string" ]
    },
    "eventValueThreshold": "string",
  }
}
```

```

    "featureTransformationParameters": {
        "string" : "string"
    },
    "algorithmHyperParameters": {
        "string" : "string"
    },
    "hpoConfig": {
        "algorithmHyperParameterRanges": {
            ...
        },
        "hpoResourceConfig": {
            "maxNumberOfTrainingJobs": "string",
            "maxParallelTrainingJobs": "string"
        }
    },
},
}

```

Different recipes use different hyperparameters. For the available hyperparameters, see the individual recipes in [Step 1: Choosing a recipe \(p. 96\)](#).

Enabling hyperparameter optimization

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). For more information about HPO, see [Automatic model tuning](#).

The following is a partial example of a `CreateSolution` request using the [HRNN \(p. 107\)](#) recipe with HPO set to `true`.

```

{
    "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" ]
                    }
                ],
                "integerHyperParameterRanges": [
                    {
                        "name": "bptt",
                        "minValue": 20,
                        "maxValue": 40
                    }
                ]
            },
            "hpoResourceConfig": {
                "maxNumberOfTrainingJobs": "4",

```

```
        "maxParallelTrainingJobs": "2"
    }
}
}
```

Viewing hyperparameters

Once training is complete, you can view the hyperparameters of the performing model by calling the [the section called “DescribeSolutionVersion” \(p. 289\)](#) operation. The following sample shows a condensed DescribeSolutionVersion output with the optimized hyperparameters displayed in the tunedHPOParams object.

```
{
  "solutionVersion": {
    "creationDateTime": 1562191944.745,
    "datasetGroupArn": "arn:aws:personalize:us-west-2:000000000000:dataset-group/hpo",
    "lastUpdatedDateTime": 1562194465.075,
    "performAutoML": false,
    "performHPO": true,
    "recipeArn": "arn:aws:personalize:::recipe/aws-hrnn",
    "solutionArn": "arn:aws:personalize:us-west-2:000000000000:solution/hpo",
    "solutionVersionArn": "arn:aws:personalize:us-west-2:000000000000:solution/hpo/5a515609",
    "status": "ACTIVE",
    "tunedHPOParams": {
      "algorithmHyperParameters": {
        "hidden_dimension": "58",
        "recency_mask": "false"
      }
    }
  }
}
```

Choosing the interactions data used for training

You can choose the events in an Interactions dataset that Amazon Personalize uses when creating a solution version (training a model). Choosing interactions data before training allows you to use only a relevant subset of your data for training or remove noise to train a more optimized model. For more information about Interactions datasets, see [Datasets and schemas \(p. 42\)](#) and [Interactions dataset \(p. 44\)](#).

You can choose interactions data as follows:

- **Choose records based on type** – When you configure a solution, if your Interactions dataset includes event types in an EVENT_TYPE column, you can optionally specify an event type to use in training. For example, if your Interactions dataset includes *purchase*, *click*, and *watch* event types, and you want Amazon Personalize to train the model with only *watch* events, when you configure your solution, you would provide *watch* as the event type that Amazon Personalize uses in training.

If your Interactions dataset has multiple event types in an EVENT_TYPE column, and you do not provide an event type when you configure your solution, Amazon Personalize uses all interactions data for training with equal weight regardless of type.

- **Choose records based on type and value** – When you configure a solution, if your Interactions dataset includes EVENT_TYPE and EVENT_VALUE fields, you can set a specific value as a threshold to exclude records from training. For example, if your EVENT_VALUE data for events with an EVENT_TYPE of *watch* is the percentage of a video that a user watched, if you set the event value threshold to 0.5, and the event type to *watch*, Amazon Personalize trains the model using only *watch* interaction events with an EVENT_VALUE greater than or equal to 0.5.

Filtering records by event value and event type (AWS SDK)

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 \(p. 32\)](#).

Prerequisites: Complete the prerequisites and verify that your Python environment is set up as described in [Getting started \(SDK for Python \(Boto3\)\) \(p. 31\)](#).

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.

```
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"
        }
    ],
    "version": "1.0"
}

# 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 \(p. 62\)](#).
3. Upload your data to an Amazon Simple Storage Service (Amazon S3) bucket. For a code sample, see [Uploading to an Amazon S3 bucket \(p. 64\)](#).
4. Import your data into Amazon Personalize with the [CreateDatasetImportJob \(p. 230\)](#) 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 \(AWS SDKs\) \(p. 71\)](#).
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.

```
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 \(p. 240\)](#) 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.

```
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']
print('Solution status: ' + solution_description['status'])
```

7. When you have the solution, use it to train a model by specifying its solution ARN in a [CreateSolutionVersion \(p. 245\)](#) request.

```
import boto3

personalize = boto3.client('personalize')

# Create a solution version
create_solution_version_response = personalize.create_solution_version(solutionArn
= solution_arn)
```

```
# Store the solution version ARN
solution_version_arn = create_solution_version_response['solutionVersionArn']

# 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'])
```

Training is complete when the status is **ACTIVE**. For more information, see [Creating a solution \(p. 96\)](#).

After you train a model, you should evaluate its performance. To optimize your model, you might want to adjust the `eventValueThreshold` or other hyperparameters. For more information, see [Step 4: Evaluating a solution version with metrics \(p. 145\)](#).

Step 3: Creating a solution version

After you have completed [Step 1: Choosing a recipe \(p. 96\)](#) and [Step 2: Configuring a solution \(p. 127\)](#), 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 create a solution version using the console, AWS Command Line Interface (AWS CLI), or AWS SDK. If your solution version has a status of **CREATE_PENDING** or **CREATE_IN_PROGRESS**, you can use the [the section called "StopSolutionVersionCreation" \(p. 325\)](#) operation to stop the solution version creation process. See [Stopping the creation of a solution version \(p. 143\)](#).

Topics

- [Creating a solution version \(console\) \(p. 139\)](#)
- [Creating a solution version \(AWS CLI\) \(p. 140\)](#)
- [Creating a solution version \(AWS SDKs\) \(p. 140\)](#)
- [Stopping the creation of a solution version \(p. 143\)](#)

Creating a solution version (console)

If you just completed [Step 2: Configuring a solution \(p. 127\)](#) 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 \(console\) \(p. 15\)](#) 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.

To create a new solution version

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> 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. On the **Solution and recipes** page, choose the solution you want to create a solution version for.
5. On the solution overview page, 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** you can evaluate it using metrics supplied by Amazon Personalize. For more information, see [Step 4: Evaluating a solution version with metrics \(p. 145\)](#).

If training does not complete because of an error, you are not charged for the training. If your solution version has a status of CREATE_PENDING or CREATE_IN_PROGRESS, you can stop the solution version creation process. To stop solution version creation, navigate to the solution version details page and choose **Stop**. See [Stopping the creation of a solution version \(p. 143\)](#).

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 \(p. 127\)](#).

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

The solution version ARN is displayed, for example:

```
{  
    "solutionVersionArn": "arn:aws:personalize:us-west-2:acct-id:solution/SolutionName/  
<version-id>"  
}
```

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 \(p. 289\)](#).

```
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": {  
        "solutionVersionArn": "arn:aws:personalize:us-west-2:acct-id:solution/solutionName/  
<version-id>",  
        ...  
        "status": "CREATE PENDING"  
    }  
}
```

Training is complete when the status is ACTIVE and you can evaluate it using metrics supplied by Amazon Personalize. For more information, see [Step 4: Evaluating a solution version with metrics \(p. 145\)](#). If training does not complete because of an error, you are not charged for the training.

If your solution version has a status of CREATE_PENDING or CREATE_IN_PROGRESS, you can use the `StopSolutionVersionCreation (p. 325)` operation to stop the solution version creation process. See [Stopping the creation of a solution version \(p. 143\)](#).

Creating a solution version (AWS SDKs)

When your solution is ACTIVE, use the following code to create a solution version with the AWS SDK for Python (Boto3) or the AWS SDK for Java 2.x.

SDK for Python (Boto3)

To create a solution version, using the following `create_solution_version` method. Replace the `solution_arn` with the Amazon Resource Name (ARN) of the solution from [Step 2: Configuring a solution \(p. 127\)](#). The following code uses the [DescribeSolutionVersion \(p. 289\)](#) operation to retrieve the solution version's status.

```
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'])
```

SDK for Java 2.x

To create a solution version, use the following `createPersonalizeSolutionVersion` method and pass as a parameter the Amazon Resource Name (ARN) of the solution from [Step 2: Configuring a solution \(p. 127\)](#). The following code uses the [DescribeSolutionVersion \(p. 289\)](#) operation to retrieve the solution version's status.

```
public static String createPersonalizeSolutionVersion(PersonalizeClient
personalizeClient, String solutionArn) {
    long maxTime = 0;
    long waitInMilliseconds = 30 * 1000; // 30 seconds
    String solutionStatus = "";
    String solutionVersionStatus = "";
    String solutionVersionArn = "";

    try {
        DescribeSolutionRequest describeSolutionRequest =
DescribeSolutionRequest.builder()
            .solutionArn(solutionArn)
            .build();

        maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

        // Wait until solution is active.
        while (Instant.now().getEpochSecond() < maxTime) {

            solutionStatus =
personalizeClient.describeSolution(describeSolutionRequest).solution().status();
            System.out.println("Solution status: " + solutionStatus);

            if (solutionStatus.equals("ACTIVE") || solutionStatus.equals("CREATE
FAILED")) {
                break;
            }
        }
    } catch (Exception e) {
        System.out.println("Error creating solution version: " + e.getMessage());
    }
}
```

```

try {
    Thread.sleep(waitInMilliseconds);
} catch (InterruptedException e) {
    System.out.println(e.getMessage());
}
}

// Once the solution is active, start creating a solution version.

if (solutionStatus.equals("ACTIVE")) {

    CreateSolutionVersionRequest createSolutionVersionRequest =
CreateSolutionVersionRequest.builder()
    .solutionArn(solutionArn)
    .build();

    CreateSolutionVersionResponse createSolutionVersionResponse =
personalizeClient.createSolutionVersion(createSolutionVersionRequest);
    solutionVersionArn =
createSolutionVersionResponse.solutionVersionArn();

    System.out.println("Solution version ARN: " + solutionVersionArn);

    DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
    .solutionVersionArn(solutionVersionArn)
    .build();

    maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

    while (Instant.now().getEpochSecond() < maxTime) {

        // Use the solution version ARN to get the solution version status.
        solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest).solutionVersion().status
        System.out.println("Solution version status: " +
solutionVersionStatus);

        if (solutionVersionStatus.equals("ACTIVE") ||
solutionVersionStatus.equals("CREATE FAILED")) {
            break;
        }
        try {
            Thread.sleep(waitInMilliseconds);
        } catch (InterruptedException e) {
            System.out.println(e.getMessage());
        }
    }
    return solutionVersionArn;
}
} catch(PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return "";
}

```

To check the current solution version status, call the [DescribeSolutionVersion \(p. 289\)](#) operation and pass the ARN of the solution version returned from the `CreateSolutionVersion` operation. Training is complete when the status is ACTIVE and you can evaluate it using metrics supplied by Amazon

Personalize. For more information, see [Step 4: Evaluating a solution version with metrics \(p. 145\)](#). If training does not complete because of an error, you are not charged for the training.

If your solution version has a status of CREATE_PENDING or CREATE_IN_PROGRESS, you can use the [StopSolutionVersionCreation \(p. 325\)](#) operation to stop the solution version creation process. See [Stopping the creation of a solution version \(p. 143\)](#).

Stopping the creation of a solution version

If your solution version has a status of CREATE_PENDING or CREATE_IN_PROGRESS, you can use the Amazon Personalize console or the [StopSolutionVersionCreation \(p. 325\)](#) operation to stop creating the solution version (stop training a model). You can't resume creating a solution version after it has stopped. You are billed for resources used up to the point when the creation of the solution version stopped.

Stopping the creation of a solution version ends model training, but doesn't delete the solution version. You can still view the solution version details in the Amazon Personalize console and with the [DescribeSolutionVersion \(p. 289\)](#) operation.

You can stop the solution version creation process with the Amazon Personalize console, the AWS Command Line Interface (AWS CLI), or the AWS SDKs.

Topics

- [Stopping the creation of a solution version \(console\) \(p. 143\)](#)
- [Stopping the creation of a solution version \(AWS CLI\) \(p. 143\)](#)
- [Stopping the creation of a solution version \(AWS SDKs\) \(p. 144\)](#)

Stopping the creation of a solution version (console)

If your solution version has a status of CREATE_PENDING or CREATE_IN_PROGRESS, you can stop creating a solution version (stop training a model).

To stop creating a solution version (console)

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> and sign into your account.
2. On the **Dataset groups** page, choose the dataset group with the solution version that you want to stop.
3. In the navigation pane, choose **Solutions and recipes**.
4. On the **Solution and recipes** page, choose the solution with the solution version that you want to stop.
5. In **Solution versions**, choose the solution version that you want to stop.
6. On the solution version details page, choose **Stop creation**. Depending on the original state of the solution version, the solution version state changes as follows:
 - CREATE_PENDING changes to CREATE_STOPPED.
 - CREATE_IN_PROGRESS changes to CREATE_STOPPING and then CREATE_STOPPED.

Stopping the creation of a solution version (AWS CLI)

If your solution version has a status of CREATE_PENDING or CREATE_IN_PROGRESS, you can stop creating a solution version (stop training a model). Use the following `stop-solution-version-creation` command to stop creating the solution version with the AWS CLI. Replace `solution`

version arn with the Amazon Resource Name (ARN) of the solution version that you want to stop. You are billed for resources used up to the point that creation of the solution version stopped.

```
aws personalize stop-solution-version-creation \
--solution-version-arn solution version arn
```

Check the training status of the solution version with the `describe-solution-version` command.

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

Depending on the original state of the solution version, the solution version state changes as follows:

- `CREATE_PENDING` changes to `CREATE_STOPPED`.
- `CREATE_IN_PROGRESS` changes to `CREATE_STOPPING` and then `CREATE_STOPPED`

Stopping the creation of a solution version (AWS SDKs)

If your solution version has a status of `CREATE_PENDING` or `CREATE_IN_PROGRESS`, you can stop creating a solution version (stop training a model). The following code shows how to stop creating a solution version with the AWS SDK for Python (Boto3) or AWS SDK for Java 2.x. You are billed for resources used up to the point when creation of the solution version stopped.

SDK for Python (Boto3)

Use the following `stop_solution_version_creation` method to stop creation of a solution version. Replace `solution_version_arn` with the Amazon Resource Name (ARN) of the solution version that you want to stop. The method uses the [DescribeSolutionVersion \(p. 289\)](#) operation to retrieve the solution version's status.

```
import boto3

personalize = boto3.client('personalize')

response = personalize.stop_solution_version_creation(
    solutionVersionArn = 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'])
```

SDK for Java 2.x

Use the following `stopSolutionVersionCreation` method to stop creating a solution version. Pass as parameters an Amazon Personalize service client and the Amazon Resource Name (ARN) of the solution version that you want to stop creating. The following code uses the [DescribeSolutionVersion \(p. 289\)](#) operation to retrieve the solution version's status.

```
public static void stopSolutionVersionCreation(PersonalizeClient personalizeClient,
    String solutionVersionArn) {
    String solutionVersionStatus = "";

    StopSolutionVersionCreationRequest stopSolutionVersionCreationRequest =
        StopSolutionVersionCreationRequest.builder()
```

```
.solutionVersionArn(solutionVersionArn)
.build();

personalizeClient.stopSolutionVersionCreation(stopSolutionVersionCreationRequest);

// Use the solution version ARN to get the solution version status.
DescribeSolutionVersionRequest describeSolutionVersionRequest =
DescribeSolutionVersionRequest.builder()
    .solutionVersionArn(solutionVersionArn)
    .build();

solutionVersionStatus =
personalizeClient.describeSolutionVersion(describeSolutionVersionRequest)
    .solutionVersion()
    .status();
System.out.println("Solution version status: " + solutionVersionStatus);
}
```

Depending on the original state of the solution version, the solution version state changes as follows:

- CREATE_PENDING changes to CREATE_STOPPED.
- CREATE_IN_PROGRESS changes to CREATE_STOPPING and then CREATE_STOPPED.

Step 4: Evaluating a solution version with metrics

You can evaluate the performance of your solution version through offline and online metrics. *Online metrics* are the empirical results you observe in your users' interactions with real-time recommendations. For example, you might record your users' click-through rate as they browse your catalog. You are responsible for generating and recording any online metrics.

Offline metrics are the metrics Amazon Personalize generates when you train a solution version. You can use offline metrics to evaluate the performance of the model before you create a campaign and provide recommendations. Offline metrics allow you to view the effects of modifying a solution's hyperparameters or compare results from solutions that use the same training data but use different recipes. For the rest of this section, the term metrics refers to *offline metrics*.

To get performance metrics, Amazon Personalize splits the input interactions data into two sets: a training set and a testing set. The training set consists of 90% of each user's interactions data. The testing set consists of the remaining 10% of each user's interactions data. Amazon Personalize then creates the solution version using the training set.

When training completes, Amazon Personalize gives the solution version the oldest 90% of each user's data from the testing set as input. Amazon Personalize then calculates metrics by comparing the recommendations the solution version generates to the actual interactions in the newest 10% of each user's data from the testing set.

To generate a baseline for comparison purposes, we recommend using the [Popularity-Count \(p. 106\)](#) 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.

Retrieving Metrics

You retrieve the metrics for a specific solution version by calling the [GetSolutionMetrics \(p. 292\)](#) operation.

Retrieve metrics using the AWS Python SDK

1. Create a solution version. For more information, see [Creating a solution \(p. 96\)](#).
2. Use the following code to retrieve metrics.

```
import boto3

personalize = boto3.client('personalize')

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

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

The following is an example the output from a solution version created using the [User-Personalization \(p. 98\)](#) recipe with an additional optimization objective.

```
{
    "solutionVersionArn": "arn:aws:personalize:us-west-2:acct-id:solution/MovieSolution/
<version-id>",
    "metrics": {
        "coverage": 0.27,
        "mean_reciprocal_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,
        "average_rewards_at_k": 0.653
    }
}
```

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 \(p. 340\)](#) call.

For each metric, higher numbers are better.

coverage

An evaluation metric that tells you the proportion of unique items that Amazon Personalize might recommend using your model out of the total number of unique items in Interactions and Items datasets. To make sure Amazon Personalize recommends more of your items, use a model with a higher coverage score. Recipes that feature item exploration, such as User-Personalization, have higher coverage than those that don't, such as popularity-count.

mean reciprocal rank at 25

An evaluation metric that assesses the relevance of a model's highest ranked recommendation. Amazon Personalize calculates this metric using the average accuracy of the model when ranking the most relevant recommendation out of the top 25 recommendations over all requests for recommendations.

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

normalized discounted cumulative gain (NCDG) at K (5/10/25)

An evaluation metric that tells you about the relevance of your model's highly ranked recommendations, where K is a sample size of 5, 10, or 25 recommendations. Amazon Personalize calculates this by assigning weight to recommendations based on their position in a ranked list, where each recommendation is discounted (given a lower weight) by a factor dependent on its position. The normalized discounted cumulative gain at K assumes that recommendations that are lower on a list are less relevant than recommendations higher on the list.

