Developer Guide

Amazon Personalize

Amazon Personalize: Developer Guide

Table of Contents

What is Amazon Personalize?

Amazon Personalize is a fully managed machine learning service that uses your data to generate item recommendations for your users. It can also generate user segments based on the users' affinity for certain items or item metadata.

Common use case include the following:

- **Personalizing a video streaming app –** You can use preconfigured or customizable Amazon Personalize resources to add multiple types of personalized video recommendations to your streaming app. For example, *Top picks for you*, *More like X* and *Most popular* video recommendations.
- **Adding product recommendations to an ecommerce app –** You can use preconfigured or customizable Amazon Personalize resources to add multiple types of personalized product recommendations to your retail app. For example, *Recommended for you*, *Frequently bought together* and *Customers who viewed X also viewed* product recommendations.
- **Adding real-time next best action recommendations to your app –** You can use customizable Amazon Personalize resources to recommend the actions that your users will most likely take based on their behavior. For example, you can add real-time recommendations for enrolling in your loyalty program, downloading your mobile app, or signing up for promotional emails.
- **Creating personalized emails –** You can use customizable Amazon Personalize resources to generate batch recommendations for all users on an email list. Then you can use an AWS service or third party service to send users personalized emails recommending items in your catalog.
- **Creating a targeted marketing campaign –** You can use Amazon Personalize to generate segments of users who will most likely interact with items in your catalog. Then you can use an AWS service or third party service to create a targeted marketing campaign that promotes different items to different user segments.
- **Personalizing search results –** You can use customizable Amazon Personalize resources to personalize search results for your users. For example, Amazon Personalize can re-rank search results that you generate with OpenSearch.

For most use cases, Amazon Personalize generates recommendations primarily based on item interaction data. Item interaction data comes from your users interacting with items in your catalog. For example, users clicking different items. Your item interaction data can come from both your historical bulk interaction records in a CSV file, and real-time events from your users as they interact with your catalog. In some cases, Amazon Personalize also uses data from items and users such as genre, price, or gender. And for next best action scenarios, it uses actions and action interaction data.

When you import bulk data, you can use Amazon SageMaker Data Wrangler to import data from 40+ sources and prepare it for Amazon Personalize. For more information, see Preparing and importing data using Amazon SageMaker Data Wrangler.

Amazon Personalize includes API operations for real-time personalization, and batch operations for bulk recommendations and user segments. You can get started quickly with use-case optimized recommenders for your business domain, or you can create your own configurable custom resources.

Topics

Pricing for Amazon Personalize

With Amazon Personalize, there are no minimum fees and no upfront commitments. The AWS Free Tier provides a monthly quota of up to 20 GB of data processing per available AWS region, up to 100 hours of training time per eligible AWS region, and up to 180,000 recommendation requests. The free tier is valid for the first two months of usage.

For a complete list of charges and prices, see Amazon Personalize pricing.

Guidance for first-time Amazon Personalize users

If you're a first-time user of Amazon Personalize, the following resources can help you get started.

Topics

Discovering Amazon Personalize with the Magic Movie Machine

The Magic Movie Machine is an interactive learning experience. It helps you discover Amazon Personalize features and learn more about generating recommendations. For a short introduction, see the video below. Then try the Magic Movie Machine.

Getting Started with Amazon Personalize

Navigating this guide

We recommend you read the following sections in order:

1. **How it works** – 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. Start with this section to make sure you have good understanding of Amazon Personalize workflows and terms before you start getting recommendations.
2. **Setting up Amazon Personalize** – 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. **Getting started** – In this section you get started using Amazon Personalize with a simple movie dataset. Complete these tutorials to get hands-on experience with Amazon Personalize. You can choose to either get started with a Domain dataset group or a Custom dataset group:
   - To get started creating a Domain dataset group, complete the Getting started prerequisites and then start the tutorials in Getting started with a Domain dataset group.
   - To get started with a Custom dataset group, complete the Getting started prerequisites and then start the tutorials in Getting started with a Domain dataset group.
4. **Domain use cases and custom recipes** – Learn about the domain use cases and custom recipes you can use to train a model in Amazon Personalize. Use this information to help you match your use case to resources in Amazon Personalize.
5. **Readiness checklist** – Review the readiness checklist to start preparing to use Amazon Personalize with your own data. This checklist provides lists of Amazon Personalize features, requirements, and data guidance. It can help you plan or you can use it as a reference as you create resources in Amazon Personalize.
6. **Amazon Personalize workflow** – This section guides you through the complete Amazon Personalize workflow. It provides step-by-step instructions for creating a Domain dataset group

or a Custom dataset group, preparing and importing data, creating recommenders or custom resources, and getting recommendations.

7. **Recording events** – This section covers how to record item interaction and action interaction events in real time. After you have set up your Amazon Personalize resources, complete this section to learn how to keep your datasets up to date with your users' behavior.
8. **Filtering recommendations and user segments** – This section covers how to filter recommendations. Complete this section to learn how to construct filter expressions to filter recommendations 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.

Related AWS services and solutions

Amazon Personalize integrates seamlessly with other AWS services and solutions. For example, you can:

- Use Amazon SageMaker Data Wrangler (Data Wrangler) to import data from 40+ sources into an Amazon Personalize dataset. Data Wrangler is a feature of Amazon SageMaker Studio that provides an end-to-end solution to import, prepare, transform, and analyze data. For more information, see Preparing and importing data using Amazon SageMaker Data Wrangler.
- Use AWS Amplify to record item interaction events. Amplify includes a JavaScript library for recording events from web client applications. And it includes a library for recording events in server code. For more information, see Amplify Documentation.
- Automate and schedule Amazon Personalize tasks with Maintaining Personalized Experiences with Machine Learning. This AWS Solutions Implementation automates the Amazon Personalize workflow, including data import, solution version training, and batch workflows.
- Use Amazon CloudWatch Evidently to perform A/B testing with Amazon Personalize recommendations. For more information, see A/B testing with CloudWatch Evidently.
- Use Amazon Pinpoint to create targeted marketing campaigns. For an example that shows how to use Amazon Pinpoint and Amplify to add Amazon Personalize recommendations to a marketing email campaign and a web app, see Web Analytics with Amplify.

Third-party services

Amazon Personalize works well with various third-party services.

- **Amplitude** – You can use Amplitude to track user actions to help you understand your users' behavior. For information on using Amplitude and Amazon Personalize, see the following AWS Partner Network (APN) blog post: Measuring the Effectiveness of Personalization with Amplitude and Amazon Personalize.
- **Braze** – You can use Braze to send users personalized emails recommending items in your catalog. Braze is a market leading messaging platform (email, push, SMS). For a workshop that shows how to integrate Amazon Personalize and Braze, see Amazon Personalize workshop.
- **mParticle** – You can use mParticle to collect event data from your app. For an example that shows how to use mParticle and Amazon Personalize to implement personalized product recommendations, see How to harness the power of a CDP for machine learning: Part 2.
- **Optimizely** – You can use Optimizely to perform A/B testing with Amazon Personalize recommendations. For information on using Optimizely and Amazon Personalize, see Optimizely integrates with Amazon Personalize to combine powerful machine learning with experimentation.
- **Segment** – You can use Segment to send your data to Amazon Personalize. For more information on integrating Segment with Amazon Personalize, see Amazon Personalize Destination.

For a complete list of partners, see Amazon Personalize Partners.

Learn more

The following resources provide additional information about Amazon Personalize:

- For a quick reference to help you determine if Amazon Personalize fits your use case, see the Amazon Personalize Cheat Sheet in the Amazon Personalize samples repository.
- For a series of videos on how to use Amazon Personalize, see the Amazon Personalize Deep Dive Video Series found on YouTube.
- For in-depth tutorials and code samples, see the amazon-personalize-samples GitHub repository.

Amazon Personalize and generative AI

Amazon Personalize works well with generative artificial intelligence (generative AI). Amazon Personalize Content Generator, with the help of generative AI, can add engaging themes to batch recommendations for related items. *Content Generator* is a generative AI capability managed by Amazon Personalize.

You can also use Amazon Personalize recommendations to integrate Amazon Personalize with your generative AI workflow and enhance your users' experience. For example, you can add recommendations to generative AI prompts to create marketing content tailored to each of your user's interests. You can also generate concise summaries for recommended content, or recommend products or content through chat bots.

The following video shows how you can enhance recommendations with Amazon Personalize and generative AI.

Enhance Recommendations with Amazon Personalize and Generative AI

The following Amazon Personalize features use generative AI or can help you build generative AI solutions that create personalized content.

Topics

Recommendations with themes from Content Generator

Amazon Personalize Content Generator can add descriptive themes to batch recommendations. *Content Generator* is a generative AI capability managed by Amazon Personalize.

When you get batch recommendations with themes, Amazon Personalize Content Generator adds a descriptive theme for each set of similar items. For example, if you get similar items recommendations for a breakfast food item, Amazon Personalize might generate a theme like *Rise and shine* or *Morning essentials*. You might use the theme to replace a generic carousel title, like *Frequently bought together*. Or you might incorporate the theme in a promotional email or marketing campaign for new menu options.

To generate themes, you import data into Item interactions and Items datasets, create a custom solution with the Similar-Items recipe, and generate batch recommendations. Your item data must include item description and title information. Detailed item descriptions and titles help Content Generator create more accurate and engaging themes.

- For information about the Amazon Personalize workflow, see Amazon Personalize workflow.
- For information about batch recommendations, see Batch recommendations and user segments (custom resources).
- For information about generating recommendations with themes, see Batch recommendations with themes from Content Generator.

Recommendation metadata

When you get recommendations, you can have Amazon Personalize return metadata about each recommended item from your Items dataset. You can add this metadata, along with Amazon Personalize recommendations, to your generative AI prompts to generate more compelling content.

For example, you might use generative AI to create marketing emails. You can use Amazon Personalize recommendations and their metadata, such as movie genres, as part of prompt engineering for generative AI. With personalized prompts, you can use generative AI to create engaging marketing emails tailored to each of your customer's interests.

To get recommendation metadata, you first complete the Amazon Personalize workflow to import data and create domain or custom resources. When you create an Amazon Personalize *recommender* or a *campaign*, enable the option to include metadata in recommendations. When you get recommendations, you can specify which columns of item data you want to include.

- For information about the Amazon Personalize workflow, see Amazon Personalize workflow.
- For information about enabling metadata for a recommender, see Enabling metadata in recommendations (domain resources).
- For information about enabling metadata for a campaign, see Enabling metadata in recommendations (custom resources).
- For more information about how you can use Amazon Personalize with generative AI to create marketing campaigns, see Elevate your marketing solutions with Amazon Personalize and generative AI.

Pre-configured LangChain code for personalization

LangChain is a framework for developing applications powered by language models. It features code built for Amazon Personalize. You can use this code to integrate Amazon Personalize recommendations with your generative AI solution.

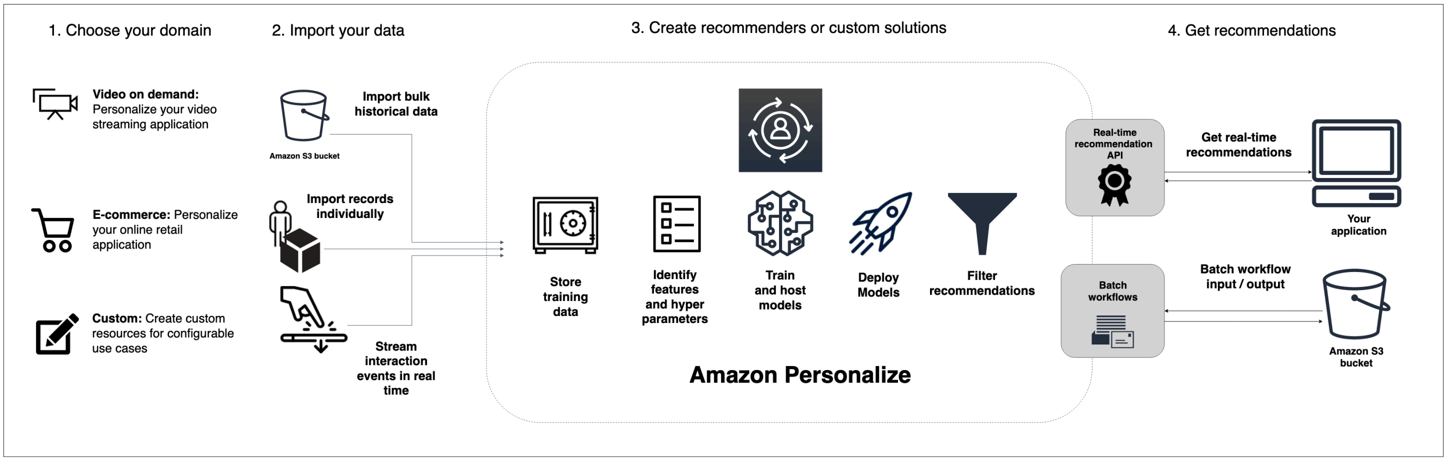
For example, you can use the following code to add Amazon Personalize recommendations for a user to your chain.

```
from aws_langchain import AmazonPersonalize
from aws_langchain import AmazonPersonalizeChain
from langchain.llms.bedrock import Bedrock

recommender_arn="RECOMMENDER ARN"

bedrock_llm = Bedrock(model_id="anthropic.claude-v2", region_name="us-west-2")
client=AmazonPersonalize(credentials_profile_name="default",region_name="us-
west-2",recommender_arn=recommender_arn)
# Create personalize chain
# Use return_direct=True if you do not want summary
chain = AmazonPersonalizeChain.from_llm(
    llm=bedrock_llm,
    client=client,
    return_direct=False
)
response = chain({'user_id': '1'})
print(response)
```

- For information about getting started with LangChain, see the Introduction in the LangChain documentation.
- For information about using LangChain code built for Amazon Personalize, including more advanced code samples, see Amazon Personalize LangChain extensions in the AWS samples repository.

How it works

Amazon Personalize uses your data to train domain-based or customizable recommendation models. You use a private recommendation API in your application to request real-time recommendations. Amazon Personalize also supports batch workflows get item recommendations and user segments.