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

An evaluation metric that tells you how relevant your model's recommendations are based on a sample size of K (5, 10, or 25) recommendations. Amazon Personalize calculates this metric based on the number of relevant recommendations out of the top K recommendations, divided by K, where K is 5, 10, or 25.

This metric rewards precise recommendation of the relevant items.

average_rewards_at_k

When you create a solution version (train a model) for a solution with an optimization objective, Amazon Personalize generates an average_rewards_at_k metric. The score for average_rewards_at_k tells you how well the solution version performs in achieving your objective. To calculate this metric, Amazon Personalize calculates the rewards for each user as follows:

```
rewards_per_user = total rewards from the user's interactions with their top 25 reward generating recommendations / total rewards from the user's interactions with recommendations
```

The final average_rewards_at_k is the average of all rewards_per_user normalized to be a decimal value less than or equal to 1 and greater than 0. The closer the value is to 1, the more gains on average per user you can expect from recommendations.

For example, if your objective is to maximize revenue from clicks, Amazon Personalize calculates each user score by dividing total revenue generated by the items the user clicked from their top 25 most expensive recommendations by the revenue from all of the recommended items the user clicked. Amazon Personalize then returns a normalized average of all user scores. The closer the average_rewards_at_k is to 1, the more revenue on average you can expect to gain per user from recommendations.

For more information see [Optimizing a solution for an additional objective \(p. 130\)](#).

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 \(p. 149\)](#).

Creating a campaign

For real-time recommendations, after you complete [Preparing and importing data \(p. 54\)](#) and [Creating a solution \(p. 96\)](#), you are ready to deploy your solution version to generate recommendations. You deploy a solution version by creating an Amazon Personalize campaign. If you are getting batch recommendations, you don't need to create a campaign. For more information see [Getting batch recommendations \(p. 163\)](#).

A campaign is a deployed solution version (trained model) with provisioned dedicated transaction capacity for creating real-time recommendations for your application users. After you create a campaign, you use the [GetRecommendations \(p. 340\)](#) or [GetPersonalizedRanking \(p. 336\)](#) API operations to get recommendations.

You create a campaign with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs.

Important

If you manually retrain your solution version or want to change your campaign settings, you must update your campaign. For more information see [Updating a campaign \(p. 153\)](#).

Topics

- [Minimum provisioned transactions per second and auto-scaling \(p. 149\)](#)
- [Creating a campaign \(console\) \(p. 149\)](#)
- [Creating a campaign \(AWS CLI\) \(p. 150\)](#)
- [Creating a campaign \(AWS SDKs\) \(p. 151\)](#)
- [Updating a campaign \(p. 153\)](#)

Minimum provisioned transactions per second and auto-scaling

When you create an Amazon Personalize campaign, you specify a dedicated transaction capacity for creating real-time recommendations for your application users. 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 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.

Creating a campaign (console)

After your solution version status is Active you are ready to deploy it with an Amazon Personalize campaign.

To create a campaign (console)

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> and sign into your account.
2. Choose the dataset group with the solution version you want to deploy.
3. In the navigation pane, choose **Campaigns**.
4. On the **Campaigns** page, choose **Create campaign**.
5. 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 [Minimum provisioned transactions per second and auto-scaling \(p. 149\)](#).
6. If you used the User-Personalization recipe, in **Campaign configuration** optionally enter values for the **Exploration weight** and **Exploration item age cut off**. For more information see [User-Personalization \(p. 98\)](#).
7. Choose **Create campaign**.
8. 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 [Getting recommendations \(p. 156\)](#).

The campaign is ready when its status is ACTIVE. If you retrain your solution version or want to change your campaign settings, you must update your campaign. For more information see [Updating a campaign \(p. 153\)](#).

Creating a campaign (AWS CLI)

After your solution version status is Active, you are ready to deploy it with an Amazon Personalize campaign. Use the following `create-campaign` AWS CLI command to create a campaign that deploys a solution version trained using the User-Personalization recipe. Give the campaign a name and specify the solution version ARN (Amazon Resource Name). Optionally change the `minProvisionedTPS` if your use case requires a higher provisioned capacity. The minimum value is 1.

The `campaign-config` parameters are specific to the recipe that you used to train the solution version (for more information about recipes see [Step 1: Choosing a recipe \(p. 96\)](#)). The example uses the following User-Personalization recipe specific `itemExplorationConfig` fields with their default values: `explorationWeight` and `explorationItemAgeCutOff`. If you omit the `campaign-config` parameter, the default values apply. For more information about the `itemExplorationConfig` fields, see the [Properties and hyperparameters \(p. 99\)](#) for the [User-Personalization \(p. 98\)](#) recipe.

```
aws personalize create-campaign \
--name campaign name \
--solution-version-arn solution version arn \
--min-provisioned-tps 1 \
--campaign-config "{\"itemExplorationConfig\":{\"explorationWeight\":\"0.3\", \
\"explorationItemAgeCutOff\":\"30\"}}"
```

The campaign is ready when its status is ACTIVE. To get the current status, call [DescribeCampaign \(p. 266\)](#) and check that the `status` field is ACTIVE.

If you retrain your solution version or want to change your campaign settings, you must update your campaign. For more information see [Updating a campaign \(p. 153\)](#).

Amazon Personalize provides operations for managing campaigns such as [ListCampaigns \(p. 297\)](#) to list the campaigns you have created. You can delete a campaign by calling [DeleteCampaign \(p. 248\)](#). 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 \(p. 156\)](#).

Creating a campaign (AWS SDKs)

After your solution version status is Active you are ready to deploy it with an Amazon Personalize campaign. Use the following code to create a campaign with the AWS SDK for Python (Boto3) or AWS SDK for Java 2.x.

The example code uses the following parameters (for a complete list of parameters, see [CreateCampaign \(p. 218\)](#)):

- A name for the campaign.
- The solution version's ARN (Amazon Resource Name).
- The [Minimum provisioned TPS \(p. 149\)](#) the campaign will support (the minimum value for this parameter is 1).
- *Optional* campaign configuration parameters `itemExplorationWeight` and `explorationItemAgeCutOff`.

The campaign configuration parameters are specific to the recipe that you used to train the solution version (for more information about recipes see [Step 1: Choosing a recipe \(p. 96\)](#)). In this example, the `itemExplorationWeight` and `explorationItemAgeCutOff` parameters are specific to the [User-Personalization \(p. 98\)](#) recipe. The default `itemExplorationWeight` is 0.3 and the default `explorationItemAgeCutOff` is 30. If you leave out campaign configuration parameters, the default values apply.

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

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'])
```

SDK for Java 2.x

```
public static void createCampaign(PersonalizeClient personalizeClient,
```

```
        String campaignName,
        String solutionVersionArn,
        Integer minProvisionedTPS,
        String itemExplorationWeight,
        String explorationItemAgeCutoff) {

    //Optional code to instantiate a HashMap and add the explorationWeight and
    explorationItemAgeCutOff values.
    //Remove if you aren't using User-Personalization.
    Map<String, String> itemExploration = new HashMap<String, String>();
    itemExploration.put("explorationWeight", itemExplorationWeight);
    itemExploration.put("explorationItemAgeCutOff", explorationItemAgeCutoff);

    try {
        // Build a User-Personalization recipe specific campaignConfig object with the
        itemExploration map.
        // CampaignConfig construction will vary by recipe.
        CampaignConfig campaignConfig = CampaignConfig.builder()
            .itemExplorationConfig(itemExploration)
            .build();

        // build the createCampaignRequest
        CreateCampaignRequest createCampaignRequest = CreateCampaignRequest.builder()
            .name(campaignName)
            .solutionVersionArn(solutionVersionArn)
            .minProvisionedTPS(minProvisionedTPS)
            .campaignConfig(campaignConfig) //
            .build();

        // create the campaign
        CreateCampaignResponse campaignResponse =
personalizeClient.createCampaign(createCampaignRequest);
        String campaignArn = campaignResponse.campaignArn();

        DescribeCampaignRequest campaignRequest = DescribeCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        DescribeCampaignResponse campaignResponse =
personalizeClient.describeCampaign(campaignRequest);
        Campaign newCampaign = campaignResponse.campaign();

        System.out.println("The Campaign status is " + newCampaign.status());

    } catch (PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

The campaign is ready when its status is ACTIVE. To get the current status, call [DescribeCampaign \(p. 266\)](#) and check that the `status` field is ACTIVE.

If you manually retrain your solution version or want to change your campaign settings, you must update your campaign. For more information see [Updating a campaign \(p. 153\)](#).

Amazon Personalize provides operations for managing campaigns such as [ListCampaigns \(p. 297\)](#) to list the campaigns you have created. You can delete a campaign by calling [DeleteCampaign \(p. 248\)](#). 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 \(p. 156\)](#).

Updating a campaign

To deploy a retrained solution version with an existing campaign or to change your campaign's [Minimum provisioned TPS \(p. 149\)](#) or campaign configuration, you must manually update the campaign.

With User-Personalization, Amazon Personalize automatically updates your latest solution version (trained with `trainingMode` set to `FULL`) every two hours to include new items in recommendations, and your campaign automatically uses the updated solution version. Manually update a campaign only when you manually retrain the solution version with `trainingMode` set to `FULL`, or when you want to make changes to your campaign's `minProvisionedTPS` or campaign configuration. For more information on automatic updates with the User-Personalization recipe see [Automatic updates \(p. 98\)](#).

You manually update a campaign with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs.

Topics

- [Updating a campaign \(console\) \(p. 153\)](#)
- [Updating a campaign \(AWS CLI\) \(p. 153\)](#)
- [Updating a campaign \(AWS SDKs\) \(p. 154\)](#)

Updating a campaign (console)

To deploy a manually retrained solution version or make changes to your campaign configuration, you must update your campaign.

To update a campaign (console)

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> and sign into your account.
2. Choose the dataset group with the campaign you want to update.
3. In the navigation pane, choose **Campaigns**.
4. On the **Campaigns** page, choose the campaign you want to update.
5. On the campaign details page, choose **Update**.
6. On the **Update campaign** page, make your changes. For example, if you are deploying a retrained solution version, for **Solution version ID**, choose the identification number for the new solution version.
7. Choose **Update**. Amazon Personalize updates the campaign to use the new solution version and any changed configurations.

Updating a campaign (AWS CLI)

To deploy a new solution version, change your campaign's [Minimum provisioned TPS \(p. 149\)](#), or change your campaign's configuration, you must update your campaign. Use the following `update-campaign` command to update a campaign to use a new solution version with the AWS CLI.

Replace `campaign Arn` with the Amazon Resource Name (ARN) of the campaign you want to update. Replace `new solution version Arn` with the solution version you want to deploy.

```
aws personalize update-campaign \
--campaign-arn campaign Arn \
--solution-version-arn new solution version Arn \
```

```
--min-provisioned-tps 1
```

Updating a campaign (AWS SDKs)

To deploy a new solution version, change your campaign's [Minimum provisioned TPS \(p. 149\)](#) or change your campaign's configuration, you must update your campaign. Use the following code to update a campaign with the SDK for Python (Boto3) or SDK for Java 2.x. For a complete list of parameters, see [UpdateCampaign \(p. 327\)](#).

SDK for Python (Boto3)

Use the following `update_campaign` method to deploy a new solution version. Replace `campaignArn` with the Amazon Resource Name (ARN) of the campaign you want to update, replace the `new solution version arn` with the new solution version ARN and optionally change the `minProvisionedTPS`.

```
import boto3

personalize = boto3.client('personalize')

response = personalize.update_campaign(
    campaignArn = 'campaign arn',
    solutionVersionArn = 'new solution version arn',
    minProvisionedTPS = 1,
)
arn = response['campaignArn']

description = personalize.describe_campaign(campaignArn = arn)['campaign']
print('Name: ' + description['name'])
print('ARN: ' + description['campaignArn'])
print('Status: ' + description['status'])
```

SDK for Java 2.x

Use the following `updateCampaign` method to update a campaign to use a new solution version. Pass as parameters an Amazon Personalize service client, the new solution version's Amazon Resource Name (ARN), and the [Minimum provisioned TPS \(p. 149\)](#).

```
public static void updateCampaign(PersonalizeClient personalizeClient,
                                    String campaignArn,
                                    String solutionVersionArn,
                                    Integer minProvisionedTPS) {

    try {
        // build the updateCampaignRequest
        UpdateCampaignRequest updateCampaignRequest = UpdateCampaignRequest.builder()
            .campaignArn(campaignArn)
            .solutionVersionArn(solutionVersionArn)
            .minProvisionedTPS(minProvisionedTPS)
            .build();

        // update the campaign
        personalizeClient.updateCampaign(updateCampaignRequest);

        DescribeCampaignRequest campaignRequest = DescribeCampaignRequest.builder()
            .campaignArn(campaignArn)
            .build();

        DescribeCampaignResponse campaignResponse =
            personalizeClient.describeCampaign(campaignRequest);
```

```
Campaign updatedCampaign = campaignResponse.campaign();

System.out.println("The Campaign status is " + updatedCampaign.status());

} catch (PersonalizeException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

Getting recommendations

With Amazon Personalize, you can get recommendations in real-time or you can get batch recommendations. For real-time recommendations, you must create a campaign before you get recommendations. For batch recommendations, you don't need to create a campaign. For information on campaigns see [Creating a campaign \(p. 149\)](#).

The following topics explain how and when to use each recommendation type.

Topics

- [Recommendation scores \(p. 156\)](#)
- [Getting real-time recommendations \(p. 156\)](#)
- [Getting batch recommendations \(p. 163\)](#)
- [Filtering recommendations \(p. 170\)](#)

Recommendation scores

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 \(p. 156\)](#) and [Getting batch recommendations \(p. 163\)](#).

Getting real-time recommendations

You get real-time recommendations from Amazon Personalize with a campaign. For example, suppose you have a campaign that deploys a solution to give movie recommendations. Depending on the recipe you used to create the solution version backing the campaign, you get recommendations for your users with the [the section called "GetRecommendations" \(p. 340\)](#) or the section called ["GetPersonalizedRanking" \(p. 336\)](#) API operations or with the Amazon Personalize console. For information on campaigns see [Creating a campaign \(p. 149\)](#). If you don't need recommendations in real-time, you can get batch recommendations without a campaign. For more information see [Getting batch recommendations \(p. 163\)](#).

If your campaign recommends similar items, you get recommendations with the [the section called "GetRecommendations" \(p. 340\)](#) operation or with the Amazon Personalize console. Amazon Personalize returns a list of related items based on the item ID you include in the request.

Topics

- [Increasing recommendation relevance with contextual metadata \(p. 157\)](#)
- [Getting recommendations \(p. 157\)](#)
- [Getting a personalized ranking \(p. 160\)](#)

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 you get recommendations or get a personalized ranking.

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

- The schema of the Interactions dataset must have contextual metadata fields (see [Datasets and schemas \(p. 42\)](#)).
- The solution version backing the campaign must use a recipe of type USER_PERSONALIZATION or PERSONALIZED_RANKING (see [Step 1: Choosing a recipe \(p. 96\)](#)).

For more information about the benefits of including contextual metadata, see [Contextual metadata \(p. 45\)](#).

For examples that show how to include contextual metadata using the SDK for Python (Boto3), see [Getting recommendations using contextual metadata \(AWS Python SDK\) \(p. 160\)](#) and [Getting a personalized ranking using contextual metadata \(AWS Python SDK\) \(p. 162\)](#).

Getting recommendations

You get personalized recommendations or similar item recommendations from an Amazon Personalize campaign. You can get recommendations with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs.

Note

If you used a PERSONALIZED_RANKING recipe, see [Getting a personalized ranking \(p. 160\)](#).

Topics

- [How scoring works \(p. 157\)](#)
- [Getting recommendations \(console\) \(p. 158\)](#)
- [Getting recommendations \(AWS CLI\) \(p. 158\)](#)
- [Getting recommendations \(AWS SDKs\) \(p. 159\)](#)
- [Getting recommendations using contextual metadata \(AWS Python SDK\) \(p. 160\)](#)

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 .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 $w_{i,j}$ are user and item embeddings respectively, and the Greek letter sigma (Σ) represents summation over all items in the item dataset:

$$\text{score}(u, i) = \frac{\exp(\bar{w}_u^\top w_i)}{\sum_j \exp(\bar{w}_u^\top w_j)}$$

Note

Amazon Personalize doesn't show scores for SIMS or Popularity-Count-based models.

Getting recommendations (console)

To get recommendations with the Amazon Personalize console, provide the recommendation request information on the details page for your Campaign.

To get recommendations for a user

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> and sign into your account.
2. Choose the dataset group that contains the campaign you are using.
3. In the navigation pane, choose **Campaigns**.
4. On the **Campaigns** page, choose the target campaign.
5. Under **Test campaign results**, enter your recommendation request details based on the recipe you used. For **USER_PERSONALIZATION** recipes, enter the **User ID** of the user that you want to get recommendations for. For **RELATED_ITEMS** recipes, enter the **Item ID** of the item you want Amazon Personalize to use to determine similar items.

If you recorded events for a user before they logged in (an anonymous user), you can get recommendations for this user by providing the **sessionId** from those events instead of a **userId**. For more information about recording events for anonymous users, see [PutEvents operation \(p. 84\)](#).

6. Optionally choose a filter. For more information, see [Filtering recommendations \(p. 170\)](#).
7. If your campaign uses contextual metadata (for requirements see [Increasing recommendation relevance with contextual metadata \(p. 157\)](#)) optionally provide context data.
For each context, for the **Key**, enter the metadata field, and for the **Value**, enter the context data.
8. Choose **Get recommendations**. A table containing the user's top 25 recommended items displays.

Getting recommendations (AWS CLI)

Use the following code to get recommendations. Change the value of **userId** to a user ID that is in the data that you used to train the solution. A list of the top 10 recommended items for the user displays. To change the number of recommended items, change the value for **numResults**. The default is 25 items. The maximum is 500 items. If you used a **RELATED_ITEMS** recipe to train the solution version backing the campaign, replace the **user-id** parameter with **item-id** and specify the item ID.

If you recorded events for a user before they logged in (an anonymous user), you can get recommendations for this user by providing the **sessionId** from those events instead of a **userId**. For more information about recording events for anonymous users, see [PutEvents operation \(p. 84\)](#).

```
aws personalize-runtime get-recommendations \
--campaign-arn campaign arn \
--user-id User ID \
--num-results 10
```

Getting recommendations (AWS SDKs)

The following code shows how to get Amazon Personalize recommendations using the SDK for Python (Boto3) or SDK for Java 2.x. Change the value of `userId` to a user ID that is in the data that you used to train the solution. A list of the top 10 recommended items for the user displays. To change the number of recommended items, change the value for `numResults`. The default is 25 items. The maximum is 500 items. If you used a RELATED_ITEMS recipe to train the solution version backing the campaign, replace the `userId` parameter with `itemId` and specify the item ID.

If you recorded events for a user before they logged in (an anonymous user), you can get recommendations for this user by providing the `sessionId` from those events instead of a `userId`. For more information about recording events for anonymous users, see [PutEvents operation \(p. 84\)](#).

SDK for Python (Boto3)

```
import boto3

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

response = personalizeRt.get_recommendations(
    campaignArn = 'Campaign ARN',
    userId = 'User ID',
    numResults = 10
)

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

SDK for Java 2.x

```
public static void getRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
String campaignArn, String userId){

    try {
        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
            .campaignArn(campaignArn)
            .numResults(20)
            .userId(userId)
            .build();

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient.getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();

        for (PredictedItem item: items) {
            System.out.println("Item Id is : "+item.itemId());
            System.out.println("Item score is : "+item.score());
        }
    } catch (AwsServiceException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

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 displays.