Topics

- Amazon Personalize workflow summary
- Amazon Personalize terms
- Types of data Amazon Personalize can use

Amazon Personalize workflow summary

The Amazon Personalize workflow is as follows:

1. **Create a dataset group**

   A dataset group is a container for Amazon Personalize resources. The type of dataset group you create determines the resources you can create in step 3 of the Amazon Personalize workflow.

   - With a *Domain dataset group*, you can create recommenders for VIDEO_ON_DEMAND or ECOMMERCE domain use cases. Amazon Personalize manages the configuration, training and updating of these recommenders. If you start with a Domain dataset group, you can still add custom resources.

- With a *Custom dataset group*, you can create only custom resources. These including solutions, solution versions, and campaigns. For these resources, you have more control over configurations, updates, and retraining.

2. **Prepare and import data**

   You import interaction, item, user, action, and action interaction records into *datasets* (Amazon Personalize containers for data). You can import records in bulk or individually. When you import bulk data, you can use Amazon SageMaker Data Wrangler to import data from 40+ sources and prepare it for Amazon Personalize. For more information, see Preparing and importing data using Amazon SageMaker Data Wrangler.

   After you import data into an Amazon Personalize dataset, you can analyze it, export it to an Amazon S3 bucket, update it, or delete it by deleting the dataset. For more information, see Managing data.

3. **Create domain recommenders or custom resources**

   After you import your data, create domain recommenders (for Domain dataset groups) or custom resources (for Custom dataset group) to train a model on your data. You use these resources to generate recommendations.

4. **Get recommendations**

   Use your recommender or custom campaign to get recommendations. With a Custom dataset group, you can also get batch recommendations or user segments.

After you complete the Amazon Personalize workflow the first time, keep your data current, and regularly re-train any custom solutions. 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.

Amazon Personalize terms

This section introduces the terms used in Amazon Personalize.

Topics

- Model deployment and recommendations

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 resources, including datasets, domain recommenders, and custom resources. A dataset group organizes your resources into independent collections, where resources from one dataset group can't influence resources in any other dataset group. A dataset group can either be a Domain dataset group or a Custom dataset group.

Domain dataset group

A dataset group containing preconfigured resources for different business domains and use cases. Amazon Personalize manages the life cycle of training models and deployment. When you create a Domain dataset group, you choose your business domain, import your data, and create recommenders for each of your use cases. You use your recommender in your application to get recommendations with the GetRecommendations operation.

If you start with a Domain dataset group, you can still add custom resources such as solutions and solution versions trained with recipes for custom use cases.

Custom dataset group

A dataset group containing only custom resources, including solutions, solution versions, filters, campaigns, and batch inference jobs. You use a campaign to get recommendations with the GetRecommendations operation. You manage the life cycle of training models and deployment. If you start with a Custom dataset group, you can't associate it with a domain later. Instead, create a new Domain 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 Item 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 Item 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 Item 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.

item-affinity

A USER_SEGMENTATION recipe that uses the data from an Item interactions dataset and Items dataset to create user segments for each item that you specify based on the likelihood that the users will interact with the item.

item-attribute-affinity

A USER_SEGMENTATION recipe that uses the data from an Item interactions dataset and Items dataset to create a user segment for each item attribute that you specify based on the likelihood that the users will interact with items with the attribute.

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.

recommender

A Domain dataset group tool that generates recommendations. You create a recommender for a Domain dataset group and use in your application to get real-time recommendations with the GetRecommendations API. When you create a recommender, you specify a use case and Amazon Personalize trains the models backing the recommender with the best configurations for the use case.

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 or Next-Best-Action, Amazon Personalize automatically updates the latest solution version trained with FULL training mode. See [Automatic updates](#).

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.

batch segment job

A tool that imports your batch input data from an Amazon S3 bucket, uses your solution version to create user segments, and exports the user segments 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 segment job with a solution backed by a USER_SEGMENTATION recipe to create segments of users based on the likelihood the user will interact with different items or items with different item attributes.

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

With exploration, recommendations include some items or actions that would be typically less likely to be recommended for the user, such as new items or actions, items or actions with few interactions, or items or actions less relevant for the user based on their previous behavior.

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).

user segments

Lists of user that Amazon Personalize predicts a user will interact with your catalogue. Depending on the USER_SEGMENTATION recipe used, you create user segments based on items (Item-Affinity recipe) item metadata (Item-Attribute-Affinity recipe). You create user segments with a batch segment job.

Types of data Amazon Personalize can use

The following topics introduce the different types of data that you can import into Amazon Personalize.

Topics

Interactions data

An *interaction* is an *event* that you record and then import as training data. Amazon Personalize generates recommendations primarily based on the interactions data. Interactions data can include the following:

- Event type and event value data
- Contextual metadata
- Impressions data

You import interactions data into an *Item interactions dataset*. For more details about interactions datasets, see Item interactions dataset.

Item data

The item metadata that Amazon Personalize can use includes the following:

- Numerical data about each item, such its price.
- Categorical metadata about each item, such as the item's genre or color.
- Creation timestamp data for each item.
- Unstructured text metadata, such as product descriptions or movie synopses.

You import metadata about your items into an *Items* dataset. For more information about Items datasets, see Items dataset.

User data

The user metadata Amazon Personalize can use includes the following:

- Numerical data about each user, such as their age.
- Categorical metadata about each user, such as their gender or loyalty membership status.

You import metadata about your users into a *Users* dataset. For more information about Users datasets, see Users dataset.

Actions data

The action data Amazon Personalize can use includes the following:

- The business value or importance of each action.
- Categorical metadata for each action, such as seasonality or action exclusivity.
- Action expiration timestamp data that specifies when Amazon Personalize should stop recommending each action.
- Repeat frequency data that specifies long Amazon Personalize should wait before recommending each action after a user interacts with it.

You import data about your actions into a *Actions dataset*. For more information about Actions datasets, see Actions dataset.

Actions interactions data

The data Amazon Personalize can use from user interactions with actions includes the following:

- Event type data
- Categorical metadata

You import interactions data into an *Action interactions dataset*. For more details about Action interactions datasets, see Action interactions dataset.

Setting up Amazon Personalize

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

Topics

Sign up for an AWS account

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

To sign up for an AWS account

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

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

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

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

Create an administrative user

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

Secure your AWS account root user

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

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

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

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

Create an administrative user

1. Enable IAM Identity Center.

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

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

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

Sign in as the administrative user

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

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

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

You must give users, groups, or roles permission to interact with Amazon Personalize resources. And you must give Amazon Personalize permission to access the resources you create in Amazon Personalize and to perform tasks on your behalf.

To set up permissions

1. Give your users, groups, or roles permission to interact with Amazon Personalize resources and pass a role to Amazon Personalize. See Giving users permission to access Amazon Personalize.
2. Give Amazon Personalize permission to access your resources in Amazon Personalize and permission to perform tasks on your behalf. See Giving Amazon Personalize permission to access your resources.
3. Modify your Amazon Personalize service role's trust policy so it prevents the confused deputy problem. For a trust relationship policy example, see Cross-service confused deputy prevention. For information modifying a role's trust policy, see Modifying a role.
4. If you use AWS Key Management Service (AWS KMS) for encryption, you must grant Amazon Personalize and your Amazon Personalize IAM service role permission to use your key. For more information, see Giving Amazon Personalize permission to use your AWS KMS key.
5. Complete the steps in Giving Amazon Personalize access to Amazon S3 resources to use IAM and Amazon S3 bucket policies to give Amazon Personalize access to your Amazon S3 resources.