```
import boto3

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

response = personalizeRt.get_recommendations(
    campaignArn = 'Campaign ARN',
    userId = 'User ID',
    context = {
        'DEVICE': 'mobile phone'
    }
)

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

Getting a personalized ranking

A personalized ranking is a list of recommended items that are re-ranked for a specific user. To get personalized rankings, call the [GetPersonalizedRanking \(p. 336\)](#) API operation or get recommendations from a campaign in the console.

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 \(p. 96\)](#).

Topics

- [How personalized ranking scoring works \(p. 160\)](#)
- [Getting a personalized ranking \(console\) \(p. 161\)](#)
- [Getting a personalized ranking \(AWS CLI\) \(p. 161\)](#)
- [Getting a personalized ranking \(AWS SDKs\) \(p. 161\)](#)
- [Getting a personalized ranking using contextual metadata \(AWS Python SDK\) \(p. 162\)](#)
- [Personalized-Ranking sample notebook \(p. 163\)](#)

How personalized ranking 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:

$$\text{score}(u, i) = \frac{\exp(\bar{w}_u^\top w_i)}{\sum_{j \in \text{input}} \exp(\bar{w}_u^\top w_j)}$$

Getting a personalized ranking (console)

To get a personalized ranking for a user from the Amazon Personalize console, choose the campaign that you are using and then provide their user ID, specify the list of items you want ranked for the user, optionally choose a filter, and optionally provide any context data.

To get a personalized ranking for a user

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> and sign into your account.
2. Choose the dataset group that contains the campaign you are using.
3. In the navigation pane, choose **Campaigns**.
4. On the **Campaigns** page, choose the target campaign.
5. Under **Test campaign results**, enter the **User ID** of the user that you want to get recommendations for.
6. For **Item IDs**, enter the list of items to be ranked for the user.
7. Optionally choose a filter. For more information, see [Filtering recommendations \(p. 170\)](#).
8. If your campaign uses contextual metadata (for requirements see [Increasing recommendation relevance with contextual metadata \(p. 157\)](#)) optionally provide context data.
For each context, for the **Key**, enter the metadata field, and for the **Value**, enter the context data.
9. Choose **Get personalized item rankings**. A table containing the items ranked in order of predicted interest for the user appears.

Getting a personalized ranking (AWS CLI)

Use the following get-personalized-ranking command to get a personalized ranking with the AWS CLI. Specify the Amazon Resource Name (ARN) for your campaign, the User ID for the user, and provide a list of item IDs for the items to be ranked for the user (each separated by a space). The items to be ranked must be in the data that you used to train the solution version. A list of ranked recommendations displays. Amazon Personalize considers the first item in the list of most interest to the user.

```
aws personalize-runtime get-personalized-ranking \
--campaign-arn Campaign ARN \
--user-id 12 \
--input-list 3 4 10 8 12 7
```

Getting a personalized ranking (AWS SDKs)

The following code shows how to get a personalized ranking with the AWS SDK for Python (Boto3) or AWS SDK for Java 2.x.

SDK for Python (Boto3)

Use the following `get_personalized_ranking` method to get a personalized ranking with the SDK for Python (Boto3). Change the value of `userId` and `inputList` to a user ID and list of item IDs to be ranked for the user. The item IDs must be in the data that you used to train the solution version. A list of ranked recommendations is displayed. Amazon Personalize considers the first item in the list of most interest to the user.

```
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'])

```

SDK for Java 2.x

Use the following `getRankedRecs` method to get a personalized ranking with the SDK for Java 2.x. Pass the following as parameters: an Amazon Personalize runtime client, your campaign's Amazon Resource Name (ARN), the user ID for the user, and a list of item IDs to be ranked for the user. The item IDs must be in the data you used to train the solution. The method returns the list of recommended items ranked from most relevant to least relevant.

```

public static List<PredictedItem> getRankedRecs(PersonalizeRuntimeClient
personalizeRuntimeClient,
                                                String campaignArn,
                                                String userId,
                                                ArrayList<String> items) {

    try {
        GetPersonalizedRankingRequest rankingRecommendationsRequest =
GetPersonalizedRankingRequest.builder()
        .campaignArn(campaignArn)
        .userId(userId)
        .inputList(items)
        .build();

        GetPersonalizedRankingResponse recommendationsResponse =
personalizeRuntimeClient.getPersonalizedRanking(rankingRecommendationsRequest);
        List<PredictedItem> rankedItems =
recommendationsResponse.personalizedRanking();
        int rank = 1;
        for (PredictedItem item : rankedItems) {
            System.out.println("Item ranked at position " + rank + " details");
            System.out.println("Item Id is : " + item.itemId());
            System.out.println("Item score is : " + item.score());
            System.out.println("-----");
            rank++;
        }
        return rankedItems;
    } catch (PersonalizeRuntimeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return null;
}

```

Getting a personalized ranking 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.

```
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

To get recommendations for large datasets that do not require real-time updates, you can use an asynchronous batch workflow with historical data only. For example, you might get product recommendations for all users on an email list, or get [item-to-item similarities \(SIMS\) \(p. 124\)](#) across an inventory. You don't need to create an Amazon Personalize campaign to get batch recommendations.

To get batch recommendations with Amazon Personalize, you use a batch inference job. A *batch inference job* is a tool that imports your batch input data from an Amazon S3 bucket, uses your solution version to generate recommendations, and exports the recommendations to an Amazon S3 bucket. We recommend that you use a different location for your output data (either a folder or a different Amazon S3 bucket). To use data you record in real time with the PutEvents API operation, you must retrain your solution version before creating the batch inference job.

The batch workflow is as follows:

1. Prepare and upload a list of users or items (or both) in JSON format to an Amazon S3 bucket. The format of your input data depends on the recipe you use. See [Preparing and importing batch input data \(p. 164\)](#).
2. Create a separate location for your output data, either a folder or a different Amazon S3 bucket.
3. Create a batch inference job. See [Creating a batch inference job \(console\) \(p. 166\)](#), [Creating a batch inference job \(AWS CLI\) \(p. 167\)](#), or [Creating a batch inference job \(AWS SDKs\) \(p. 167\)](#).
4. When the batch inference job is complete, retrieve the recommendations from your output location in Amazon S3.

Topics

- [Batch workflow permissions requirements \(p. 164\)](#)
- [Batch workflow scoring \(p. 164\)](#)
- [Preparing and importing batch input data \(p. 164\)](#)
- [Creating a batch inference job \(console\) \(p. 166\)](#)

- [Creating a batch inference job \(AWS CLI\) \(p. 167\)](#)
- [Creating a batch inference job \(AWS SDKs\) \(p. 167\)](#)

Batch workflow permissions requirements

For batch workflows, your Amazon Personalize IAM service role needs permission to access and add files to your Amazon S3 buckets. For information on granting permissions, see [Service-linked role policy for batch workflows \(p. 66\)](#). For more information on bucket permissions, see [User policy examples](#) in the [Amazon Simple Storage Service \(S3\) Developer Guide](#).

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 service-linked role permission to use your key. You must also add Amazon Personalize as a Principle in your AWS KMS key policy. For more information, see [Using key policies in AWS KMS](#) in the [AWS Key Management Service Developer Guide](#).

Batch workflow scoring

Item scores calculated by batch inference jobs are calculated just as described in [Getting real-time recommendations \(p. 156\)](#). You can view scores in the batch inference job's output JSON file. Scores are only returned by models trained with the HRNN and Personalize-Ranking recipes.

Preparing and importing batch input data

A batch inference job uses a solution version to make recommendations based on the users or items (or both) that you provide in an input JSON file. Before you can get batch recommendations, you must prepare and upload your JSON file to an Amazon S3 bucket. We recommend that you create an output folder in your Amazon S3 bucket or use a separate output Amazon S3 bucket. This allows you to run multiple batch inference jobs using the same input data location.

To prepare and import data

1. Format your batch input data depending on the recipe your solution uses. Your input data is a JSON file with either a list of userids (USER_PERSONALIZATION recipes), a list of itemids (RELATED_ITEMS recipes), or both (PERSONALIZED_RANKING recipes). Separate each user, item, or user and list of items with a new line. For examples, see [Input and output JSON examples \(p. 164\)](#).
2. Upload your input JSON to an input folder in your Amazon S3 bucket. For more information, see [Uploading files and folders by using drag and drop](#) in the [Amazon Simple Storage Service User Guide](#).
3. Create a separate location for your output data, either a folder or a different Amazon S3 bucket. Creating a separate location for the output JSON allows you to run multiple batch inference jobs using the same input data location.

Input and output JSON examples

How you format your input data depends on the recipe you use. The following are correctly formatted JSON input and output examples for each recipe type.

User-Personalization and legacy HRNN recipes

Input

Separate each `userId` with a new line as follows.

```
{"userId": "4638"}
```

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

Output

```
{"input":{"userId":"4638"},"output":{"recommendedItems":  
["63992","115149","110102","148626","148888","31685","102445","69526","92535","143355","62374","745  
[0.0152238,0.0069081,0.0068222,0.006394,0.0059746,0.0055851,0.0049357,0.0044644,0.0042968,0.004015,  
 {"input":{"userId":"663"},"output":{"recommendedItems":  
["368","377","25","780","1610","648","1270","6","165","1196","1097","300","1183","608","104","474",  
 [0.0406197,0.0372557,0.0254077,0.0151975,0.014991,0.0127175,0.0124547,0.0116712,0.0091098,0.0085492  
 {"input":{"userId":"3384"},"output":{"recommendedItems":  
["597","21","223","2144","208","2424","594","595","920","104","520","367","2081","39","1035","2054"  
 [0.0241061,0.0119394,0.0118012,0.010662,0.0086972,0.0079428,0.0073218,0.0071438,0.0069602,0.0056961  
 ...
```

Popularity-Count

Input

Separate each `userId` with a new line as follows.

```
{"userId": "12"}  
 {"userId": "105"}  
 {"userId": "41"}  
 ...
```

Output

```
{"input": {"userId": "12"}, "output": {"recommendedItems": ["105", "106", "441"]}}  
 {"input": {"userId": "105"}, "output": {"recommendedItems": ["105", "106", "441"]}}  
 {"input": {"userId": "41"}, "output": {"recommendedItems": ["105", "106", "441"]}}  
 ...
```

PERSONALIZED_RANKING recipes

Input

Separate each `userId` and list of `itemIds` to be ranked with a new line as follows.

```
{"userId": "891", "itemList": ["27", "886", "101"]}  
 {"userId": "445", "itemList": ["527", "55", "901"]}  
 {"userId": "71", "itemList": ["27", "351", "101"]}  
 ...
```

Output

```
{"input": {"userId": "891", "itemList": ["27", "886", "101"]}, "output": {"recommendedItems":  
 ["27", "101", "886"], "scores": [0.48421, 0.28133, 0.23446]}  
 {"input": {"userId": "445", "itemList": ["527", "55", "901"]}, "output": {"recommendedItems":  
 ["901", "527", "55"], "scores": [0.46972, 0.31011, 0.22017]}  
 {"input": {"userId": "71", "itemList": ["29", "351", "199"]}, "output": {"recommendedItems":  
 ["351", "29", "199"], "scores": [0.68937, 0.24829, 0.06232]}}  
 ...
```

RELATED_ITEMS recipes

Input

Separate each `itemId` with a new line as follows.

```
{"itemId": "105"}  
 {"itemId": "106"}  
 {"itemId": "441"}  
 ...
```

Output

```
{"input": {"itemId": "105"}, "output": {"recommendedItems": ["106", "107", "49"]}}  
 {"input": {"itemId": "106"}, "output": {"recommendedItems": ["105", "107", "49"]}}  
 {"input": {"itemId": "441"}, "output": {"recommendedItems": ["2", "442", "435"]}}  
 ...
```

Creating a batch inference job (console)

After you have completed [Preparing and importing batch input data \(p. 164\)](#), you are ready to create a batch inference job. This procedure assumes that you have already created a solution and a solution version (trained model). To get batch recommendations, create a batch inference job and specify the job name,

To get batch recommendations (console)

1. Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> and sign in to your account.
2. On the **Dataset groups** page, choose your dataset group.
3. Choose **Batch inference jobs** in the navigation pane, then choose **Create batch inference job**.
4. In **Batch inference job details**, in **Batch inference job name**, specify a name for your batch inference job.
5. For **IAM service role**, choose the IAM service role you created for Amazon Personalize during set up. This role must have read and write access to your input and output Amazon S3 buckets respectively.
6. For **Solution**, choose the solution and then choose the **Solution version ID** that you want to use to generate the recommendations.
7. For **Input data configuration**, specify the Amazon S3 path to your input file.

Use the following syntax: `s3://<name of your S3 bucket>/<folder name>/<input JSON file name>`

Your input data must be in the correct format for the recipe your solution uses. For input data examples see [Input and output JSON examples \(p. 164\)](#).

8. For **Output data configuration**, specify the path to your output location. We recommend using a different location for your output data (either a folder or a different Amazon S3 bucket).
- Use the following syntax: `s3://<name of your S3 bucket>/<output folder name>/`
9. For **Filter configuration** optionally choose a filter to apply a filter to the recommendations added to the output JSON file. For more information see [Filtering batch recommendations \(p. 182\)](#).
 10. Choose **Create batch inference job**. Batch inference job creation starts and the **Batch inference jobs** page appears with the **Batch inference job detail** section displayed.
 11. 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`.

Creating a batch inference job (AWS CLI)

After you have completed [Preparing and importing batch input data \(p. 164\)](#), you are ready to create a batch inference job using the following `create-batch-inference-job` code. Specify a job name, replace `Solution version ARN` with the Amazon Resource Name (ARN) of your solution version, and replace the `IAM service role ARN` with the ARN of the IAM service role you created for Amazon Personalize during set up. This role must have read and write access to your input and output Amazon S3 buckets respectively.

Replace `S3 input path` and `S3 output path` with the Amazon S3 path to your input file and output locations. We recommend using a different location for your output data (either a folder or a different Amazon S3 bucket). You can apply a filter to the recommendations added to the output JSON file. For more information see [Filtering batch recommendations \(p. 182\)](#).

Use the following syntax for input and output locations: `s3://<name of your S3 bucket>/<folder name>/<input JSON file name>` and `s3://<name of your S3 bucket>/<output folder name>/`.

The example includes optional 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 \(p. 98\)](#).

```
aws personalize create-batch-inference-job --job-name Batch job name \
    --solution-version-arn Solution version ARN \
    --job-input s3DataSource={path=s3://S3 input path} \
    --job-output s3DataDestination={path=s3://S3 output path} \
    --role-arn IAM service role ARN \
    --batch-inference-job-config "{\"itemExplorationConfig\": \
    {\"explorationWeight\":\"0.3\", \"explorationItemAgeCutOff\":\"30\"}}"
{
    "batchInferenceJobArn": "arn:aws:personalize:us-west-2:acct-id:batch-inference-job/
batchInferenceJobName"
}
```

Creating a batch inference job (AWS SDKs)

After you have completed [Preparing and importing batch input data \(p. 164\)](#), you are ready to create a batch inference job with the [CreateBatchInferenceJob \(p. 214\)](#) operation. The following code shows how to create a batch inference job using the AWS SDK for Python (Boto3) or AWS SDK for Java 2.x.

Use the following syntax for input and output locations: `s3://<name of your S3 bucket>/<folder name>/<input JSON file name>` and `s3://<name of your S3 bucket>/<output folder name>/`.

SDK for Python (Boto3)

Use the following `create_batch_inference_job` code to create a batch inference job. Specify a `Batch job name`, replace `Solution version ARN` with the Amazon Resource Name (ARN) of your solution version, and replace the `IAM service role ARN` with the ARN of the IAM service role you created for Amazon Personalize during set up. This role must have read and write access to your input and output Amazon S3 buckets respectively.

Replace Amazon S3 data source and data destinations with the Amazon S3 path to your input file and output locations. We recommend using a different location for your output data (either a folder or a different Amazon S3 bucket). You can apply a filter to the recommendations added to the output JSON file. For more information see [Filtering batch recommendations \(p. 182\)](#).

The example includes optional 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 \(p. 98\)](#).

```
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 service role ARN",
    batchInferenceJobConfig = {
        # optional USER_PERSONALIZATION recipe hyperparameters
        "itemExplorationConfig": {
            "explorationWeight": "0.3",
            "explorationItemAgeCutOff": "30"
        }
    },
    jobInput =
        {"s3DataSource": {"path": "s3://<name of your S3 bucket>/<folder name>/<input JSON file name>"}},
    jobOutput =
        {"s3DataDestination": {"path": "s3://<name of your S3 bucket>/<output folder name>/"}}
)
```

SDK for Java 2.x

Use the following `createPersonalizeBatchInferenceJob` method to create a batch inference job. Pass the following as parameters: an Amazon Personalize service client, the solution version's ARN (Amazon Resource Name), a name for the job, the Amazon S3 location where you stored your input data (`s3InputDataSourcePath`), the bucket-name/folder name of your output data location (`s3DataDestinationPath`), and your service-linked role's ARN (see [Creating an IAM service role for Amazon Personalize \(p. 10\)](#)). We recommend using a different location for your output data (either a folder or a different Amazon S3 bucket).

The example includes optional User-Personalization recipe specific `itemExplorationConfig` fields: `explorationWeight` and `explorationItemAgeCutOff`. Optionally include `explorationWeight` and `explorationItemAgeCutOff` values to configure exploration. For more information, see [User-Personalization recipe \(p. 98\)](#).

```
public static String createPersonalizeBatchInferenceJob(PersonalizeClient
personalizeClient,
                                                       String solutionVersionArn,
                                                       String jobName,
                                                       String s3InputDataSourcePath,
                                                       String s3DataDestinationPath,
                                                       String roleArn,
                                                       String explorationWeight,
                                                       String
explorationItemAgeCutOff) {

    long waitInMilliseconds = 60 * 1000;
    String status;
    String batchInferenceJobArn;

    try {
        // Set up data input and output parameters.
        S3DataConfig inputSource = S3DataConfig.builder()
            .path(s3InputDataSourcePath)
            .build();
    }
```

```

S3DataConfig outputDestination = S3DataConfig.builder()
    .path(s3DataDestinationPath)
    .build();

BatchInferenceJobInput jobInput = BatchInferenceJobInput.builder()
    .s3DataSource(inputSource)
    .build();
BatchInferenceJobOutput jobOutputLocation = BatchInferenceJobOutput.builder()
    .s3DataDestination(outputDestination)
    .build();

// Optional code to build the User-Personalization specific item exploration
config.
HashMap<String, String> explorationConfig = new HashMap<>();

explorationConfig.put("explorationWeight", explorationWeight);
explorationConfig.put("explorationItemAgeCutOff", explorationItemAgeCutOff);

BatchInferenceJobConfig jobConfig = BatchInferenceJobConfig.builder()
    .itemExplorationConfig(explorationConfig)
    .build();
// End optional User-Personalization recipe specific code.