Topics

- Giving users permission to access Amazon Personalize
- Giving Amazon Personalize permission to access your resources
- Giving Amazon Personalize access to Amazon S3 resources

- [Giving Amazon Personalize permission to use your AWS KMS key]

Giving users permission to access Amazon Personalize

To provide your users access to Amazon Personalize, you create an IAM policy that grants permission to access your Amazon Personalize resources and pass a role to Amazon Personalize. Then you use that policy when you add permissions to your users, groups or roles.

Creating a new IAM policy for your users

Create an IAM policy that provides Amazon Personalize full access to your Amazon Personalize resources.

To use the JSON policy editor to create a policy

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane on the left, choose **Policies**.

   If this is your first time choosing **Policies**, the **Welcome to Managed Policies** page appears. Choose **Get Started**.
3. At the top of the page, choose **Create policy**.
4. In the **Policy editor** section, choose the **JSON** option.
5. Enter the following JSON policy document:

```
{
    "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**.

> **Note**
>
> You can switch between the **Visual** and **JSON** editor options anytime. However, if you make changes or choose **Next** in the **Visual** editor, IAM might restructure your policy to optimize it for the visual editor. For more information, see Policy restructuring in the *IAM User Guide*.

7. On the **Review and create** page, enter a **Policy name** and a **Description** (optional) for the policy that you are creating. Review **Permissions defined in this policy** to see the permissions that are granted by your policy.
8. Choose **Create policy** to save your new policy.

To grant only the permissions required to perform a task in Amazon Personalize, modify the preceding policy to include only the required actions for your user. For a complete list of Amazon Personalize actions, see Actions, resources, and condition keys for Amazon Personalize.

Providing access to Amazon Personalize

Attach the new IAM policy when you provide permissions to your users.

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

- Users and groups in AWS IAM Identity Center:

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

Create a role for identity federation. Follow the instructions in Creating a role for a third-party identity provider (federation) in the *IAM User Guide*.

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

Giving Amazon Personalize permission to access your resources

To give Amazon Personalize permission to access your resources, you create an IAM policy that provides Amazon Personalize full access to your Amazon Personalize resources. Or you can use the AWS managed `AmazonPersonalizeFullAccess` policy. `AmazonPersonalizeFullAccess` provides more permissions than are necessary. We recommend creating a new IAM policy that only grants the necessary permissions. For more information about managed policies, see AWS managed policies.

After you create a policy, you create an IAM role for Amazon Personalize and attach the new policy to it.

Topics

- Creating a new IAM policy for Amazon Personalize
- Creating an IAM role for Amazon Personalize

Creating a new IAM policy for Amazon Personalize

Create an IAM policy that provides Amazon Personalize full access to your Amazon Personalize resources.

To use the JSON policy editor to create a policy

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane on the left, choose **Policies**.

If this is your first time choosing **Policies**, the **Welcome to Managed Policies** page appears. Choose **Get Started**.

3. At the top of the page, choose **Create policy**.
4. In the **Policy editor** section, choose the **JSON** option.
5. Enter the following JSON policy document:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "personalize:*"
            ],
            "Resource": "*"
        }
    ]
}
```

6. Choose **Next**.

> **Note**
>
> You can switch between the **Visual** and **JSON** editor options anytime. However, if you make changes or choose **Next** in the **Visual** editor, IAM might restructure your policy to optimize it for the visual editor. For more information, see Policy restructuring in the *IAM User Guide*.

7. On the **Review and create** page, enter a **Policy name** and a **Description** (optional) for the policy that you are creating. Review **Permissions defined in this policy** to see the permissions that are granted by your policy.
8. Choose **Create policy** to save your new policy.

Creating an IAM role for Amazon Personalize

To use Amazon Personalize, you must create an AWS Identity and Access Management service role for Amazon Personalize. 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*. After you create a service role for Amazon Personalize, grant the role additional permissions listed in Additional service role permissions as necessary.

To create the service role for Amazon Personalize (IAM console)

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane of the IAM console, choose **Roles**, and then choose **Create role**.
3. For **Trusted entity type**, choose **AWS service**.
4. For **Service or use case**, choose **Amazon Personalize**, and then choose the **Personalize** use case.
5. Choose **Next**.
6. Chose the policy that you created in the previous procedure.
7. (Optional) Set a permissions boundary. This is an advanced feature that is available for service roles, but not service-linked roles.
    a. Open the **Set permissions boundary** section, and then choose **Use a permissions boundary to control the maximum role permissions**.

       IAM includes a list of the AWS managed and customer-managed policies in your account.
    b. Select the policy to use for the permissions boundary.
8. Choose **Next**.
9. Enter a role name or a role name suffix to help you identify the purpose of the role.

> **Important**
>
> When you name a role, note the following:
>
> - Role names must be unique within your AWS account, and can't be made unique by case.
>
>   For example, don't create roles named both **`PRODROLE`** and **`prodrole`**. When a role name is used in a policy or as part of an ARN, the role name is case sensitive, however when a role name appears to customers in the console, such as during the sign-in process, the role name is case insensitive.

- You can't edit the name of the role after it's created because other entities might reference the role.

10. (Optional) For **Description**, enter a description for the role.
11. (Optional) To edit the use cases and permissions for the role, in the **Step 1: Select trusted entities** or **Step 2: Add permissions** sections, choose **Edit**.
12. (Optional) To help identify, organize, or search for the role, add tags as key-value pairs. For more information about using tags in IAM, see Tagging IAM resources in the *IAM User Guide*.
13. Review the role, and then choose **Create role**.

After you create a role for Amazon Personalize, you are ready to grant it access to your Amazon S3 bucket and any AWS KMS keys.

Additional service role permissions

After you create the role and grant it permissions to access your resources in Amazon Personalize, do the following:

1. Modify your Amazon Personalize service role's trust policy so it prevents the confused deputy problem. For a trust relationship policy example, see Cross-service confused deputy prevention. For information modifying a role's trust policy, see Modifying a role.
2. If you use AWS Key Management Service (AWS KMS) for encryption, you must grant Amazon Personalize and your Amazon Personalize IAM service role permission to use your key. For more information, see Giving Amazon Personalize permission to use your AWS KMS key.

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 to set up permissions so Amazon Personalize can access your resources in Amazon Personalize on your behalf.
2. Attach a policy to the Amazon Personalize service role (see Creating an IAM role for Amazon Personalize) that allows access to your Amazon S3 bucket. For more information, see Attaching an Amazon S3 policy to your Amazon Personalize service role.

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.
4. If you use AWS Key Management Service (AWS KMS) for encryption, you must grant Amazon Personalize and your Amazon Personalize IAM service role permission to use your key. For more information, see Giving Amazon Personalize permission to use your AWS KMS key.

> **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

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 role policy for batch workflows.

```
{
    "Version": "2012-10-17",
    "Id": "PersonalizeS3BucketAccessPolicy",
    "Statement": [
        {
            "Sid": "PersonalizeS3BucketAccessPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

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 role for Amazon Personalize](#).
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](#).

Service role policy for batch workflows

To complete a batch worklfow, 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 [Batch recommendations and user segments (custom resources)](#).