CreateBatchInferenceJobRequest createBatchInferenceJobRequest =
CreateBatchInferenceJobRequest.builder()
    .solutionVersionArn(solutionVersionArn)
    .jobInput(jobInput)
    .jobOutput(jobOutputLocation)
    .jobName(jobName)
    .roleArn(roleArn)
    .batchInferenceJobConfig(jobConfig) // Optional
    .build();

batchInferenceJobArn =
personalizeClient.createBatchInferenceJob(createBatchInferenceJobRequest)
    .batchInferenceJobArn();
DescribeBatchInferenceJobRequest describeBatchInferenceJobRequest =
DescribeBatchInferenceJobRequest.builder()
    .batchInferenceJobArn(batchInferenceJobArn)
    .build();

long maxTime = Instant.now().getEpochSecond() + 3 * 60 * 60;

// wait until the batch inference job is complete.
while (Instant.now().getEpochSecond() < maxTime) {

    BatchInferenceJob batchInferenceJob = personalizeClient
        .describeBatchInferenceJob(describeBatchInferenceJobRequest)
        .batchInferenceJob();

    status = batchInferenceJob.status();
    System.out.println("Batch inference job status: " + status);

    if (status.equals("ACTIVE") || status.equals("CREATE FAILED")) {
        break;
    }
    try {
        Thread.sleep(waitInMilliseconds);
    } catch (InterruptedException e) {
        System.out.println(e.getMessage());
    }
}
return batchInferenceJobArn;

} catch (PersonalizeException e) {
    System.out.println(e.awsErrorDetails().errorMessage());
}

```

```
    }
    return "";
}
```

Processing the batch job might take a while to complete. You can check a job's status by calling [DescribeBatchInferenceJob \(p. 264\)](#) 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 \(p. 294\)](#).

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 \(p. 330\)](#) 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 \(p. 73\)](#).

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 \(p. 340\)](#) or [GetPersonalizedRanking \(p. 336\)](#) 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 \(p. 214\)](#) 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 \(p. 207\)](#).

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 \(p. 327\)](#) operation or create a new campaign.

Topics

- [Filter expressions \(p. 170\)](#)
- [Filtering real-time recommendations \(p. 174\)](#)
- [Filtering batch recommendations \(p. 182\)](#)

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 \(p. 172\)](#). For examples of filter expressions, see [Filter expression examples \(p. 173\)](#).

Note

Amazon Personalize ignores case only when matching event types.

Topics

- [Creating filter expressions \(p. 171\)](#)
- [Filter expression examples \(p. 173\)](#)

Creating filter expressions

The general structure of a filter expression is as follows:

```
EXCLUDE/INCLUDE ItemID WHERE dataset type.property IN/NOT IN (value/parameter)
```

You can either manually create filter expressions or get help with expression syntax and structure by using the [Expression builder \(p. 177\)](#) 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 considers up to 200 historical interactions for a user, and up to 100 streamed interactions you record for the user with the PutEvents operation.

When filtering, the number of *historical interactions* Amazon Personalize considers for a user depends on the `max_user_history_length_percentile` and `min_user_history_length_percentile` hyperparameters you defined before training.

For example, if you used .99 for the `max_user_history_length_percentile`, and 99% of your users have at most 4 interactions, Amazon Personalize will only filter based on the user's most recent 4 historical interactions. If a user has less than the number historical interactions at the `min_user_history_length_percentile`, Amazon Personalize doesn't consider the user's interactions when filtering.

To filter based on up to 200 historical interactions for a user, set the `max_user_history_length_percentile` to 1.0 and retrain the model. For more information on hyperparameters, see [Step 1: Choosing a recipe \(p. 96\)](#) and navigate to the *Properties and Hyperparameters* section for your recipe.

- **Items:** Use filter expressions to include or exclude items based on specific item conditions. You can't use filters to include or exclude items based on unstructured textual item metadata such as product descriptions.
- **Users:** Use filter expressions to include or exclude items based on specific properties of the user you are getting recommendations for (the `CurrentUser`). If you have created a Users dataset, you can add an `IF` condition to your expression to check conditions for the `CurrentUser` regardless of the dataset that is being used in the expression.

When creating a filter expression, note the following limitations:

- You can't use filters to include or exclude items based on unstructured textual item metadata such as product descriptions.
- 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 [Combining multiple expressions \(p. 174\)](#).

- You can't create filter expressions that filter using values with a boolean type in your schema. To filter based on boolean values, use a schema with a field of type *String* and use the values "True" and "False" in your data. Or you can use type *int* or *long* and values 0 and 1.

Filter expression elements

Use the following elements to create filter expressions:

INCLUDE or EXCLUDE

Use **INCLUDE** to limit recommendations to only items that meet the filter criteria *OR* use **EXCLUDE** to remove all items that meet the filter criteria.

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

To chain multiple conditions together within the same filter expression, use **AND** or **OR**. 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 **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. You can't use filters to include or exclude items based on textual item metadata.

IF condition

Use an **IF** condition *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**.

CurrentUser.property

To filter recommendations based on the user you are getting recommendations for, in *only* an **IF** condition, use **CurrentUser** and provide the user property. For example, **CurrentUser.AGE**.

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.

Comparison operators

Use **=**, **<**, **<=**, **>**, **>=** operators to test numerical data for equality.

Asterisk (*) character

Use ***** to include or exclude interactions of all types. Use ***** *only* for filter expressions that use the **EVENT_TYPE** property of an **Interactions** dataset.

Pipe separator

Use the pipe separator (**|**) to chain multiple expressions together. For more information, see [Combining multiple expressions \(p. 174\)](#).

Parameters

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 \(p. 207\)](#).

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) or event types 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 or categories 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 only items with a genre or genres that you specify when you get recommendations using the \$GENRE parameter.

```
INCLUDE ItemID WHERE Items.GENRE IN ($GENRE)
```

Users

The following expression excludes items with a genre or genres 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 only 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
```

Combining multiple expressions

To filter by Items and Interactions datasets with one filter, combine multiple expressions together using a pipe separator (|). Each expression is first evaluated independently and the result is either the union or the intersection of the two results.

Matching expressions example

If both expressions use `EXCLUDE` or both expressions use `INCLUDE`, the result is the union of the two results as follows (A and B are different expressions):

- `Exclude A | Exclude B` is equal to `Exclude result from A or result from B`
- `Include A | Include B` is equal to `Include result from A or result from B`

The following example shows how to combine two expressions that use `INCLUDE`. The first expression includes only items with a category or categories that you specify when you get recommendations using the `$CATEGORY` parameter. The second expression includes items the user has marked as a favorite. Recommendations will include only items with the category you specify along with items that the user has marked as a favorite.

```
INCLUDE ItemID WHERE Items.CATEGORY IN ($CATEGORY) | INCLUDE ItemID WHERE  
Interactions.EVENT_TYPE IN ("favorite")
```

INCLUDE and EXCLUDE example

If one or more expression uses `INCLUDE` and one more expression uses `EXCLUDE`, the result is the subtraction of the `EXCLUDE` expression result from the `INCLUDE` expression result as follows (A, B, C, and D are different expressions).

- `Include A | Exclude B` is equal to `Include result from A - result from B`
- `Include A | Include B | Exclude C | Exclude D` is equal to `Include (A or B) - (C or D)`

The following example shows how to combine an `INCLUDE` expression and a `EXCLUDE` expression. The first expression includes only items with a genre or genres that you specify when you get recommendations using the `$GENRE` parameter. The second expression excludes items that the user has clicked or streamed. Recommendations will include only items with a genre that you specify 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.

Topics

- [Filtering real-time recommendations \(console\) \(p. 175\)](#)
- [Filtering real-time recommendations \(AWS CLI\) \(p. 178\)](#)
- [Filtering real-time recommendations \(AWS SDKs\) \(p. 180\)](#)

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. 327) 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/home> 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.

The screenshot shows the 'Create filter' page in the Amazon Personalize console. The 'Filter configuration' section includes a 'Filter name' field containing 'your-filter-name'. The 'Expression' section has 'Build expression' selected. Below it is a 'Build expression info' panel and an 'Add expression' button. At the bottom right are 'Cancel' and 'Create filter' buttons.

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 \(p. 177\)](#).
 - To input your own expression, choose **Add expression manually**. For more information, see [Filter expression elements \(p. 172\)](#).

- 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 \(console\) \(p. 178\)](#).

Filter overview		
Filter name my-new-filter	Filter ARN arn:aws:personalize:us-west-2:<account number>:filter/my-new-filter	Status Active

Filter expression	
EXCLUDE ItemID WHERE Interactions.event_type IN ("click")	

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)

- In the navigation pane, choose **Campaigns**.
- On the **Campaigns** page, choose the target campaign.
- 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.

Item ID	Score
3948	0.0107270
1676	0.0069995
2657	0.0064348
2985	0.0055178
2081	0.0054022

- 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.
- 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.

The screenshot shows the 'Test campaign results' interface. At the top, there's a 'User ID info' section with a text input field containing '1'. Below it is a 'Filter name' section with a dropdown menu set to 'Remove-all-previously-purchased' and a 'Refresh' button. A note says 'To find a filter, go to the filter page.' At the bottom, a large orange button labeled 'Get recommendations' is visible. Below this button is a table with columns 'Item ID' and 'Score'. The table contains the following data:

Item ID	Score
3948	0.0107270
1676	0.0069995
2081	0.0054022
2641	0.0040574
1573	0.0039259

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

The screenshot shows the 'expression builder tool' interface. It has two main sections: 'expression builder tool:' and 'or prefer to input text.' Below these is a 'Build expression Info' section with a note: 'Build an expression using the fields below. For the value, enter a value (or comma separated values) to set the filter criteria, or enter "\$" + parameter name to add a placeholder parameter. When you get recommendations, you'll pass a value to this parameter to set the filter criteria.' The main area contains a table with columns 'Action', 'Property', 'Operator', and 'Value'. The 'Action' dropdown is set to 'Exclude'. The 'Property' dropdown is set to 'ItemID WHERE'. The 'Operator' dropdown is set to 'IN'. The 'Value' dropdown is set to 'Value or \${PARAMETER}'. Below this table are buttons for 'AND', 'Interactions.event_type', 'IN', 'Value or \${PARAMETER}', 'X', and '+'. At the bottom left is a 'Add expression' button.

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 X 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.

Build expression info

Build an expression using the fields below. For the value, enter a value (or comma separated values) to set the filter criteria, or enter "\$" + parameter name to add a placeholder parameter. When you get recommendations, you'll pass a value to this parameter to set the filter criteria.

Action	Property	Operator	Value
Exclude ▾	ItemID WHERE Items.GENRES	IN ▾	\$GENRES
AND ▾ Items.DOWNLOAD_COUNT ▾ > ▾ 200 X			
IF ▾ currentUser.AGE ▾ > ▾ 17 X +			

Add expression

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)

- Open the Amazon Personalize console at <https://console.aws.amazon.com/personalize/home> and sign into your account.
- From the **Dataset groups** list, choose the dataset group that contains the filter that you want to delete.
- In the navigation pane, choose **Filters**.
- From the list of filters, choose the filter that you want to delete and choose **View Details**. The filter details page appears.
- 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. 340) or [GetPersonalizedRanking](#) (p. 336) 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 \(p. 327\)](#) 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 `filter-expression` with your own filter expression.

```
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 \(p. 280\)](#) operation before you use the filter.

For more information about the API, see [CreateFilter \(p. 236\)](#). For more information about filter expressions, including examples, see [Creating filter expressions \(p. 171\)](#).

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 '{
    "Parameter name": "\\"value\\",
    "Parameter name": "\\"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 SDKs)

To filter recommendations using the AWS SDKs, you create a filter and then apply it by specifying the filter ARN in a [GetRecommendations \(p. 340\)](#) or [GetPersonalizedRanking \(p. 336\)](#) 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 \(p. 327\)](#) call or create a new campaign.

Creating a filter (AWS SDKs)

Create a new filter with the [CreateFilter \(p. 236\)](#) operation.

SDK for Python (Boto3)

Use the following `create_filter` method to create a filter with the AWS SDK for Python (Boto3). 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 \(p. 280\)](#) operation before using the filter. For more information about the API, see [CreateFilter \(p. 236\)](#). For more information about filter expressions, including examples, see [Creating filter expressions \(p. 171\)](#).

SDK for Java 2.x

The following `createFilter` method shows how to create a filter with the AWS SDK for Java 2.x. The method returns the ARN (Amazon Resource Number) of the new filter. Pass the following as parameters: an Amazon Personalize service client, a name for the filter, the dataset group where you want to create the filter, and the filter expression. For more information about filter expressions, including examples, see [Creating filter expressions \(p. 171\)](#).

```
public static String createFilter(PersonalizeClient personalizeClient,
                                    String filterName,
                                    String datasetGroupArn,
                                    String filterExpression) {
    try {
        CreateFilterRequest request = CreateFilterRequest.builder()
            .name(filterName)
            .datasetGroupArn(datasetGroupArn)
            .filterExpression(filterExpression)
            .build();

        return personalizeClient.createFilter(request).filterArn();
    }
    catch(PersonalizeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }
    return "";
}
```

Applying a filter (AWS SDKs)

When you use the `get_recommendations` or `get_personalized_ranking` methods, apply a filter by passing a `filterArn` and any filter values as parameters.

The following code shows how to filter recommendations with the AWS SDK for Python (Boto3) or the AWS SDK for Java 2.x.

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.

SDK for Python (Boto3)

Use the following `get_recommendations` method to filter recommendations with the SDK for Python (Boto3). 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\\"`.

```
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\\"
        ...
    }
)
```

SDK for Java 2.x

Use the following `getFilteredRecs` method to apply a filter to an Amazon Personalize recommendation request. Pass the following as parameters: an Amazon Personalize runtime service client, the campaign's ARN (Amazon Resource Name), the user's `userId`, the filter's ARN, and any filter parameter names (case sensitive) and their values. For example, if your filter expression has a `$GENRES` parameter, provide `"GENRES"` as the parameter name.

The following example uses two parameters, one with two values and one with one value. Depending on your filter expression, modify the code to add or remove `parameterName` and `parameterValue` fields.

```
public static void getFilteredRecs(PersonalizeRuntimeClient personalizeRuntimeClient,
```

```

        String campaignArn,
        String userId,
        String filterArn,
        String parameter1Name,
        String parameter1Value1,
        String parameter1Value2,
        String parameter2Name,
        String parameter2Value){

    try {

        Map<String, String> filterValues = new HashMap<>();

        filterValues.put(parameter1Name, String.format("\"%1$s\", \"%2$s\"",
            parameter1Value1, parameter1Value2));
        filterValues.put(parameter2Name, String.format("\"%1$s\"",
            parameter2Value));

        GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
            .campaignArn(campaignArn)
            .numResults(20)
            .userId(userId)
            .filterArn(filterArn)
            .filterValues(filterValues)
            .build();

        GetRecommendationsResponse recommendationsResponse =
personalizeRuntimeClient.getRecommendations(recommendationsRequest);
        List<PredictedItem> items = recommendationsResponse.itemList();

        for (PredictedItem item: items) {
            System.out.println("Item Id is : "+item.itemId());
            System.out.println("Item score is : "+item.score());
        }
    } catch (PersonalizeRuntimeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

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")

response = personalize.delete_filter(
    filterArn = "filter ARN"
)

```

Filtering batch recommendations

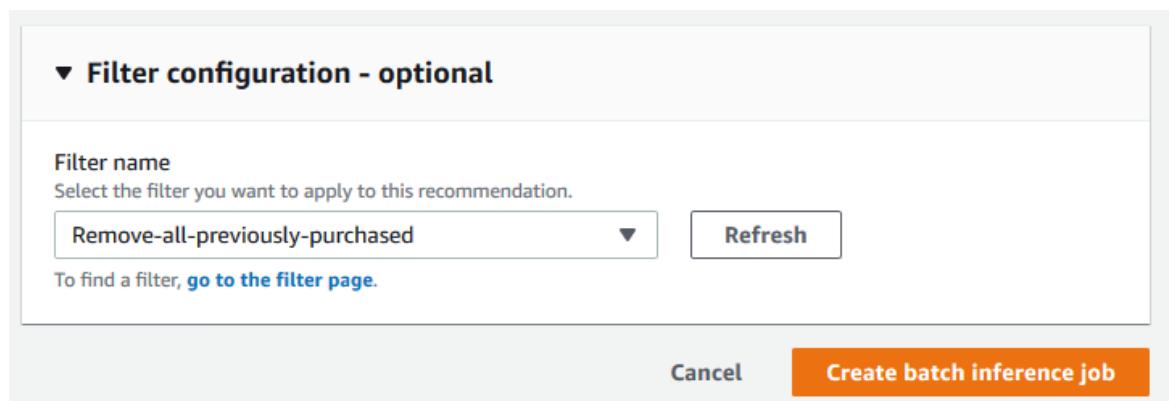
Filtering batch recommendations works nearly the same as filtering real-time recommendations. To filter batch recommendations, you [create a filter \(p. 174\)](#) and then apply it to a [CreateBatchInferenceJob \(p. 214\)](#) operation or new batch inference job in the Amazon Personalize console. Amazon Personalize then filters the recommendations from the batch job's output JSON file. For more information about batch inference jobs, see [Creating a batch inference job \(console\) \(p. 166\)](#).

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 \(p. 174\)](#).
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 SDKs to [create a filter \(p. 174\)](#).
2. Include the FilterArn parameter in the [CreateBatchInferenceJob \(p. 214\)](#) request.

```
import boto3

personalize = boto3.client("personalize")

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"}}
)
```

Maintaining recommendation relevance

Maintain the relevance of recommendations to increase user engagement, click-through rate, and conversion rate for your application as your catalogue grows. To maintain and improve the relevance of Amazon Personalize recommendations for your users, keep your data and solution versions up to date. This allows Amazon Personalize to learn from your user's most recent behavior and include your newest items in recommendations.

You can automate and schedule re-training and data import tasks with **Maintaining Personalized Experiences with Machine Learning**, an AWS Solutions Implementation that automates the Amazon Personalize workflow, including data import, solution version training, and batch workflows. For more information see [Maintaining Personalized Experiences with Machine Learning](#).

Topics

- [Keeping datasets current \(p. 184\)](#)
- [Keeping solution versions up to date \(p. 185\)](#)

Keeping datasets current

As your catalog grows, update your historical data with bulk or incremental data import operations. We recommend that you first import your records in bulk, and then incrementally add items and users as your catalog grows. For more information about importing historical data see [Preparing and importing data \(p. 54\)](#).

For real-time recommendations, keep your Interactions dataset up to date with your users' behavior by recording interaction *events* with an event tracker and the [PutEvents \(p. 330\)](#) operation. Amazon Personalize updates recommendations based on your user's most recent activity as they interact with your application. For more information on recording real-time events, see [Recording events \(p. 81\)](#).

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

- For *new events*, Amazon Personalize immediately uses historical and real-time interaction events between a user and existing items (items you included in the data you used to train the latest model) when generating recommendations for the same user. Historical events that you import using the Amazon Personalize console and events that you record in real-time influence recommendations in the same way. For more information, see [How real-time events influence recommendations \(p. 82\)](#).
- For *new items*, if you trained the solution version with User-Personalization, Amazon Personalize automatically updates the model every two hours. After each update, the new items can be included in recommendations with exploration. For information about exploration see [User-Personalization recipe \(p. 98\)](#).

For any other recipe, you must retrain the model for the new items to be included in recommendations.

- For *new users*, recommendations will initially be only for popular items. Starting with the first event, user recommendations will be more relevant as you record events. For more information, see [Recording events \(p. 81\)](#).

Keeping solution versions up to date

Create a new solution version (retrain the model) to include new items in recommendations and update the model with your user's most recent behavior. Once you retrain the model, you must update the campaign to deploy it. For more information see [Updating a campaign \(p. 153\)](#).

Retraining frequency depends on your business requirements and the recipe that you use. For most workloads, we recommend training a new model weekly with training mode set to **FULL**. This creates a completely new solution version based on the entirety of the training data from the datasets in your dataset group. If you add new items frequently and you don't use User-Personalization, you may need to do a full retraining more often to include those new items in recommendations.

With User-Personalization, Amazon Personalize automatically updates your latest fully trained solution version (trained with `trainingMode` set to **FULL**) every two hours to include new items in recommendations with exploration. Your campaign automatically uses the updated solution version. This is not a full retraining; you should still train a new solution version weekly with `trainingMode` set to **FULL** so the model can learn from your users' behavior.

If every two hours is not frequent enough, you can manually create a solution version with `trainingMode` set to **UPDATE** to include those new items in recommendations. Just remember that Amazon Personalize automatically updates only your latest fully trained solution version, so the manually updated solution version won't be automatically updated in the future. For more information, see [User-Personalization recipe \(p. 98\)](#).

For information on creating a new solution version, see [Step 3: Creating a solution version \(p. 139\)](#).

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. Amazon Personalize uses data encryption to protect your data. For more information see [Data encryption \(p. 187\)](#). 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 \(p. 186\)](#)
- [Identity and access management for Amazon Personalize \(p. 187\)](#)
- [Logging and monitoring in Amazon Personalize \(p. 199\)](#)
- [Compliance validation for Amazon Personalize \(p. 206\)](#)
- [Resilience in Amazon Personalize \(p. 206\)](#)
- [Infrastructure security in Amazon Personalize \(p. 206\)](#)

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 confidential or sensitive information, such as your customers' email addresses, into tags or 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 tags or free-form fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

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 \(p. 187\)](#)
- [Authenticating with identities \(p. 188\)](#)
- [Managing access using policies \(p. 190\)](#)
- [How Amazon Personalize works with IAM \(p. 191\)](#)
- [Amazon Personalize identity-based policy examples \(p. 194\)](#)
- [Troubleshooting Amazon Personalize identity and access \(p. 197\)](#)

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 \(p. 197\)](#).

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 \(p. 191\)](#).

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 \(p. 194\)](#).

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. 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 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. 191)
- [Amazon Personalize resource-based policies](#) (p. 193)
- [Authorization based on Amazon Personalize tags](#) (p. 193)
- [Amazon Personalize IAM roles](#) (p. 193)

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 \(p. 194\)](#).

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 \(p. 194\)](#)
- [AWS managed policies \(p. 195\)](#)
- [Using the Amazon Personalize console \(p. 195\)](#)
- [Allow users to view their own permissions \(p. 196\)](#)
- [Allowing full access to Amazon Personalize resources \(p. 196\)](#)
- [Allowing read-only access to Amazon Personalize resources \(p. 196\)](#)

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 \(p. 9\)](#)).

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 \(p. 199\)](#).

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.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "ViewOwnUserInfo",  
            "Effect": "Allow",  
            "Action": [  
                "iam:GetUserPolicy",  
                "iam>ListGroupsForUser",  
                "iam>ListAttachedUserPolicies",  
                "iam>ListUserPolicies",  
                "iam GetUser"  
            ],  
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]  
        },  
        {  
            "Sid": "NavigateInConsole",  
            "Effect": "Allow",  
            "Action": [  
                "iam:GetGroupPolicy",  
                "iam:GetPolicyVersion",  
                "iam GetPolicy",  
                "iam>ListAttachedGroupPolicies",  
                "iam>ListGroupPolicies",  
                "iam>ListPolicyVersions",  
                "iam>ListPolicies",  
                "iam>ListUsers"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

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.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "personalize:*"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

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.

```
{  
    "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.

Topics

- [I am not authorized to perform an action in Amazon Personalize \(p. 197\)](#)
- [I Am not authorized to perform iam:PassRole \(p. 198\)](#)
- [I want to view my access keys \(p. 198\)](#)
- [I'm an administrator and want to allow others to access Amazon Personalize \(p. 198\)](#)
- [I want to allow people outside of my AWS account to access my Amazon Personalize resources \(p. 199\)](#)

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/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

Important

Do not provide your access keys to a third party, even to help [find your canonical user ID](#). By doing this, you might give someone permanent access to your 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 \(p. 191\)](#).
- 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 \(p. 199\)](#)
- [CloudWatch metrics for Amazon Personalize \(p. 202\)](#)
- [Logging Amazon Personalize API calls with AWS CloudTrail \(p. 204\)](#)

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](#).

Topics

- [Using CloudWatch metrics for Amazon Personalize \(p. 199\)](#)
- [Accessing Amazon Personalize metrics \(p. 200\)](#)
- [Creating an alarm \(p. 201\)](#)
- [Amazon Personalize serverless monitoring app example \(p. 202\)](#)

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 AWS 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.

How do I?	Relevant metric
How do I track the number of events that have been recorded?	Monitor the <code>PutEventsRequests</code> metric.
How can I monitor the <code>DatasetImportJob</code> errors?	Use the <code>DatasetImportJobError</code> metric.
How can I monitor the latency of <code>GetRecommendations</code> calls?	Use the <code>GetRecommendationsLatency</code> metric.

You must have the appropriate CloudWatch permissions to monitor Amazon Personalize with CloudWatch. For more information, see [Authentication and access control for Amazon CloudWatch](#).

Accessing 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)

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
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 \
    --metric-name PutEventsRequests \
    --start-time 2019-03-15T00:00:20Z \
    --period 3600 \
    --end-time 2019-03-16T00:00:00Z \
    --namespace AWS/Personalize \
    --dimensions Name=EventTrackerArn,Value=EventTrackerArn \
    --statistics Sum
```

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](#).

Creating 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](#).

Amazon Personalize serverless monitoring app example

For an example app that adds monitoring, alerting, and optimization capabilities for Amazon Personalize see [Amazon Personalize monitor](#) in the [Amazon Personalize samples](#) repository.

CloudWatch metrics for Amazon Personalize

This section contains information about the Amazon CloudWatch metrics available for Amazon Personalize. For more information, see [Monitoring Amazon Personalize \(p. 199\)](#).

The following table lists the Amazon Personalize metrics. All metrics except GetRecommendations support these statistics: Average, Minimum, Maximum, Sum. GetRecommendations supports Sum only.

Metric	Description
DatasetImportJobRequests	<p>The number of successful CreateDatasetImportJob (p. 230) API calls.</p> <p>Dimensions: DatasetGroupArn, DatasetArn, DatasetImportJobArn</p>
DatasetImportJobError	<p>The number of CreateDatasetImportJob API calls that resulted in an error.</p> <p>Dimensions: DatasetGroupArn, DatasetArn, DatasetImportJobArn</p>
DatasetImportJobExecutionTime	<p>The time between the CreateDatasetImportJob API call and the completion (or failure) of the operation.</p> <p>Dimensions: DatasetGroupArn, DatasetArn, DatasetImportJobArn</p> <p>Unit: Seconds</p>
DatasetSize	<p>The size of data imported by the dataset import job.</p> <p>Dimensions: DatasetGroupArn, DatasetArn, DatasetImportJobArn</p> <p>Unit: Bytes</p>
SolutionTrainingJobRequests	<p>The number of successful CreateSolutionVersion (p. 245) API calls.</p> <p>Dimensions: SolutionArn, SolutionVersionArn</p>

Metric	Description
SolutionTrainingJobError	The number of CreateSolutionVersion API calls that resulted in an error. Dimensions: SolutionArn, SolutionVersionArn
SolutionTrainingJobExecutionTime	The time between the CreateSolutionVersion API call and the completion (or failure) of the operation. Dimensions: SolutionArn, SolutionVersionArn Unit: Seconds
GetPersonalizedRankingRequests	The number of successful GetPersonalizedRanking (p. 336) API calls. Dimension: CampaignArn
GetPersonalizedRanking4xxErrors	The number of GetPersonalizedRanking API calls that returned a 4xx HTTP response code. Dimension: CampaignArn
GetPersonalizedRanking5xxErrors	The number of GetPersonalizedRanking API calls that returned a 5xx HTTP response code. Dimension: CampaignArn
GetPersonalizedRankingLatency	The time between receiving the GetPersonalizedRanking API call and the sending of recommendations (excludes 4xx and 5xx errors). Dimension: CampaignArn Unit: Milliseconds
GetRecommendations	The number of successful GetRecommendations (p. 340) API calls. Dimension: CampaignArn
GetRecommendations4xxErrors	The number of GetRecommendations API calls that returned a 4xx HTTP response code. Dimension: CampaignArn
GetRecommendations5xxErrors	The number of GetRecommendations API calls that returned a 5xx HTTP response code. Dimension: CampaignArn
GetRecommendationsLatency	The time between receiving the GetRecommendations API call and the sending of recommendations (excludes 4xx and 5xx errors). Dimension: CampaignArn Unit: Milliseconds

Metric	Description
<code>PutEventsRequests</code>	The number of successful PutEvents (p. 330) API calls. Dimension: <code>EventTrackerArn</code>
<code>PutEvents4xxErrors</code>	The number of <code>PutEvents</code> API calls that returned a 4xx HTTP response code. Dimension: <code>EventTrackerArn</code>
<code>PutEvents5xxErrors</code>	The number of <code>PutEvents</code> API calls that returned a 5xx HTTP response code. Dimension: <code>EventTrackerArn</code>
<code>PutEventLatency</code>	The time taken for the completion of the <code>PutEvents</code> API call (excludes 4xx and 5xx errors). Dimension: <code>EventTrackerArn</code> Unit: Milliseconds

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 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 \(p. 211\)](#).

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:

```
{  
    "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",  
                "datasetGroupArn": "arn:aws:personalize:us-west-2:nnnnnnnnnn:dataset-group/testdataset",  
                "status": "ACTIVE",  
                "creationDateTime": "Nov 5, 2018 6:06:01 AM"  
            }  
        ]  
    },  
    "requestID": "request-id",  
    "eventID": "event-id",  
    "eventType": "AwsApiCall",  
    "recipientAccountId": "recipient-account-id"  
}
```

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.

Amazon Personalize endpoints and quotas

The following sections contain information about Amazon Personalize guidelines, quotas, and endpoints. For adjustable quotas, you can request a quota increase using the using the [Service Quotas console](#). For more information see [Requesting a quota increase \(p. 210\)](#).

Topics

- [Amazon Personalize endpoints and regions \(p. 207\)](#)
- [Compliance \(p. 207\)](#)
- [Service quotas \(p. 207\)](#)
- [Requesting a quota increase \(p. 210\)](#)

Amazon Personalize endpoints and regions

For a list of Amazon Personalize endpoints by region, see [AWS regions and endpoints](#) in the *Amazon Web Services General Reference*.

Compliance

For more information about Amazon Personalize compliance programs, see [AWS compliance](#), [AWS compliance programs](#), and [AWS services in scope by compliance program](#).

Service quotas

Your AWS account has the following quotas for Amazon Personalize.

Resource	Quota
Interactions	
Minimum number of unique combined historical and event interactions (after filtering by eventType and eventValueThreshold, if provided) required to train a model (create a solution version).	1000
Maximum number of interactions that are considered by a model during training.	500 million
Maximum number of distinct event types combined with total number of optional metadata columns in Interactions datasets.	10
Maximum number of metadata columns, excluding reserved fields, in Interactions datasets.	5

Resource	Quota
Maximum number of characters for categorical data values.	1000
Users	
Minimum number of unique users, with at least 2 interactions each, required to train a model (create a solution version).	25
Maximum number of users that are considered by a model during training.	50 million
Maximum number of metadata fields for a Users dataset.	5
Maximum number of characters for USER_ID data values.	256
Maximum number of characters for categorical data values.	1,000 characters
Items	
Maximum number of items that are considered by a model during training.	750,000
Maximum number of metadata fields for an Items dataset.	50
Maximum number of characters for ITEM_ID data values.	256
Maximum number of characters for categorical data values.	1,000 characters
Maximum number of characters for textual data values.	20,000 characters
Data import APIs	
Maximum rate of <code>PutEvents</code> requests.	1000/second
Maximum number of events in a <code>PutEvents</code> call.	10
Maximum size of an event.	10 KB
Maximum rate of <code>PutItems</code> requests.	10/second
Maximum number of items in a <code>PutItems</code> call.	10
Maximum rate of <code>PutUsers</code> requests.	10/second
Maximum number of users in a <code>PutUsers</code> call.	10
Recipes	
Maximum amount of data for an individual dataset (Users, Items, or Interactions) for HRNN, SIMS, Popularity-Count, and Personalized-Ranking recipes.	100 GB

Resource	Quota
Maximum amount of data for Interactions dataset for HRNN-metadata and HRNN-coldstart recipes.	100 GB
Maximum amount of combined data for Users and Items datasets for HRNN-metadata and HRNN-coldstart recipes.	5 GB
Maximum number of cold start items the HRNN-Coldstart recipe supports to train a model (create a solution version).	80,000
Minimum number of cold start items the HRNN-Coldstart recipe requires to train a model (create a solution version).	100
Filters	
Maximum number of filters per account	10 filters
Maximum number of parameters for a filter expression.	5 parameters
Maximum number of parameters across all filters in a dataset group.	10 parameters
GetRecommendations / GetPersonalizedRanking requests	
Maximum transaction rate (GetRecommendations and GetPersonalizedRanking requests).	2500/sec
Maximum number of GetRecommendations requests per second per campaign.	500/sec
Maximum number of GetPersonalizedRanking requests per second per campaign.	500/sec
Batch inference jobs	
Maximum number of input files in a batch inference job.	1000
Maximum size of batch inference job input.	1 GB
Maximum number of records per input file in a batch inference job.	50 million

Your AWS account has the following quotas per region for Amazon Personalize.

Resource	Quota
Total number of active schemas.	500
Total number of active dataset groups.	500
Total number of active event trackers.	500

Resource	Quota
Total number of active solutions.	500
Total number of active campaigns.	5
Total number of pending or in progress dataset import jobs.	5
Total number of pending or in progress batch inference jobs.	5
Total number of pending or in progress solution versions.	20

Requesting a quota increase

For adjustable quotas, you can request a quota increase using the [Service Quotas console](#). 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 `PutEvents` requests

To request a quota increase, use the [Service Quotas console](#) and follow the steps in the [Requesting a quota increase](#) section of the [Service Quotas User Guide](#).

API reference

This section provides documentation for the Amazon Personalize API operations. For a list of Amazon Personalize endpoints by region, see [AWS regions and endpoints](#) in the [AWS General Reference](#).

Topics

- [Actions \(p. 211\)](#)
- [Data Types \(p. 343\)](#)
- [Common Errors \(p. 433\)](#)
- [Common Parameters \(p. 435\)](#)

Actions

The following actions are supported by Amazon Personalize:

- [CreateBatchInferenceJob \(p. 214\)](#)
- [CreateCampaign \(p. 218\)](#)
- [CreateDataset \(p. 221\)](#)
- [CreateDatasetExportJob \(p. 224\)](#)
- [CreateDatasetGroup \(p. 227\)](#)
- [CreateDatasetImportJob \(p. 230\)](#)
- [CreateEventTracker \(p. 233\)](#)
- [CreateFilter \(p. 236\)](#)
- [CreateSchema \(p. 238\)](#)
- [CreateSolution \(p. 240\)](#)
- [CreateSolutionVersion \(p. 245\)](#)
- [DeleteCampaign \(p. 248\)](#)
- [DeleteDataset \(p. 250\)](#)
- [DeleteDatasetGroup \(p. 252\)](#)
- [DeleteEventTracker \(p. 254\)](#)
- [DeleteFilter \(p. 256\)](#)
- [DeleteSchema \(p. 258\)](#)
- [DeleteSolution \(p. 260\)](#)
- [DescribeAlgorithm \(p. 262\)](#)
- [DescribeBatchInferenceJob \(p. 264\)](#)
- [DescribeCampaign \(p. 266\)](#)
- [DescribeDataset \(p. 268\)](#)
- [DescribeDatasetExportJob \(p. 270\)](#)
- [DescribeDatasetGroup \(p. 272\)](#)
- [DescribeDatasetImportJob \(p. 274\)](#)
- [DescribeEventTracker \(p. 276\)](#)
- [DescribeFeatureTransformation \(p. 278\)](#)
- [DescribeFilter \(p. 280\)](#)
- [DescribeRecipe \(p. 282\)](#)

- [DescribeSchema \(p. 284\)](#)
- [DescribeSolution \(p. 286\)](#)
- [DescribeSolutionVersion \(p. 289\)](#)
- [GetSolutionMetrics \(p. 292\)](#)
- [ListBatchInferenceJobs \(p. 294\)](#)
- [ListCampaigns \(p. 297\)](#)
- [ListDatasetExportJobs \(p. 300\)](#)
- [ListDatasetGroups \(p. 303\)](#)
- [ListDatasetImportJobs \(p. 305\)](#)
- [ListDatasets \(p. 308\)](#)
- [ListEventTrackers \(p. 311\)](#)
- [ListFilters \(p. 313\)](#)
- [ListRecipes \(p. 315\)](#)
- [ListSchemas \(p. 317\)](#)
- [ListSolutions \(p. 319\)](#)
- [ListSolutionVersions \(p. 322\)](#)
- [StopSolutionVersionCreation \(p. 325\)](#)
- [UpdateCampaign \(p. 327\)](#)

The following actions are supported by Amazon Personalize Events:

- [PutEvents \(p. 330\)](#)
- [PutItems \(p. 332\)](#)
- [PutUsers \(p. 334\)](#)

The following actions are supported by Amazon Personalize Runtime:

- [GetPersonalizedRanking \(p. 336\)](#)
- [GetRecommendations \(p. 340\)](#)

Amazon Personalize

The following actions are supported by Amazon Personalize:

- [CreateBatchInferenceJob \(p. 214\)](#)
- [CreateCampaign \(p. 218\)](#)
- [CreateDataset \(p. 221\)](#)
- [CreateDatasetExportJob \(p. 224\)](#)
- [CreateDatasetGroup \(p. 227\)](#)
- [CreateDatasetImportJob \(p. 230\)](#)
- [CreateEventTracker \(p. 233\)](#)
- [CreateFilter \(p. 236\)](#)
- [CreateSchema \(p. 238\)](#)
- [CreateSolution \(p. 240\)](#)
- [CreateSolutionVersion \(p. 245\)](#)
- [DeleteCampaign \(p. 248\)](#)
- [DeleteDataset \(p. 250\)](#)

- [DeleteDatasetGroup \(p. 252\)](#)
- [DeleteEventTracker \(p. 254\)](#)
- [DeleteFilter \(p. 256\)](#)
- [DeleteSchema \(p. 258\)](#)
- [DeleteSolution \(p. 260\)](#)
- [DescribeAlgorithm \(p. 262\)](#)
- [DescribeBatchInferenceJob \(p. 264\)](#)
- [DescribeCampaign \(p. 266\)](#)
- [DescribeDataset \(p. 268\)](#)
- [DescribeDatasetExportJob \(p. 270\)](#)
- [DescribeDatasetGroup \(p. 272\)](#)
- [DescribeDatasetImportJob \(p. 274\)](#)
- [DescribeEventTracker \(p. 276\)](#)
- [DescribeFeatureTransformation \(p. 278\)](#)
- [DescribeFilter \(p. 280\)](#)
- [DescribeRecipe \(p. 282\)](#)
- [DescribeSchema \(p. 284\)](#)
- [DescribeSolution \(p. 286\)](#)
- [DescribeSolutionVersion \(p. 289\)](#)
- [GetSolutionMetrics \(p. 292\)](#)
- [ListBatchInferenceJobs \(p. 294\)](#)
- [ListCampaigns \(p. 297\)](#)
- [ListDatasetExportJobs \(p. 300\)](#)
- [ListDatasetGroups \(p. 303\)](#)
- [ListDatasetImportJobs \(p. 305\)](#)
- [ListDatasets \(p. 308\)](#)
- [ListEventTrackers \(p. 311\)](#)
- [ListFilters \(p. 313\)](#)
- [ListRecipes \(p. 315\)](#)
- [ListSchemas \(p. 317\)](#)
- [ListSolutions \(p. 319\)](#)
- [ListSolutionVersions \(p. 322\)](#)
- [StopSolutionVersionCreation \(p. 325\)](#)
- [UpdateCampaign \(p. 327\)](#)

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 \(p. 163\)](#).

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 \(p. 214\)](#)

The configuration details of a batch inference job.

Type: [BatchInferenceJobConfig \(p. 355\)](#) object

Required: No

[filterArn \(p. 214\)](#)

The ARN of the filter to apply to the batch inference job. For more information on using filters, see [Filtering Batch Recommendations..](#)

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.*:.*:+

Required: No

[jobInput \(p. 214\)](#)

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 \(p. 356\)](#) object

Required: Yes

[jobName \(p. 214\)](#)

The name of the batch inference job to create.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

Required: Yes

[jobOutput \(p. 214\)](#)

The path to the Amazon S3 bucket where the job's output will be stored.

Type: [BatchInferenceJobOutput \(p. 357\)](#) object

Required: Yes

[numResults \(p. 214\)](#)

The number of recommendations to retrieve.

Type: Integer

Required: No

[roleArn \(p. 214\)](#)

The ARN of the Amazon Identity and Access Management role that has permissions to read and write to your input and output Amazon S3 buckets respectively.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-zA-Z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-\/_]+

Required: Yes

[solutionVersionArn \(p. 214\)](#)

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