```
{
    "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 role policy for exporting a dataset

To export a dataset, your Amazon Personalize service 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 role for Amazon Personalize. For information about attaching policies to a service role see Attaching an Amazon S3 policy to your Amazon Personalize service role.

```
{
    "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](#).

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 [Batch recommendations and user segments (custom resources)](#).

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

Giving Amazon Personalize permission to use your AWS KMS key

If you specify a AWS Key Management Service (AWS KMS) key when you use the Amazon Personalize console or APIs, or if you use your AWS KMS key to encrypt an Amazon S3 bucket, you must grant Amazon Personalize permission to use your key. To grant permissions, your AWS KMS key policy *and* IAM policy attached to your service role must grant Amazon Personalize permission to use your key. This applies for creating the following in Amazon Personalize.

- Dataset groups
- Dataset import job (only AWS KMS key policy must grant permissions)
- Dataset export jobs
- Batch inference jobs
- Batch segment jobs
- Metric attributions

Your AWS KMS key policy and IAM policies must grant permissions for the following actions:

- Decrypt
- GenerateDataKey
- DescribeKey
- CreateGrant (only required in key policy)
- ListGrants

Revoking AWS KMS key permissions after creating a resource can lead to issues when creating a filter or getting recommendations. For more information about AWS KMS policies, see Using key policies in AWS KMS in the *AWS Key Management Service Developer Guide*. For information on creating an IAM policy, see Creating IAM policies in the *IAM User Guide*. For information on attaching an IAM policy to role, see [Adding and removing IAM identity permissions](#) in the *IAM User Guide*.

Topics

- [Key policy example](#)
- [IAM policy example](#)

Key policy example

The following key policy example grants Amazon Personalize and your role the minimum permissions for the preceding Amazon Personalize operations. If you specify a key when you create a dataset group and want to export data from a dataset, your key policy must include the `GenerateDataKeyWithoutPlaintext` action.

```
{
  "Version": "2012-10-17",
  "Id": "key-policy-123",
  "Statement": [
    {
      "Sid": "Allow use of the key",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<account-id>:role/<personalize-role-name>",
        "Service": "personalize.amazonaws.com"
      },
      "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:DescribeKey",
                "kms:CreateGrant",
                "kms:ListGrants"
            ],
      "Resource": "*"
    }
  ]
}
```

IAM policy example

The following IAM policy example grants a role the minimum AWS KMS permissions required for the preceding Amazon Personalize operations. For dataset import jobs, only the AWS KMS key policy needs to grant permissions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:DescribeKey",
                "kms:ListGrants"
            ],
            "Resource": "*"
        }
    ]
}
```

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 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 [****************xgyZ]:
   ```

```
Default region name [us-west-2]:
Default output format [json]:
```

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

   ```
   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), SDK for Java 2.x, and SDK for JavaScript v3. For information about other AWS SDKs, see Tools for Amazon Web Services. For information about setting up Amplify, see AWS Amplify.

- AWS SDK for Python (Boto3)

  To install the SDK for Python (Boto3), follow the Quickstart instructions in the Boto3 documentation.

- SDK for Java 2.x

To learn about setting up the SDK for Java 2.x, see the Get started with the SDK for Java 2.x topic in the *AWS SDK for Java 2.x Developer Guide*.

For code examples for Amazon Personalize, see Amazon Personalize Java code samples in the AWS SDK examples repository.

- AWS SDK for JavaScript v3

  To learn about setting up the SDK for JavaScript v3, see the Get started with the AWS SDK for JavaScript topic in the *AWS SDK for JavaScript Developer Guide*.

  For code examples for Amazon Personalize, see Amazon Personalize code examples for SDK for JavaScript v3 in the AWS SDK examples repository.

Getting started

The following sections help you get started using Amazon Personalize with the Amazon Personalize console, AWS CLI, and AWS SDKs. The tutorials use historical data that consists of 100,000 movie ratings on 9,700 movies from 600 users.

To simplify tutorials:

- We use a small dataset. This might negatively impact any metrics generated by resources. The tutorials serve as an introduction to the Amazon Personalize workflow and won't necessarily generate the highest performing models.
- We create only an Item interactions dataset, and 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.

You can choose to get started with a Domain dataset group or a Custom dataset group:

- Domain dataset groups provide resources that are optimized for different use cases based on your domain. To get started creating a Domain dataset group, complete the Getting started prerequisites and then complete the tutorial in Getting started with a Domain dataset group.
- Custom dataset groups allow you to create and configure only custom resources. To get started providing personalized movie recommendations for your users with a custom resources and the User-Personalization recipe, complete the Getting started prerequisites and then start the tutorials in Getting started with a Custom dataset group.

When you finish the getting started exercise, to avoid incurring unnecessary charges, follow the steps in Cleaning up resources to delete the resources you created.

Topics

Getting started prerequisites

The following steps are prerequisites for the getting started exercises.

1. Set up permissions so Amazon Personalize can access your resources on your behalf. This involves creating a service role for Amazon Personalize and granting it access to Amazon Personalize resources with an IAM policy. For more information see Giving Amazon Personalize permission to access your resources.
2. Prepare your training data and upload the data to your Amazon S3 bucket:
   - For Domain dataset group tutorials, see Creating the training data (Domain dataset group).
   - For Custom dataset group tutorials, see Creating the training data (Custom dataset group).
3. Give your Amazon Personalize service role permission to access your Amazon S3 resources, as specified in Giving Amazon Personalize access to Amazon S3 resources.

Creating the training data (Domain dataset group)

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.

To create the training data

1. Download and unzip the movie ratings zip file, ml-latest-small.zip from MovieLens under *recommended for education and development* (F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872).
2. Open the `ratings.csv` file. This file contains the interactions data for this tutorial.
   a. Delete the *rating* column.
   b. Rename the `userId` and `movieId` columns to `USER_ID` and `ITEM_ID` respectively.
   c. Add an EVENT_TYPE column set the value for every record to `watch`. If you're using Microsoft Excel, you can set the EVENT_TYPE for every record by entering `watch` in the first cell in the column and then double-clicking the bottom-right corner of the cell. Your header should be the following:

   ```
   USER_ID,ITEM_ID,TIMESTAMP,EVENT_TYPE
   ```

These columns must be exactly as shown for Amazon Personalize to recognize the data. The first few rows of your data should look as follows:

```
USER_ID,ITEM_ID,TIMESTAMP,EVENT_TYPE
1,1,964982703,watch
1,3,964981247,watch
1,6,964982224,watch
1,47,964983815,watch
1,50,964982931,watch
....
....
```

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].

Creating the training data (Custom dataset group)

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 and unzip the movie ratings zip file, ml-latest-small.zip from MovieLens under *recommended for education and development* (F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872).
2. Open the `ratings.csv` file. This file contains the interactions data for this tutorial.

   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.

Getting started with a Domain dataset group

In this getting started tutorial you create a Domain dataset group for the VIDEO_ON_DEMAND domain, import interactions data from a CSV file, and create a recommender with the *Top picks for you* use case. Then you use the recommender to get personalized movie recommendations for a user. The tutorial uses historical data that consists of 100,000 movie ratings on 9,700 movies from 600 users.