```
{  
    "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. 215)

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 V2](#)
- [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. 266).

Note

Wait until the `status` of the campaign is `ACTIVE` before asking the campaign for recommendations.

Related APIs

- [ListCampaigns](#) (p. 297)
- [DescribeCampaign](#) (p. 266)
- [UpdateCampaign](#) (p. 327)
- [DeleteCampaign](#) (p. 248)

Request Syntax

```
{  
    "campaignConfig": {  
        "itemExplorationConfig": {  
            "string" : "string"  
        }  
    },  
    "minProvisionedTPS": number,  
    "name": "string",  
    "solutionVersionArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[campaignConfig \(p. 218\)](#)

The configuration details of a campaign.

Type: [CampaignConfig \(p. 362\)](#) object

Required: No

[minProvisionedTPS \(p. 218\)](#)

Specifies the requested minimum provisioned transactions (recommendations) per second that Amazon Personalize will support.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

[name \(p. 218\)](#)

A name for the new campaign. The campaign name must be unique within your account.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

Required: Yes

[solutionVersionArn \(p. 218\)](#)

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

```
{  
    "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 \(p. 219\)](#)

The Amazon Resource Name (ARN) of the campaign.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.*:.*:+

Errors

InvalidArgumentException

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 V2](#)
- [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 \(p. 230\)](#) 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 \(p. 268\)](#).

Related APIs

- [CreateDatasetGroup \(p. 227\)](#)
- [ListDatasets \(p. 308\)](#)
- [DescribeDataset \(p. 268\)](#)
- [DeleteDataset \(p. 250\)](#)

Request Syntax

```
{  
    "datasetGroupArn": "string",  
    "datasetType": "string",  
    "name": "string",  
    "schemaArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn (p. 221)

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 \(p. 221\)](#)

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 \(p. 221\)](#)

The name for the dataset.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

Required: Yes

[schemaArn \(p. 221\)](#)

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 \(p. 222\)](#)

The ARN of the dataset.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.*:.*:.*

Errors

InvalidArgumentException

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 V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

CreateDatasetExportJob

Service: Amazon Personalize

Creates a job that exports data from your dataset to an Amazon S3 bucket. To allow Amazon Personalize to export the training data, you must specify an service-linked IAM role that gives Amazon Personalize `PutObject` permissions for your Amazon S3 bucket. For information, see [Exporting a dataset](#) in the Amazon Personalize developer guide.

Status

A dataset export job can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED

To get the status of the export job, call [DescribeDatasetExportJob \(p. 270\)](#), and specify the Amazon Resource Name (ARN) of the dataset export job. The dataset export 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.

Request Syntax

```
{  
    "datasetArn": "string",  
    "ingestionMode": "string",  
    "jobName": "string",  
    "jobOutput": {  
        "s3DataDestination": {  
            "kmsKeyArn": "string",  
            "path": "string"  
        }  
    },  
    "roleArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetArn (p. 224)

The Amazon Resource Name (ARN) of the dataset that contains the data to export.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:+`

Required: Yes

ingestionMode (p. 224)

The data to export, based on how you imported the data. You can choose to export only `BULK` data that you imported using a dataset import job, only `PUT` data that you imported incrementally (using the console, `PutEvents`, `PutUsers` and `PutItems` operations), or `ALL` for both types. The default value is `PUT`.

Type: String

Valid Values: `BULK` | `PUT` | `ALL`

Required: No

[jobName \(p. 224\)](#)

The name for the dataset export job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

Required: Yes

[jobOutput \(p. 224\)](#)

The path to the Amazon S3 bucket where the job's output is stored.

Type: [DatasetExportJobOutput \(p. 374\)](#) object

Required: Yes

[roleArn \(p. 224\)](#)

The Amazon Resource Name (ARN) of the IAM service role that has permissions to add data to your output Amazon S3 bucket.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-zA-Z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-\/_]+

Required: Yes

Response Syntax

```
{  
    "datasetExportJobArn": "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.

[datasetExportJobArn \(p. 225\)](#)

The Amazon Resource Name (ARN) of the dataset export job.

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

LimitExceeded

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 V2](#)
- [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 \(p. 221\)](#) 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 \(p. 272\)](#). 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 \(p. 221\)](#)
- [CreateEventTracker \(p. 233\)](#)
- [CreateSolution \(p. 240\)](#)

Related APIs

- [ListDatasetGroups \(p. 303\)](#)
- [DescribeDatasetGroup \(p. 272\)](#)
- [DeleteDatasetGroup \(p. 252\)](#)

Request Syntax

```
{  
  "kmsKeyArn": "string",  
  "name": "string",  
  "roleArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[kmsKeyArn \(p. 227\)](#)

The Amazon Resource Name (ARN) of a AWS Key Management Service (KMS) key used to encrypt the datasets.

Type: String

Pattern: `arn:aws.*:kms.*:[0-9]{12}:key/.*`

Required: No

[name \(p. 227\)](#)

The name for the new dataset group.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

Required: Yes

[roleArn \(p. 227\)](#)

The ARN of the AWS Identity and Access Management (IAM) role that has permissions to access the AWS Key Management Service (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-zA-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 \(p. 228\)](#)

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

InvalidArgumentException

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 V2](#)
- [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 IAM service role that has permission to read from the data source, as Amazon Personalize makes a copy of your data and processes it internally. For information on granting access to your Amazon S3 bucket, see [Giving Amazon Personalize Access to Amazon S3 Resources](#).

Important

The dataset import job replaces any existing data in the dataset that you imported in bulk.

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 \(p. 274\)](#), 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 \(p. 305\)](#)
- [DescribeDatasetImportJob \(p. 274\)](#)

Request Syntax

```
{  
    "datasetArn": "string",  
    "dataSource": {  
        "dataLocation": "string"  
    },  
    "jobName": "string",  
    "roleArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetArn (p. 230)

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 \(p. 230\)](#)

The Amazon S3 bucket that contains the training data to import.

Type: [DataSource \(p. 390\)](#) object

Required: Yes

[jobName \(p. 230\)](#)

The name for the dataset import job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

Required: Yes

[roleArn \(p. 230\)](#)

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 \(p. 231\)](#)

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

LimitExceeded

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 V2](#)
- [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. 276).

Note

The event tracker must be in the ACTIVE state before using the tracking ID.

Related APIs

- [ListEventTrackers](#) (p. 311)
- [DescribeEventTracker](#) (p. 276)
- [DeleteEventTracker](#) (p. 254)

Request Syntax

```
{  
    "datasetGroupArn": "string",  
    "name": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[datasetGroupArn](#) (p. 233)

The Amazon Resource Name (ARN) of the dataset group that receives the event data.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:(\w{1,2}\d{-})+\:personalize\:\.*\:\.*`

Required: Yes

[name](#) (p. 233)

The name for the event tracker.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

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 \(p. 234\)](#)

The ARN of the event tracker.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.*:.*:+

[trackingId \(p. 234\)](#)

The ID of the event tracker. Include this ID in requests to the [PutEvents](#) API.

Type: String

Length Constraints: Maximum length of 256.

Errors

InvalidArgumentException

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 V2](#)
- [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 \(p. 170\)](#).

Request Syntax

```
{  
    "datasetGroupArn": "string",  
    "filterExpression": "string",  
    "name": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[datasetGroupArn \(p. 236\)](#)

The ARN of the dataset group that the filter will belong to.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-zA-Z-]+):personalize:.*:.*:+

Required: Yes

[filterExpression \(p. 236\)](#)

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 \(p. 170\)](#).

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2500.

Required: Yes

[name \(p. 236\)](#)

The name of the filter to create.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

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 \(p. 236\)](#)

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 V2](#)
- [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 \(p. 221\)](#).

For more information on schemas, see [Datasets and Schemas](#).

Related APIs

- [ListSchemas \(p. 317\)](#)
- [DescribeSchema \(p. 284\)](#)
- [DeleteSchema \(p. 258\)](#)

Request Syntax

```
{  
    "name": "string",  
    "schema": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[name \(p. 238\)](#)

The name for the schema.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

Required: Yes

[schema \(p. 238\)](#)

A schema in Avro JSON format.

Type: String

Length Constraints: Maximum length of 10000.

Required: Yes

Response Syntax

```
{  
    "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. 238)

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 V2](#)
- [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 \(p. 245\)](#) 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 \(p. 292\)](#). When you are satisfied with the version, you deploy it using [CreateCampaign \(p. 218\)](#). 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.

Note

Amazon Personalize doesn't support configuring the `hpObjective` for solution hyperparameter optimization at this time.

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 \(p. 286\)](#). Wait until the status shows as ACTIVE before calling `CreateSolutionVersion`.

Related APIs

- [ListSolutions \(p. 319\)](#)
- [CreateSolutionVersion \(p. 245\)](#)
- [DescribeSolution \(p. 286\)](#)
- [DeleteSolution \(p. 260\)](#)

- [ListSolutionVersions \(p. 322\)](#)
- [DescribeSolutionVersion \(p. 289\)](#)

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"
},
"hpConfig": {
    "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"
    }
},
"optimizationObjective": {
    "itemAttribute": "string",
    "objectiveSensitivity": "string"
}
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn (p. 240)

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:

[eventType \(p. 240\)](#)

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 \(p. 240\)](#)

The name for the solution.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

Required: Yes

[performAutoML \(p. 240\)](#)

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 \(p. 240\)](#)

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 \(p. 240\)](#)

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 \(p. 240\)](#)

The configuration to use with the solution. When `performAutoML` is set to `true`, Amazon Personalize only evaluates the `autoMLConfig` section of the solution configuration.

Note

Amazon Personalize doesn't support configuring the `hpoObjective` at this time.

Type: [SolutionConfig \(p. 419\)](#) object

Required: No

Response Syntax

```
{  
    "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 \(p. 243\)](#)

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 V2](#)
- [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 \(p. 240\)](#) 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
- CREATE FAILED
- CREATE STOPPING
- CREATE STOPPED

To get the status of the version, call [DescribeSolutionVersion \(p. 289\)](#). 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 \(p. 322\)](#)
- [DescribeSolutionVersion \(p. 289\)](#)
- [ListSolutions \(p. 319\)](#)
- [CreateSolution \(p. 240\)](#)
- [DescribeSolution \(p. 286\)](#)
- [DeleteSolution \(p. 260\)](#)

Request Syntax

```
{  
    "solutionArn": "string",  
    "trainingMode": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[solutionArn \(p. 245\)](#)

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 \(p. 245\)](#)

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 or the [HRNN-Coldstart](#) recipe.

Type: String

Valid Values: `FULL` | `UPDATE`

Required: No

Response Syntax

```
{  
    "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 \(p. 246\)](#)

The ARN of the new solution version.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:+`

Errors

InvalidArgumentException

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 V2](#)
- [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. 218).

Request Syntax

```
{  
    "campaignArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

campaignArn (p. 248)

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 V2](#)
- [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 \(p. 221\)](#).

Request Syntax

```
{  
    "datasetArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetArn (p. 250)

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 V2](#)
- [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

```
{  
    "datasetGroupArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn (p. 252)

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 V2](#)
- [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 \(p. 233\)](#).

Request Syntax

```
{  
    "eventTrackerArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

eventTrackerArn (p. 254)

The Amazon Resource Name (ARN) of the event tracker 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 V2](#)
- [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

```
{  
    "filterArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

filterArn (p. 256)

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 V2](#)
- [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 \(p. 238\)](#).

Request Syntax

```
{  
    "schemaArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

schemaArn (p. 258)

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 V2](#)
- [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 \(p. 297\)](#) 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 \(p. 240\)](#).

Request Syntax

```
{  
    "solutionArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

solutionArn (p. 260)

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 V2](#)
- [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

```
{  
    "algorithmArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

algorithmArn (p. 262)

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

```
{  
    "algorithm": {  
        "algorithmArn": "string",  
        "algorithmImage": {  
            "dockerURI": "string",  
            "name": "string"  
        },  
        "creationDateTime": number,  
        "defaultHyperParameterRanges": {  
            "categoricalHyperParameterRanges": [  
                {  
                    "isTunable": boolean,  
                    "name": "string",  
                    "values": [ "string" ]  
                }  
            ],  
            "continuousHyperParameterRanges": [  
                {  
                    "isTunable": boolean,  
                    "maxValue": number,  
                    "minValue": number,  
                    "name": "string"  
                }  
            ],  
            "integerHyperParameterRanges": [  
                {  
                    "isTunable": boolean,  
                    "maxValue": number,  
                    "minValue": number,  
                    "name": "string"  
                }  
            ]  
        }  
    }  
}
```

```
        ],
},
"defaultHyperParameters": {
    "string" : "string"
},
"defaultResourceConfig": {
    "string" : "string"
},
"lastUpdatedDateTime": number,
"name": "string",
"roleArn": "string",
"trainingInputMode": "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.

algorithm (p. 262)

A listing of the properties of the algorithm.

Type: [Algorithm \(p. 347\)](#) object

Errors

InvalidArgumentException

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 V2](#)
- [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. 264)

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. 264)

Information on the specified batch inference job.

Type: [BatchInferenceJob](#) (p. 352) 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 V2](#)
- [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 \(p. 218\)](#).

Request Syntax

```
{  
    "campaignArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

campaignArn (p. 266)

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

```
{  
    "campaign": {  
        "campaignArn": "string",  
        "campaignConfig": {  
            "itemExplorationConfig": {  
                "string" : "string"  
            }  
        },  
        "creationDateTime": number,  
        "failureReason": "string",  
        "lastUpdatedDateTime": number,  
        "latestCampaignUpdate": {  
            "campaignConfig": {  
                "itemExplorationConfig": {  
                    "string" : "string"  
                }  
            },  
            "creationDateTime": number,  
            "failureReason": "string",  
            "lastUpdatedDateTime": number,  
            "status": "string",  
            "statusReason": "string"  
        }  
    }  
}
```

```
        "lastUpdatedDateTime": number,
        "minProvisionedTPS": number,
        "solutionVersionArn": "string",
        "status": "string"
    },
    "minProvisionedTPS": number,
    "name": "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.

[campaign \(p. 266\)](#)

The properties of the campaign.

Type: [Campaign \(p. 360\)](#) object

Errors

InvalidArgumentException

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 V2](#)
- [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 \(p. 221\)](#).

Request Syntax

```
{  
    "datasetArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[datasetArn \(p. 268\)](#)

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

```
{  
    "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 \(p. 268\)](#)

A listing of the dataset's properties.

Type: [Dataset \(p. 369\)](#) 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 V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

DescribeDatasetExportJob

Service: Amazon Personalize

Describes the dataset export job created by [CreateDatasetExportJob \(p. 224\)](#), including the export job status.

Request Syntax

```
{  
    "datasetExportJobArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[datasetExportJobArn \(p. 270\)](#)

The Amazon Resource Name (ARN) of the dataset export job to describe.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.*:.*:+

Required: Yes

Response Syntax

```
{  
    "datasetExportJob": {  
        "creationDateTime": number,  
        "datasetArn": "string",  
        "datasetExportJobArn": "string",  
        "failureReason": "string",  
        "ingestionMode": "string",  
        "jobName": "string",  
        "jobOutput": {  
            "s3DataDestination": {  
                "kmsKeyArn": "string",  
                "path": "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.

[datasetExportJob \(p. 270\)](#)

Information about the dataset export job, including the status.

The status is one of the following values:

- CREATE PENDING
- CREATE IN_PROGRESS
- ACTIVE
- CREATE FAILED

Type: [DatasetExportJob \(p. 371\)](#) object

Errors

InvalidArgumentException

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 V2](#)
- [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 \(p. 227\)](#).

Request Syntax

```
{  
    "datasetGroupArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn (p. 272)

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

```
{  
    "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. 272)

A listing of the dataset group's properties.

Type: [DatasetGroup \(p. 377\)](#) object

Errors

InvalidArgumentException

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 V2](#)
- [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 \(p. 230\)](#), including the import job status.

Request Syntax

```
{  
    "datasetImportJobArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[datasetImportJobArn \(p. 274\)](#)

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

```
{  
    "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 \(p. 274\)](#)

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 \(p. 381\)](#) object

Errors

InvalidArgumentException

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 V2](#)
- [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 \(p. 233\)](#).

Request Syntax

```
{  
    "eventTrackerArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[eventTrackerArn \(p. 276\)](#)

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 \(p. 276\)](#)

An object that describes the event tracker.

Type: [EventTracker \(p. 395\)](#) object

Errors

InvalidArgumentException

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 V2](#)
- [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 \(p. 278\)](#)

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 \(p. 278\)](#)

A listing of the FeatureTransformation properties.

Type: [FeatureTransformation \(p. 399\)](#) 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 V2](#)
- [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 \(p. 280\)](#)

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 \(p. 280\)](#)

The filter's details.

Type: [Filter \(p. 401\)](#) 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 V2](#)
- [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 \(p. 240\)](#) 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

```
{  
    "recipeArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[recipeArn \(p. 282\)](#)

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

```
{  
    "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 \(p. 282\)](#)

An object that describes the recipe.

Type: [Recipe \(p. 411\)](#) object

Errors

InvalidArgumentException

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 V2](#)
- [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 \(p. 238\)](#).

Request Syntax

```
{  
    "schemaArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[schemaArn \(p. 284\)](#)

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

```
{  
    "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 \(p. 284\)](#)

The requested schema.

Type: [DatasetSchema \(p. 385\)](#) 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 V2](#)
- [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 \(p. 240\)](#).

Request Syntax

```
{  
    "solutionArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[solutionArn \(p. 286\)](#)

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

```
{  
    "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"  
            }  
        }  
    }  
}
```

```
        "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"
        }
    },
    "optimizationObjective": {
        "itemAttribute": "string",
        "objectiveSensitivity": "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 \(p. 286\)](#)

An object that describes the solution.

Type: [Solution \(p. 416\)](#) 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 V2](#)
- [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 \(p. 240\)](#).

Request Syntax

```
{  
    "solutionVersionArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[solutionVersionArn \(p. 289\)](#)

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

```
{  
    "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"
    }
},
"optimizationObjective": {
    "itemAttribute": "string",
    "objectiveSensitivity": "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. 289](#))

The solution version.

Type: [SolutionVersion](#) ([p. 423](#)) 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 V2](#)
- [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 \(p. 292\)](#)

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 \(p. 292\)](#)

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 \(p. 292\)](#)

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

InvalidArgumentException

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 V2](#)
- [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

```
{  
    "maxResults": number,  
    "nextToken": "string",  
    "solutionVersionArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[maxResults \(p. 294\)](#)

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 \(p. 294\)](#)

The token to request the next page of results.

Type: String

Length Constraints: Maximum length of 1300.

Required: No

[solutionVersionArn \(p. 294\)](#)

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

```
{  
    "batchInferenceJobs": [  
        {  
            "batchInferenceJobArn": "string",  
            "creationDateTime": number,  
            "failureReason": "string",  
            "jobName": "string",  
            "lastUpdatedDateTime": number,  
            "status": "string",  
            "statusDetail": "string"  
        }  
    ]  
}
```

```
        "solutionVersionArn": "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.

[batchInferenceJobs \(p. 294\)](#)

A list containing information on each job that is returned.

Type: Array of [BatchInferenceJobSummary \(p. 358\)](#) objects

Array Members: Maximum number of 100 items.

[nextToken \(p. 294\)](#)

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

InvalidArgumentException

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 V2](#)
- [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 \(p. 218\)](#).

Request Syntax

```
{  
    "maxResults": number,  
    "nextToken": "string",  
    "solutionArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[maxResults \(p. 297\)](#)

The maximum number of campaigns to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

[nextToken \(p. 297\)](#)

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 \(p. 297\)](#)

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",  
            "status": "string"  
        }  
    ]  
}
```

```
        "creationDateTime": number,  
        "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.

[campaigns \(p. 297\)](#)

A list of the campaigns.

Type: Array of [CampaignSummary \(p. 363\)](#) objects

Array Members: Maximum number of 100 items.

[nextToken \(p. 297\)](#)

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 V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)

- [AWS SDK for Ruby V3](#)

ListDatasetExportJobs

Service: Amazon Personalize

Returns a list of dataset export jobs that use the given dataset. When a dataset is not specified, all the dataset export jobs associated with the account are listed. The response provides the properties for each dataset export job, including the Amazon Resource Name (ARN). For more information on dataset export jobs, see [CreateDatasetExportJob](#) (p. 224). For more information on datasets, see [CreateDataset](#) (p. 221).

Request Syntax

```
{  
  "datasetArn": "string",  
  "maxResults": number,  
  "nextToken": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetArn (p. 300)

The Amazon Resource Name (ARN) of the dataset to list the dataset export jobs for.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-z\d-]+):personalize:.*:.*:+

Required: No

maxResults (p. 300)

The maximum number of dataset export jobs to return.

Type: Integer

Valid Range: Minimum value of 1 Maximum value of 100

Required: No

nextToken (n. 300)

A token returned from the previous call to `ListDatasetExportJobs` for getting the next set of dataset export jobs (if they exist).

Type: String

Length Constraints: Maximum length of 1300

Required: No

Response Syntax

```
{  
  "datasetExportJobs": [  
    {  
      "creationDateTime": number,  
      "datasetId": string,  
      "status": string  
    }  
  ]  
}
```

```
        "datasetExportJobArn": "string",
        "failureReason": "string",
        "jobName": "string",
        "lastUpdatedDateTime": number,
        "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.

[datasetExportJobs \(p. 300\)](#)

The list of dataset export jobs.

Type: Array of [DatasetExportJobSummary \(p. 375\)](#) objects

Array Members: Maximum number of 100 items.

[nextToken \(p. 300\)](#)

A token for getting the next set of dataset export jobs (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Errors

InvalidArgumentException

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 V2](#)
- [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 \(p. 227\)](#).

Request Syntax

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

Request Parameters

The request accepts the following data in JSON format.

maxResults ([p. 303](#))

The maximum number of dataset groups to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken ([p. 303](#))

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

```
{  
    "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. 303)

The list of your dataset groups.

Type: Array of [DatasetGroupSummary \(p. 379\)](#) objects

Array Members: Maximum number of 100 items.

nextToken (p. 303)

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 V2](#)
- [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. 230). For more information on datasets, see [CreateDataset](#) (p. 221).

Request Syntax

```
{  
  "datasetArn": "string",  
  "maxResults": number,  
  "nextToken": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetArn (p. 305)

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. 305)

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. 305)

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

```
{  
  "datasetImportJobs": [  
    {  
      "creationDateTime": number  
    }  
  ]  
}
```

```
    "datasetImportJobArn": "string",
    "failureReason": "string",
    "jobName": "string",
    "lastUpdatedDateTime": number,
    "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.

[datasetImportJobs \(p. 305\)](#)

The list of dataset import jobs.

Type: Array of [DatasetImportJobSummary \(p. 383\)](#) objects

Array Members: Maximum number of 100 items.

[nextToken \(p. 305\)](#)

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 V2](#)
- [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 \(p. 221\)](#).

Request Syntax

```
{  
    "datasetGroupArn": "string",  
    "maxResults": number,  
    "nextToken": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[datasetGroupArn \(p. 308\)](#)

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 \(p. 308\)](#)

The maximum number of datasets to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

[nextToken \(p. 308\)](#)

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",  
            "lastModifiedDate": number,  
            "status": "string",  
            "statusDetail": "string",  
            "version": "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.

[datasets \(p. 308\)](#)

An array of `Dataset` objects. Each object provides metadata information.

Type: Array of [DatasetSummary \(p. 388\)](#) objects

Array Members: Maximum number of 100 items.

[nextToken \(p. 308\)](#)

A token for getting the next set of datasets (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Errors

InvalidArgumentException

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 V2](#)
- [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 \(p. 233\)](#).

Request Syntax

```
{  
    "datasetGroupArn": "string",  
    "maxResults": number,  
    "nextToken": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[datasetGroupArn \(p. 311\)](#)

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 \(p. 311\)](#)

The maximum number of event trackers to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

[nextToken \(p. 311\)](#)

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,  
            "status": "string"  
        }  
    ]  
}
```

```
        "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.

[eventTrackers \(p. 311\)](#)

A list of event trackers.

Type: Array of [EventTrackerSummary \(p. 397\)](#) objects

Array Members: Maximum number of 100 items.

[nextToken \(p. 311\)](#)

A token for getting the next set of event trackers (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Errors

InvalidArgumentException

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 V2](#)
- [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 \(p. 313\)](#)

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 \(p. 313\)](#)

The maximum number of filters to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

[nextToken \(p. 313\)](#)

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,  
            "status": "string"  
        }  
    ]  
}
```

```
        "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.

[Filters \(p. 313\)](#)

A list of returned filters.

Type: Array of [FilterSummary \(p. 403\)](#) objects

Array Members: Maximum number of 100 items.

[nextToken \(p. 313\)](#)

A token for getting the next set of filters (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

Errors

InvalidArgumentException

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 V2](#)
- [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

```
{  
    "maxResults": number,  
    "nextToken": "string",  
    "recipeProvider": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[maxResults \(p. 315\)](#)

The maximum number of recipes to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

[nextToken \(p. 315\)](#)

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 \(p. 315\)](#)

The default is `SERVICE`.

Type: String

Valid Values: `SERVICE`

Required: No

Response Syntax

```
{  
    "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. 315)

A token for getting the next set of recipes.

Type: String

Length Constraints: Maximum length of 1300.

[recipes](#) (p. 315)

The list of available recipes.

Type: Array of [RecipeSummary](#) (p. 413) 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 V2](#)
- [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 \(p. 238\)](#).

Request Syntax

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

Request Parameters

The request accepts the following data in JSON format.

[maxResults \(p. 317\)](#)

The maximum number of schemas to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

[nextToken \(p. 317\)](#)

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

```
{  
    "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 \(p. 317\)](#)

A token used to get the next set of schemas (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

[schemas \(p. 317\)](#)

A list of schemas.

Type: Array of [DatasetSchemaSummary \(p. 387\)](#) 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 V2](#)
- [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 \(p. 240\)](#).

Request Syntax

```
{  
    "datasetGroupArn": "string",  
    "maxResults": number,  
    "nextToken": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[datasetGroupArn \(p. 319\)](#)

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 \(p. 319\)](#)

The maximum number of solutions to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

[nextToken \(p. 319\)](#)

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

```
{  
    "nextToken": "string",  
    "solutions": [  
        {
```

```
        "creationDateTime": number,  
        "lastUpdatedDateTime": number,  
        "name": string,  
        "solutionArn": 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. 319)

A token for getting the next set of solutions (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

[solutions](#) (p. 319)

A list of the current solutions.

Type: Array of [SolutionSummary](#) (p. 421) 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 V2](#)
- [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 \(p. 240\)](#).

Request Syntax

```
{  
    "maxResults": number,  
    "nextToken": "string",  
    "solutionArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[maxResults \(p. 322\)](#)

The maximum number of solution versions to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

[nextToken \(p. 322\)](#)

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 \(p. 322\)](#)

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

```
{  
    "nextToken": "string",  
    "solutionVersions": [  
        {  
            "creationDateTime": number,  
            "lastUpdatedTime": number,  
            "solutionArn": "string",  
            "version": "string"  
        }  
    ]  
}
```

```
        "failureReason": "string",
        "lastUpdatedDateTime": number,
        "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.

nextToken ([p. 322](#))

A token for getting the next set of solution versions (if they exist).

Type: String

Length Constraints: Maximum length of 1300.

solutionVersions ([p. 322](#))

A list of solution versions describing the version properties.

Type: Array of [SolutionVersionSummary](#) ([p. 426](#)) 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 V2](#)
- [AWS SDK for JavaScript](#)

- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

StopSolutionVersionCreation

Service: Amazon Personalize

Stops creating a solution version that is in a state of CREATE_PENDING or CREATE_IN_PROGRESS.

Depending on the current state of the solution version, the solution version state changes as follows:

- CREATE_PENDING > CREATE_STOPPED
 - or
- CREATE_IN_PROGRESS > CREATE_STOPPING > CREATE_STOPPED

You are billed for all of the training completed up until you stop the solution version creation. You cannot resume creating a solution version once it has been stopped.

Request Syntax

```
{  
    "solutionVersionArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

solutionVersionArn (p. 325)

The Amazon Resource Name (ARN) of the solution version you want to stop creating.

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 V2](#)
- [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 \(p. 266\)](#) 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 \(p. 218\)](#).

Request Syntax

```
{  
    "campaignArn": "string",  
    "campaignConfig": {  
        "itemExplorationConfig": {  
            "string" : "string"  
        }  
    },  
    "minProvisionedTPS": number,  
    "solutionVersionArn": "string"  
}
```

Request Parameters

The request accepts the following data in JSON format.

[campaignArn \(p. 327\)](#)

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 \(p. 327\)](#)

The configuration details of a campaign.

Type: [CampaignConfig \(p. 362\)](#) object

Required: No

[minProvisionedTPS \(p. 327\)](#)

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 \(p. 327\)](#)

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 \(p. 328\)](#)

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](#)

- [AWS SDK for .NET](#)
- [AWS SDK for C++](#)
- [AWS SDK for Go](#)
- [AWS SDK for Java V2](#)
- [AWS SDK for JavaScript](#)
- [AWS SDK for PHP V3](#)
- [AWS SDK for Python](#)
- [AWS SDK for Ruby V3](#)

Amazon Personalize Events

The following actions are supported by Amazon Personalize Events:

- [PutEvents \(p. 330\)](#)
- [PutItems \(p. 332\)](#)
- [PutUsers \(p. 334\)](#)

PutEvents

Service: Amazon Personalize Events

Records user interaction event data. For more information see [Recording Events](#).

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 \(p. 330\)](#)

A list of event data from the session.

Type: Array of [Event \(p. 429\)](#) objects

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Required: Yes

[sessionId \(p. 330\)](#)

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 [Recording Events](#).

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: Yes

[trackingId \(p. 330\)](#)

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 \(p. 330\)](#)

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

InvalidArgumentException

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 V2](#)
- [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](#).

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 \(p. 332\)](#)

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 \(p. 332\)](#)

A list of item data.

Type: Array of [Item \(p. 431\)](#) 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

InvalidArgumentException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

ResourceInUseException

The specified resource is in use.

HTTP Status Code: 409

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 V2](#)
- [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](#).

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 \(p. 334\)](#)

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 \(p. 334\)](#)

A list of user data.

Type: Array of [User \(p. 432\)](#) 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

InvalidArgumentException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

ResourceInUseException

The specified resource is in use.

HTTP Status Code: 409

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 V2](#)
- [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 \(p. 336\)](#)
- [GetRecommendations \(p. 340\)](#)

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 \(p. 336\)](#)

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 \(p. 336\)](#)

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 \(p. 336\)](#)

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 \(p. 336\)](#)

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 \(p. 336\)](#)

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 \(p. 336\)](#)

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

```
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 \(p. 338\)](#)

A list of items in order of most likely interest to the user. The maximum is 500.

Type: Array of [PredictedItem \(p. 433\)](#) objects

[recommendationId \(p. 338\)](#)

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 V2](#)
- [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:

- USER_PERSONALIZATION - `userId` required, `itemId` not used
- RELATED_ITEMS - `itemId` required, `userId` not used

Note

Campaigns that are backed by a solution created using a recipe of type PERSONALIZED_RANKING use the [GetPersonalizedRanking \(p. 336\)](#) API.

Request Syntax

```
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 \(p. 340\)](#)

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 \(p. 340\)](#)

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 \(p. 340\)](#)

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:(\w+\d-+):personalize::.*::.+

Required: No

[filterValues \(p. 340\)](#)

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 \(p. 340\)](#)

The item ID to provide recommendations for.

Required for RELATED_ITEMS recipe type.

Type: String

Length Constraints: Maximum length of 256.

Required: No

[numResults \(p. 340\)](#)

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 \(p. 340\)](#)

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 \(p. 342\)](#)

A list of recommendations sorted in descending order by prediction score. There can be a maximum of 500 items in the list.

Type: Array of [PredictedItem \(p. 433\)](#) objects

[recommendationId \(p. 342\)](#)

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 V2](#)
- [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 \(p. 347\)](#)
- [AlgorithmlImage \(p. 349\)](#)
- [AutoMLConfig \(p. 350\)](#)
- [AutoMLResult \(p. 351\)](#)
- [BatchInferenceJob \(p. 352\)](#)
- [BatchInferenceJobConfig \(p. 355\)](#)
- [BatchInferenceJobInput \(p. 356\)](#)
- [BatchInferenceJobOutput \(p. 357\)](#)
- [BatchInferenceJobSummary \(p. 358\)](#)
- [Campaign \(p. 360\)](#)
- [CampaignConfig \(p. 362\)](#)
- [CampaignSummary \(p. 363\)](#)
- [CampaignUpdateSummary \(p. 365\)](#)
- [CategoricalHyperParameterRange \(p. 367\)](#)
- [ContinuousHyperParameterRange \(p. 368\)](#)
- [Dataset \(p. 369\)](#)
- [DatasetExportJob \(p. 371\)](#)
- [DatasetExportJobOutput \(p. 374\)](#)
- [DatasetExportJobSummary \(p. 375\)](#)
- [DatasetGroup \(p. 377\)](#)
- [DatasetGroupSummary \(p. 379\)](#)
- [DatasetImportJob \(p. 381\)](#)
- [DatasetImportJobSummary \(p. 383\)](#)

- [DatasetSchema \(p. 385\)](#)
- [DatasetSchemaSummary \(p. 387\)](#)
- [DatasetSummary \(p. 388\)](#)
- [DataSource \(p. 390\)](#)
- [DefaultCategoricalHyperParameterRange \(p. 391\)](#)
- [DefaultContinuousHyperParameterRange \(p. 392\)](#)
- [DefaultHyperParameterRanges \(p. 393\)](#)
- [DefaultIntegerHyperParameterRange \(p. 394\)](#)
- [EventTracker \(p. 395\)](#)
- [EventTrackerSummary \(p. 397\)](#)
- [FeatureTransformation \(p. 399\)](#)
- [Filter \(p. 401\)](#)
- [FilterSummary \(p. 403\)](#)
- [HPOConfig \(p. 405\)](#)
- [HPOObjective \(p. 406\)](#)
- [HPOResourceConfig \(p. 407\)](#)
- [HyperParameterRanges \(p. 408\)](#)
- [IntegerHyperParameterRange \(p. 409\)](#)
- [OptimizationObjective \(p. 410\)](#)
- [Recipe \(p. 411\)](#)
- [RecipeSummary \(p. 413\)](#)
- [S3DataConfig \(p. 415\)](#)
- [Solution \(p. 416\)](#)
- [SolutionConfig \(p. 419\)](#)
- [SolutionSummary \(p. 421\)](#)
- [SolutionVersion \(p. 423\)](#)
- [SolutionVersionSummary \(p. 426\)](#)
- [TunedHPOParams \(p. 428\)](#)

The following data types are supported by Amazon Personalize Events:

- [Event \(p. 429\)](#)
- [Item \(p. 431\)](#)
- [User \(p. 432\)](#)

The following data types are supported by Amazon Personalize Runtime:

- [PredictedItem \(p. 433\)](#)

Amazon Personalize

The following data types are supported by Amazon Personalize:

- [Algorithm \(p. 347\)](#)
- [AlgorithmImage \(p. 349\)](#)
- [AutoMLConfig \(p. 350\)](#)
- [AutoMLResult \(p. 351\)](#)

- [BatchInferenceJob \(p. 352\)](#)
- [BatchInferenceJobConfig \(p. 355\)](#)
- [BatchInferenceJobInput \(p. 356\)](#)
- [BatchInferenceJobOutput \(p. 357\)](#)
- [BatchInferenceJobSummary \(p. 358\)](#)
- [Campaign \(p. 360\)](#)
- [CampaignConfig \(p. 362\)](#)
- [CampaignSummary \(p. 363\)](#)
- [CampaignUpdateSummary \(p. 365\)](#)
- [CategoricalHyperParameterRange \(p. 367\)](#)
- [ContinuousHyperParameterRange \(p. 368\)](#)
- [Dataset \(p. 369\)](#)
- [DatasetExportJob \(p. 371\)](#)
- [DatasetExportJobOutput \(p. 374\)](#)
- [DatasetExportJobSummary \(p. 375\)](#)
- [DatasetGroup \(p. 377\)](#)
- [DatasetGroupSummary \(p. 379\)](#)
- [DatasetImportJob \(p. 381\)](#)
- [DatasetImportJobSummary \(p. 383\)](#)
- [DatasetSchema \(p. 385\)](#)
- [DatasetSchemaSummary \(p. 387\)](#)
- [DatasetSummary \(p. 388\)](#)
- [DataSource \(p. 390\)](#)
- [DefaultCategoricalHyperParameterRange \(p. 391\)](#)
- [DefaultContinuousHyperParameterRange \(p. 392\)](#)
- [DefaultHyperParameterRanges \(p. 393\)](#)
- [DefaultIntegerHyperParameterRange \(p. 394\)](#)
- [EventTracker \(p. 395\)](#)
- [EventTrackerSummary \(p. 397\)](#)
- [FeatureTransformation \(p. 399\)](#)
- [Filter \(p. 401\)](#)
- [FilterSummary \(p. 403\)](#)
- [HPOConfig \(p. 405\)](#)
- [HPOObjective \(p. 406\)](#)
- [HPOResourceConfig \(p. 407\)](#)
- [HyperParameterRanges \(p. 408\)](#)
- [IntegerHyperParameterRange \(p. 409\)](#)
- [OptimizationObjective \(p. 410\)](#)
- [Recipe \(p. 411\)](#)
- [RecipeSummary \(p. 413\)](#)
- [S3DataConfig \(p. 415\)](#)
- [Solution \(p. 416\)](#)
- [SolutionConfig \(p. 419\)](#)
- [SolutionSummary \(p. 421\)](#)
- [SolutionVersion \(p. 423\)](#)
- [SolutionVersionSummary \(p. 426\)](#)

- [TunedHPOParams \(p. 428\)](#)

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 \(p. 349\)](#) 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 \(p. 393\)](#) 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

Length Constraints: Minimum length of 1. Maximum length of 63.