To begin, complete the Getting started prerequisites and then depending on how you want to create Amazon Personalize resources, proceed to Getting started with a Domain dataset group (console), Getting started with a Domain dataset group (SDK for Python (Boto3)), Getting started with a Domain dataset group (SDK for Java 2.x), or Getting started with a Domain dataset group (SDK for JavaScript v3).

When you finish the getting started exercise, to avoid incurring unnecessary charges, follow the steps in Cleaning up resources to delete the resources you created.

Topics

- Getting started with a Domain dataset group (console)
- Getting started with a Domain dataset group (SDK for Java 2.x)
- Getting started with a Domain dataset group (SDK for Python (Boto3))
- Getting started with a Domain dataset group (SDK for JavaScript v3)

Getting started with a Domain dataset group (console)

In this exercise, you use the Amazon Personalize console to create a Domain dataset group and a recommender that returns movie recommendations for a given user.

Before you start this exercise, review the Getting started prerequisites.

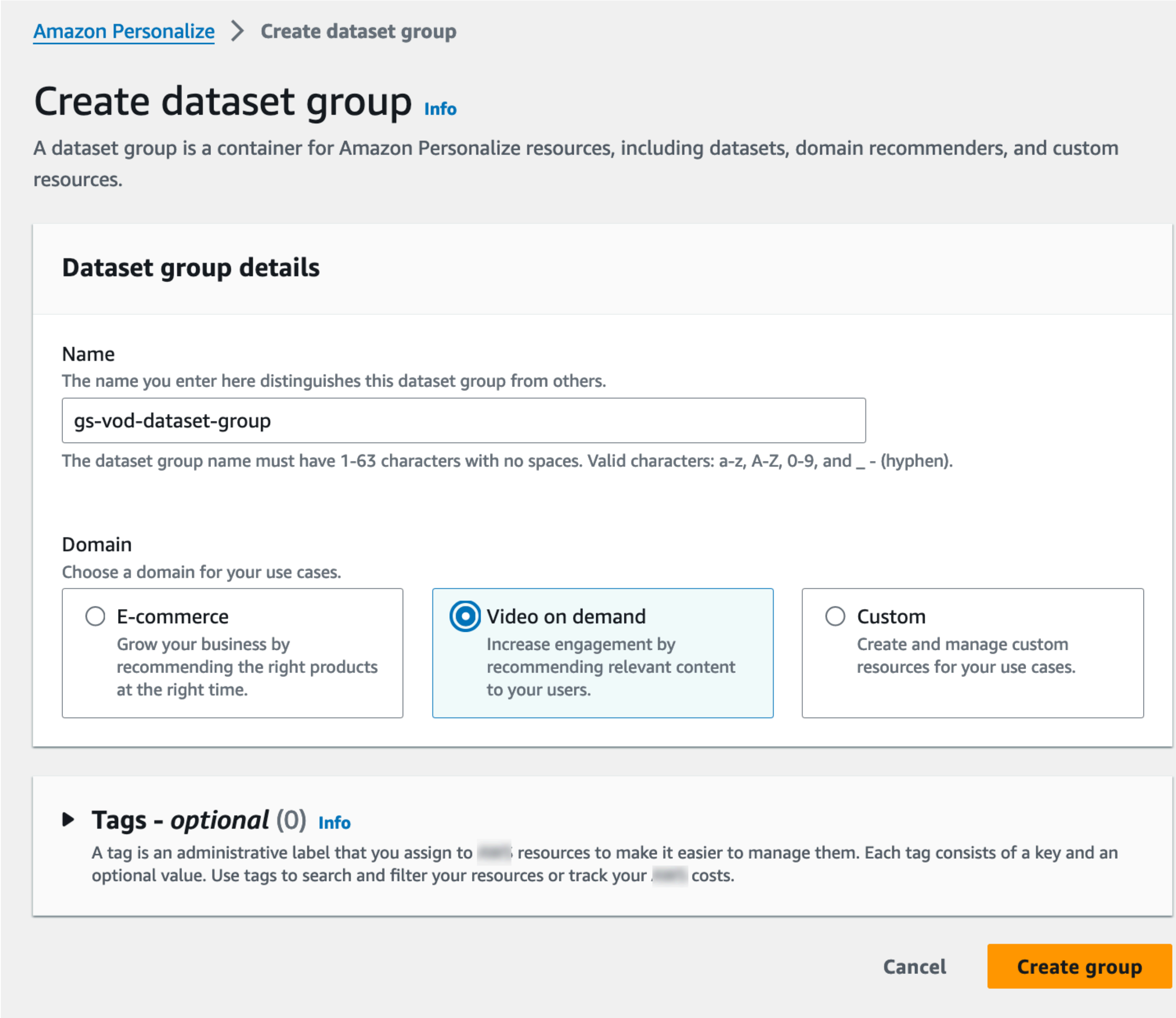
When you finish the getting started exercise, to avoid incurring unnecessary charges, follow the steps in Cleaning up resources to delete the resources you created.

Step 1: Create a Domain dataset group

In this procedure you create Domain dataset group for the VIDEO_ON_DEMAND domain, create an Item interactions dataset with the default schema for the VIDEO_ON_DEMAND domain, and import the item interactions data you created in Creating the training data (Domain dataset group).

To create a Domain dataset group

1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/home and sign in to your account.
2. In the navigation pane, choose **Create dataset group**.
3. In **Dataset group details**, specify a name for your dataset group.
4. For **Domain**, choose **Video on demand**. The domain you choose determines the default schema you use when importing data. It also determines what use cases are available for recommenders. Your screen should look similar to the following.



5. Choose **Create dataset group**. The Overview page appears. Proceed to [Step 2: Import data](#).

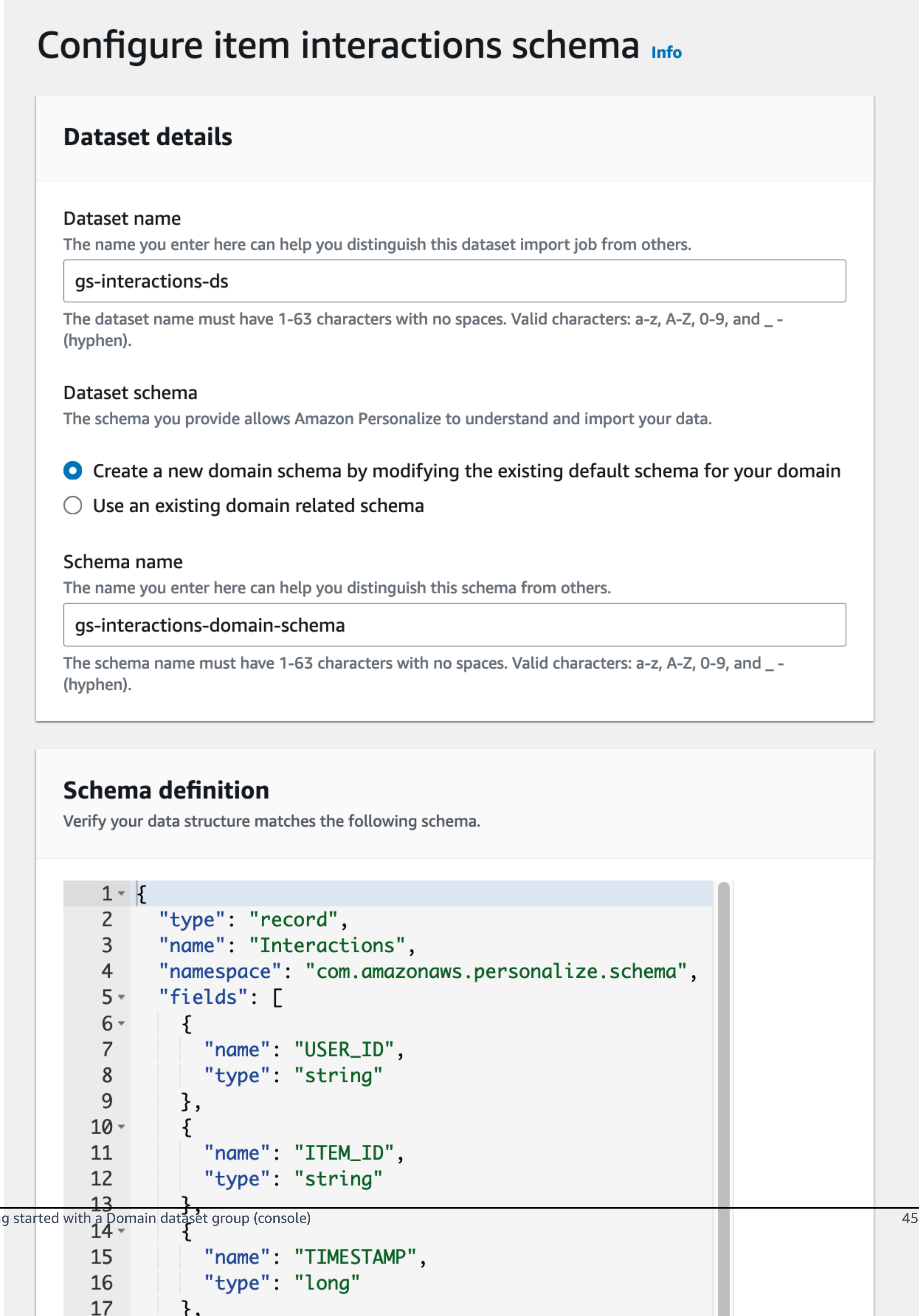
Step 2: Import data

In this procedure you create an Item interactions dataset with the default VIDEO_ON_DEMAND domain schema. Then you import the item interactions data you created in [Creating the training data (Domain dataset group)](#).

To import data

1. On the Overview page, in **Step 1. Create datasets and import data**, choose **Create dataset** and choose **Item interactions dataset**.

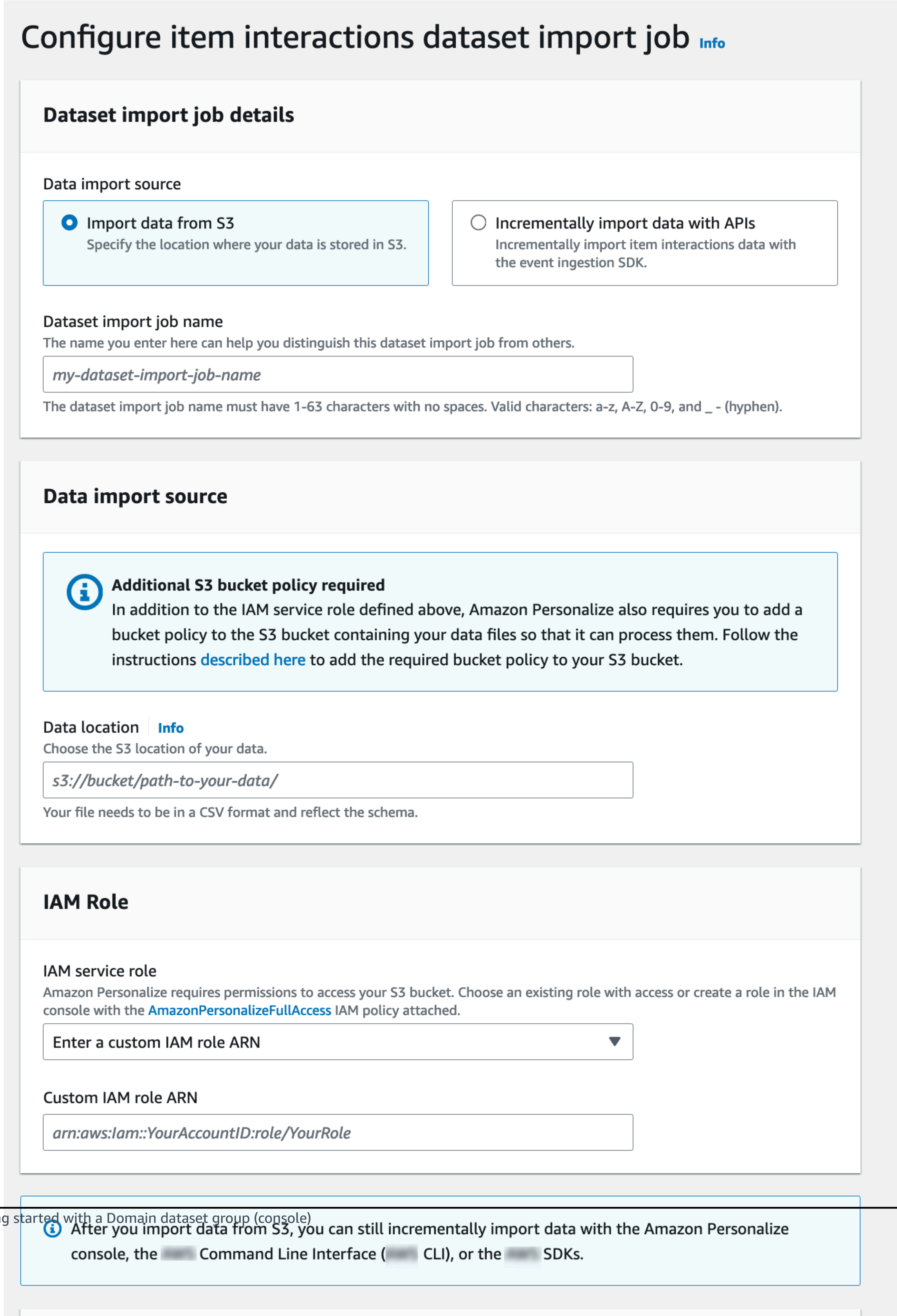
2. Choose **Import data directly into Amazon Personalize datasets** and choose **Next**.
3. On the **Configure item interactions schema** page, for **Dataset name** provide a name for your Item interactions dataset.
4. For **Dataset schema**, choose **Create a new domain schema by modifying the existing default schema for your domain** and enter a name for the schema. The **Schema definition** updates to display the default schema for the VIDEO_ON_DEMAND domain. Leave the schema unchanged. Your screen should look similar to the following.