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:(\w+\:){1}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 V2](#)
- [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

Length Constraints: Minimum length of 1. Maximum length of 63.

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 V2](#)
- [AWS SDK for Ruby V3](#)

AutoMLConfig

Service: Amazon Personalize

When the solution performs AutoML (`performAutoML` is true in [CreateSolution \(p. 240\)](#)), 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 V2](#)
- [AWS SDK for Ruby V3](#)

AutoMLResult

Service: Amazon Personalize

When the solution performs AutoML (`performAutoML` is true in [CreateSolution \(p. 240\)](#)), 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 V2](#)
- [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 \(p. 355\)](#) 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 \(p. 356\)](#) object

Required: No

jobName

The name of the batch inference job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

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 \(p. 357\)](#) 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-zA-Z\d-]+):iam:::\d{12}:role/?[a-zA-Z_0-9+=,.@\-\/_]+

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-zA-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 V2](#)
- [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](#).

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 V2](#)
- [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 \(p. 415\)](#) 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 V2](#)
- [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 \(p. 415\)](#) 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 V2](#)
- [AWS SDK for Ruby V3](#)

BatchInferenceJobSummary

Service: Amazon Personalize

A truncated version of the [BatchInferenceJob \(p. 352\)](#) datatype. The [ListBatchInferenceJobs \(p. 294\)](#) 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-zA-Z-]+):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

Length Constraints: Minimum length of 1. Maximum length of 63.

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-zA-Z-]+):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 V2](#)
- [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 \(p. 218\)](#).

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 \(p. 362\)](#) 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 \(p. 266\)](#) API.

Type: [CampaignUpdateSummary \(p. 365\)](#) 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

Length Constraints: Minimum length of 1. Maximum length of 63.

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-zA-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 V2](#)
- [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. Provide `itemExplorationConfig` data only if your solution uses the [User-Personalization](#) recipe.

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 V2](#)
- [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. 266) API.

Contents

campaignArn

The Amazon Resource Name (ARN) of the campaign.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-zA-Z-]+):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

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: `^[\w\-_]{1,63}`

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 V2](#)
- [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 \(p. 266\)](#) API.

Contents

campaignConfig

The configuration details of a campaign.

Type: [CampaignConfig \(p. 362\)](#) 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:(\w+\:){1}personalize:(\w+\:){1}\w+`

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 V2](#)
- [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 V2](#)
- [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 V2](#)
- [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

Length Constraints: Minimum length of 1. Maximum length of 63.

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-zA-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 V2](#)
- [AWS SDK for Ruby V3](#)

DatasetExportJob

Service: Amazon Personalize

Describes a job that exports a dataset to an Amazon S3 bucket. For more information, see [CreateDatasetExportJob \(p. 224\)](#).

A dataset export 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 export job.

Type: Timestamp

Required: No

datasetArn

The Amazon Resource Name (ARN) of the dataset to export.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.*`

Required: No

datasetExportJobArn

The Amazon Resource Name (ARN) of the dataset export job.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.*`

Required: No

failureReason

If a dataset export job fails, provides the reason why.

Type: String

Required: No

ingestionMode

The data to export, based on how you imported the data. You can choose to export `BULK` data that you imported using a dataset import job, `PUT` data that you imported incrementally (using the `console`, `PutEvents`, `PutUsers` and `PutItems` operations), or `ALL` for both types. The default value is `PUT`.

Type: String

Valid Values: `BULK` | `PUT` | `ALL`

Required: No

jobName

The name of the export job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

Required: No

jobOutput

The path to the Amazon S3 bucket where the job's output is stored. For example:

s3://bucket-name/folder-name/

Type: [DatasetExportJobOutput \(p. 374\)](#) object

Required: No

lastUpdatedDateTime

The date and time (in Unix time) the status of the dataset export job was last updated.

Type: Timestamp

Required: No

roleArn

The Amazon Resource Name (ARN) of the IAM service role that has permissions to add data to your output Amazon S3 bucket.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-zA-Z\d-]+):personalize:.*:.*:.*

Required: No

status

The status of the dataset export job.

A dataset export 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 V2](#)
- [AWS SDK for Ruby V3](#)

DatasetExportJobOutput

Service: Amazon Personalize

The output configuration parameters of a dataset export job.

Contents

s3DataDestination

The configuration details of an Amazon S3 input or output bucket.

Type: [S3DataConfig \(p. 415\)](#) 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 V2](#)
- [AWS SDK for Ruby V3](#)

DatasetExportJobSummary

Service: Amazon Personalize

Provides a summary of the properties of a dataset export job. For a complete listing, call the [DescribeDatasetExportJob \(p. 270\)](#) API.

Contents

creationDateTime

The date and time (in Unix time) that the dataset export job was created.

Type: Timestamp

Required: No

datasetExportJobArn

The Amazon Resource Name (ARN) of the dataset export job.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-zA-Z-]+):personalize:.*:.*:+`

Required: No

failureReason

If a dataset export job fails, the reason behind the failure.

Type: String

Required: No

jobName

The name of the dataset export job.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: `^[[a-zA-Z0-9][a-zA-Z0-9\-_]*`

Required: No

lastUpdatedDateTime

The date and time (in Unix time) that the dataset export job status was last updated.

Type: Timestamp

Required: No

status

The status of the dataset export job.

A dataset export 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 V2](#)
- [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 \(p. 227\)](#). You then create a dataset and add it to a dataset group by calling [CreateDataset \(p. 221\)](#). The dataset group is used to create and train a solution by calling [CreateSolution \(p. 240\)](#). 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 AWS Key Management Service (KMS) key used to encrypt the datasets.

Type: String

Pattern: arn:aws.*:kms:.*:[0-9]{12}:key/.*

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

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

Required: No

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-zA-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 V2](#)
- [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 \(p. 272\)](#) 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-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

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

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 V2](#)
- [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 \(p. 230\)](#).

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 \(p. 390\)](#) 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

Length Constraints: Minimum length of 1. Maximum length of 63.

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 IAM role that has permissions to read from the Amazon S3 data source.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:([a-zA-Z\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 V2](#)
- [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 \(p. 274\)](#) 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-zA-Z-]+):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][a-zA-Z0-9\-_]*`

Required: No

lastUpdatedDateTime

The date and time (in Unix time) that the dataset import job status 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

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 V2](#)
- [AWS SDK for Ruby V3](#)

DatasetSchema

Service: Amazon Personalize

Describes the schema for a dataset. For more information on schemas, see [CreateSchema \(p. 238\)](#).

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

Length Constraints: Minimum length of 1. Maximum length of 63.

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 V2](#)
- [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 \(p. 284\)](#) 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

Length Constraints: Minimum length of 1. Maximum length of 63.

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 V2](#)
- [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. 268) 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-zA-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

Length Constraints: Minimum length of 1. Maximum length of 63.

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 V2](#)
- [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/folder-name/`

Type: String

Length Constraints: Maximum length of 256.

Pattern: (`s3|http|https`)://.+

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 V2](#)
- [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 V2](#)
- [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 V2](#)
- [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 \(p. 391\)](#) objects

Array Members: Maximum number of 100 items.

Required: No

continuousHyperParameterRanges

The continuous hyperparameters and their default ranges.

Type: Array of [DefaultContinuousHyperParameterRange \(p. 392\)](#) objects

Array Members: Maximum number of 100 items.

Required: No

integerHyperParameterRanges

The integer-valued hyperparameters and their default ranges.

Type: Array of [DefaultIntegerHyperParameterRange \(p. 394\)](#) 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 V2](#)
- [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 V2](#)
- [AWS SDK for Ruby V3](#)

EventTracker

Service: Amazon Personalize

Provides information about an event tracker.

Contents

accountId

The 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

Length Constraints: Minimum length of 1. Maximum length of 63.

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 V2](#)
- [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 \(p. 276\)](#) 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-zA-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

Length Constraints: Minimum length of 1. Maximum length of 63.

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 V2](#)
- [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-zA-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

Length Constraints: Minimum length of 1. Maximum length of 63.

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 V2](#)
- [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 \(p. 170\)](#).

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2500.

Required: No

lastUpdatedDateTime

The time at which the filter was last updated.

Type: Timestamp

Required: No

name

The name of the filter.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

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 V2](#)
- [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-zA-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-zA-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

Length Constraints: Minimum length of 1. Maximum length of 63.

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 V2](#)
- [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 \(p. 408\)](#) object

Required: No

hpoObjective

The metric to optimize during HPO.

Note

Amazon Personalize doesn't support configuring the `hpoObjective` at this time.

Type: [HPOObjective \(p. 406\)](#) object

Required: No

hpoResourceConfig

Describes the resource configuration for HPO.

Type: [HPOResourceConfig \(p. 407\)](#) 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 V2](#)
- [AWS SDK for Ruby V3](#)

HPOObjective

Service: Amazon Personalize

The metric to optimize during hyperparameter optimization (HPO).

Note

Amazon Personalize doesn't support configuring the `hpoObjective` at this time.

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 V2](#)
- [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 V2](#)
- [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 \(p. 367\)](#) objects

Array Members: Maximum number of 100 items.

Required: No

continuousHyperParameterRanges

The continuous hyperparameters and their ranges.

Type: Array of [ContinuousHyperParameterRange \(p. 368\)](#) objects

Array Members: Maximum number of 100 items.

Required: No

integerHyperParameterRanges

The integer-valued hyperparameters and their ranges.

Type: Array of [IntegerHyperParameterRange \(p. 409\)](#) 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 V2](#)
- [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 V2](#)
- [AWS SDK for Ruby V3](#)

OptimizationObjective

Service: Amazon Personalize

Describes the additional objective for the solution, such as maximizing streaming minutes or increasing revenue. For more information see [Optimizing a solution](#).

Contents

itemAttribute

The numerical metadata column in an Items dataset related to the optimization objective. For example, VIDEO_LENGTH (to maximize streaming minutes), or PRICE (to maximize revenue).

Type: String

Length Constraints: Minimum length of 1. Maximum length of 150.

Required: No

objectiveSensitivity

Specifies how Amazon Personalize balances the importance of your optimization objective versus relevance.

Type: String

Valid Values: LOW | MEDIUM | HIGH | OFF

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 V2](#)
- [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 \(p. 240\)](#) 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

Length Constraints: Minimum length of 1. Maximum length of 63.

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-zA-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 V2](#)
- [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 \(p. 282\)](#) 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

Length Constraints: Minimum length of 1. Maximum length of 63.

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-zA-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 V2](#)
- [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 AWS 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

Pattern: `arn:aws.*:kms:.*:[0-9]{12}:key/.*`

Required: No

path

The file path of the Amazon S3 bucket.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `(s3|http|https)://.+`

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 V2](#)
- [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 \(p. 351\)](#) 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 \(p. 426\)](#) object

Required: No

name

The name of the solution.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

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 \(p. 419\)](#) 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 V2](#)
- [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 \(p. 350\)](#) object containing a list of recipes to search when AutoML is performed.

Type: [AutoMLConfig \(p. 350\)](#) 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 \(p. 405\)](#) object

Required: No

optimizationObjective

Describes the additional objective for the solution, such as maximizing streaming minutes or increasing revenue. For more information see [Optimizing a solution](#).

Type: [OptimizationObjective \(p. 410\)](#) 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 V2](#)
- [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. 286) 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

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: ^[a-zA-Z0-9][a-zA-Z0-9\-_]*

Required: No

solutionArn

The Amazon Resource Name (ARN) of the solution.

Type: String

Length Constraints: Maximum length of 256.

Pattern: arn:(\w+\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 V2](#)
- [AWS SDK for Ruby V3](#)

SolutionVersion

Service: Amazon Personalize

An object that provides information about a specific version of a [Solution \(p. 416\)](#).

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 \(p. 419\)](#) 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
- CREATE STOPPING
- CREATE STOPPED

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 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 or the [HRNN-Coldstart](#) recipe.

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 \(p. 428\)](#) 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 V2](#)
- [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 \(p. 289\)](#) 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-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 -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 V2](#)
- [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 V2](#)
- [AWS SDK for Ruby V3](#)

Amazon Personalize Events

The following data types are supported by Amazon Personalize Events:

- [Event \(p. 429\)](#)
- [Item \(p. 431\)](#)
- [User \(p. 432\)](#)

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

Length Constraints: Minimum length of 1. Maximum length of 1024.

Required: No

recommendationId

The ID of the recommendation.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 40.

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 V2](#)
- [AWS SDK for Ruby V3](#)

Item

Service: Amazon Personalize Events

Represents item metadata added to an Items dataset using the `PutItems` API. For more information see [Importing Items Incrementally](#).

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 schema for the Items dataset. In the previous example, the `numberOfRatings` matches the 'NUMBER_OF_RATINGS' field defined in the Items schema. For categorical string data, to include multiple categories for a single item, separate each category with a pipe separator (|). For example, \ "Horror|Action\".

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

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 V2](#)
- [AWS SDK for Ruby V3](#)

User

Service: Amazon Personalize Events

Represents user metadata added to a Users dataset using the [PutUsers API](#). For more information see [Importing Users Incrementally](#).

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 schema for the Users dataset. In the previous example, the `numberOfVideosWatched` matches the 'NUMBER_OF_VIDEOS_WATCHED' field defined in the Users schema. For categorical string data, to include multiple categories for a single user, separate each category with a pipe separator (|). For example, \ "Member|Frequent shopper\ ".

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

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 V2](#)
- [AWS SDK for Ruby V3](#)

Amazon Personalize Runtime

The following data types are supported by Amazon Personalize Runtime:

- [PredictedItem \(p. 433\)](#)

PredictedItem

Service: Amazon Personalize Runtime

An object that identifies an item.

The [GetRecommendations \(p. 340\)](#) and [GetPersonalizedRanking \(p. 336\)](#) 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 [Recommendation scores \(p. 156\)](#).

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 V2](#)
- [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

ValidationException

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.

update-history-change	update-history-description	update-history-date
New feature (p. 438)	Amazon Personalize now supports a new RELATED_ITEMS recommendation recipe Similar-Items. Use the Similar-Items recipe to generate recommendations for similar items based on both interactions data and item metadata. For more information, see Similar Items recipe .	October 5, 2021
New documentation feature (p. 438)	The Amazon Personalize developer guide now includes a getting started tutorial for using Amazon Personalize with the SDK for Java 2.x. For more information, see Getting started (SDK for Java 2.x) .	August 25, 2021
New feature (p. 438)	Amazon Personalize can now extract meaningful information from unstructured text metadata in an Items dataset. For more information, see Items dataset .	June 9, 2021
New feature (p. 438)	Amazon Personalize now supports the ability to stop creating a solution version (stop training a model). For more information, see Stopping the creation of a solution version .	May 20, 2021
New feature (preview release) (p. 438)	Amazon Personalize can now optimize a solution for an objective in addition to maximizing relevance, such as maximizing revenue. This feature is in preview release. For more information, see Optimizing a solution for an additional objective .	May 18, 2021
New feature (p. 438)	Amazon Personalize can now export the records in an Amazon Personalize dataset	April 26, 2021

	to an Amazon S3 bucket for analysis and tracking. For more information, see Exporting a dataset .	
New feature (p. 438)	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 .	November 17, 2020
New feature (p. 438)	Amazon Personalize can now filter recommendations based on criteria you specify when you get recommendations. For more information, see Filtering recommendations .	November 10, 2020
New feature (p. 438)	Amazon Personalize now supports the ability to incrementally import users and items. For more information, see Importing records incrementally .	October 2, 2020
New feature (p. 438)	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 .	August 5, 2020
New feature (p. 438)	Amazon Personalize can now filter recommendations based on item and user metadata using custom filter expressions. For more information, see Filtering recommendations .	July 31, 2020
New feature (p. 438)	Amazon Personalize now allows you to filter results based on which items a user has interacted with. For more information, see Filtering recommendations .	June 3, 2020

New feature (p. 438)	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 .	April 3, 2020
New Region (p. 438)	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 <i>Amazon Web Services General Reference</i> .	January 21, 2020
New feature (p. 438)	Amazon Personalize can now get recommendations based on contextual metadata. For more information, see Getting recommendations .	December 19, 2019
New regions (p. 438)	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 <i>Amazon Web Services General Reference</i> .	December 18, 2019
New feature (p. 438)	Amazon Personalize now supports batch recommendation workflows. For more information, see Get batch recommendations .	November 14, 2019
Amazon Personalize general availability (p. 438)	Amazon Personalize is now available for general use.	June 10, 2019
Amazon Personalize preview release (p. 438)	This is the first preview release of the documentation for Amazon Personalize.	November 28, 2018

AWS glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS General Reference*.