Configure item interactions schema Info
Dataset details
Dataset name
The name you enter here can help you distinguish this dataset import job from others.
gs-interactions-ds
The dataset name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).
Dataset schema
The schema you provide allows Amazon Personalize to understand and import your data.
Create a new domain schema by modifying the existing default schema for your domain
Use an existing domain related schema
Schema name
The name you enter here can help you distinguish this schema from others.
gs-interactions-domain-schema
The schema name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).
Schema definition
Verify your data structure matches the following schema.
{
"type": "record",
"name": "Interactions",
"namespace": "com.amazonaws.personalize.schema",
"fields": [
{
"name": "USER_ID",
"type": "string"
},
{
"name": "ITEM_ID",
"type": "string"
},
{
"name": "TIMESTAMP",
"type": "long"
},

5. Choose **Next**. The **Configure item interactions dataset import job** page appears.
6. On the **Configure item interactions dataset import job** page, leave the **Data import source** unchanged as **Import data from S3**.
7. For **Dataset import job name**, give your import job a name.
8. In **Data import source**, specify where your data is stored in Amazon Simple Storage Service (S3). Use the following syntax:

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

9. In **IAM role**, for **IAM service role** choose **Enter a custom IAM role ARN** and enter the Amazon Resource Name (ARN) of the role you created in Creating an IAM role for Amazon Personalize. Your screen should look similar to the following.


Configure item interactions dataset import job Info
Dataset import job details
Data import source
Import data from S3
Specify the location where your data is stored in S3.
Incrementally import data with APIs
Incrementally import item interactions data with the event ingestion SDK.
Dataset import job name
The name you enter here can help you distinguish this dataset import job from others.
my-dataset-import-job-name
The dataset import job name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).
Data import source
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://bucket/path-to-your-data/
Your file needs to be in a CSV format and reflect the schema.
IAM Role
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::YourAccountID:role/YourRole
After you import data from S3, you can still incrementally import data with the Amazon Personalize console, the Command Line Interface (CLI), or the SDKs.

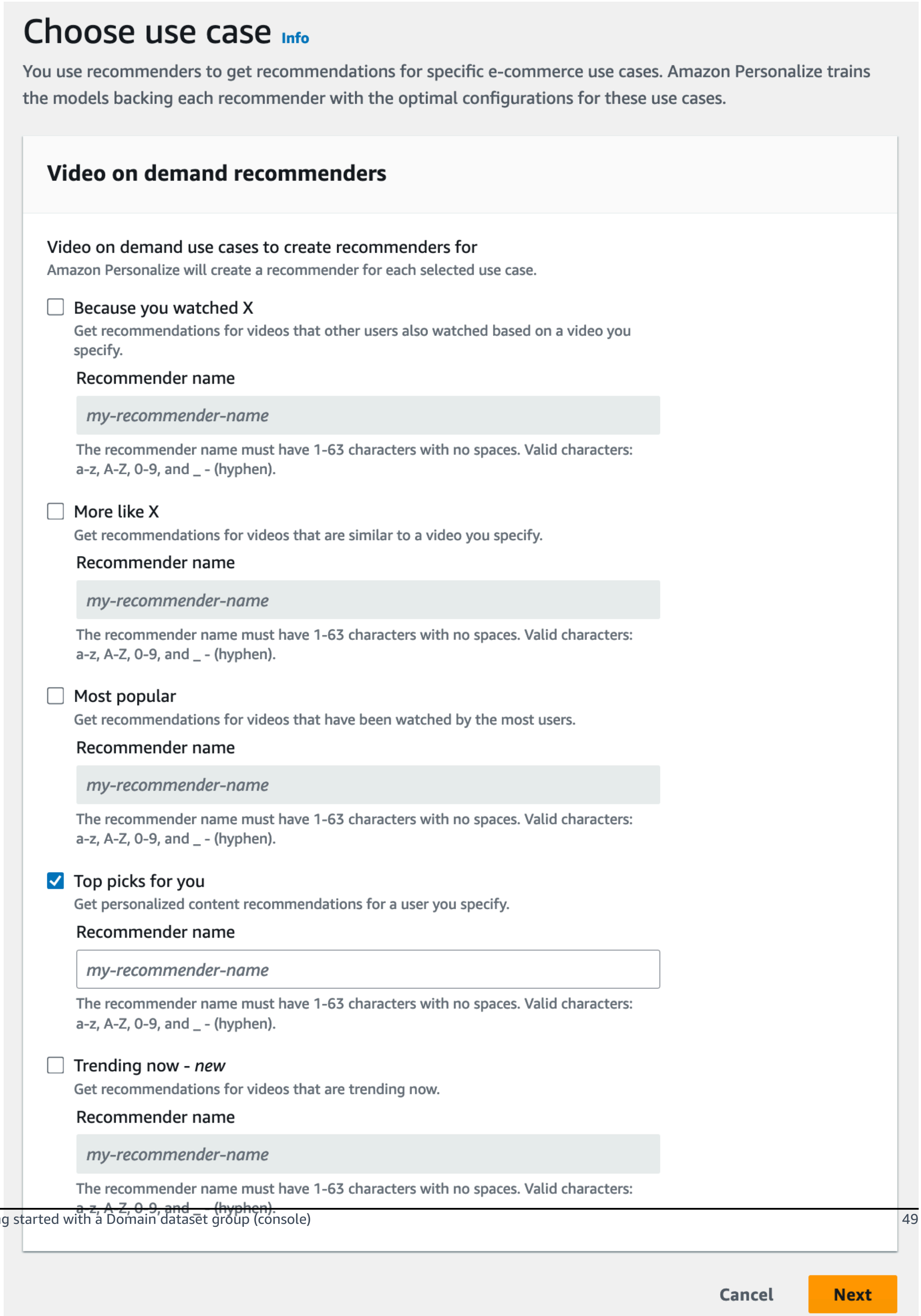
10. Choose **Start import** to import data. The **Overview** page for your Domain dataset group appears. Note the status of the import in the **Set up datasets** section. When the status is `Interaction data active` proceed to [Step 3: Create a recommender](#).

Step 3: Create a recommender

In this procedure, you create a recommender for the *Top picks for you* use case for the VIDEO_ON_DEMAND domain.

To create a recommender

1. On the **Overview** page for your Domain dataset group, in **Step 3** choose the **Use video on demand recommenders** tab and choose **Create recommenders**.
2. On the **Choose use case** page, choose **Top picks for you** and provide a **Recommender name**. Your screen should appear similar to the following.


Choose use case Info
You use recommenders to get recommendations for specific e-commerce use cases. Amazon Personalize trains the models backing each recommender with the optimal configurations for these use cases.
Video on demand recommenders
Video on demand use cases to create recommenders for
Amazon Personalize will create a recommender for each selected use case.
Because you watched X
Get recommendations for videos that other users also watched based on a video you specify.
Recommender name
my-recommender-name
The recommender name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).
More like X
Get recommendations for videos that are similar to a video you specify.
Recommender name
my-recommender-name
The recommender name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).
Most popular
Get recommendations for videos that have been watched by the most users.
Recommender name
my-recommender-name
The recommender name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).
Top picks for you
Get personalized content recommendations for a user you specify.
Recommender name
my-recommender-name
The recommender name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).
Trending now - new
Get recommendations for videos that are trending now.
Recommender name
my-recommender-name
The recommender name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).
Cancel
Next

3. Choose **Next**.
4. Leave the fields on the **Advanced configuration** page unchanged and choose **Next**.
5. Review the recommender details and choose **Create recommenders** to create your recommender.

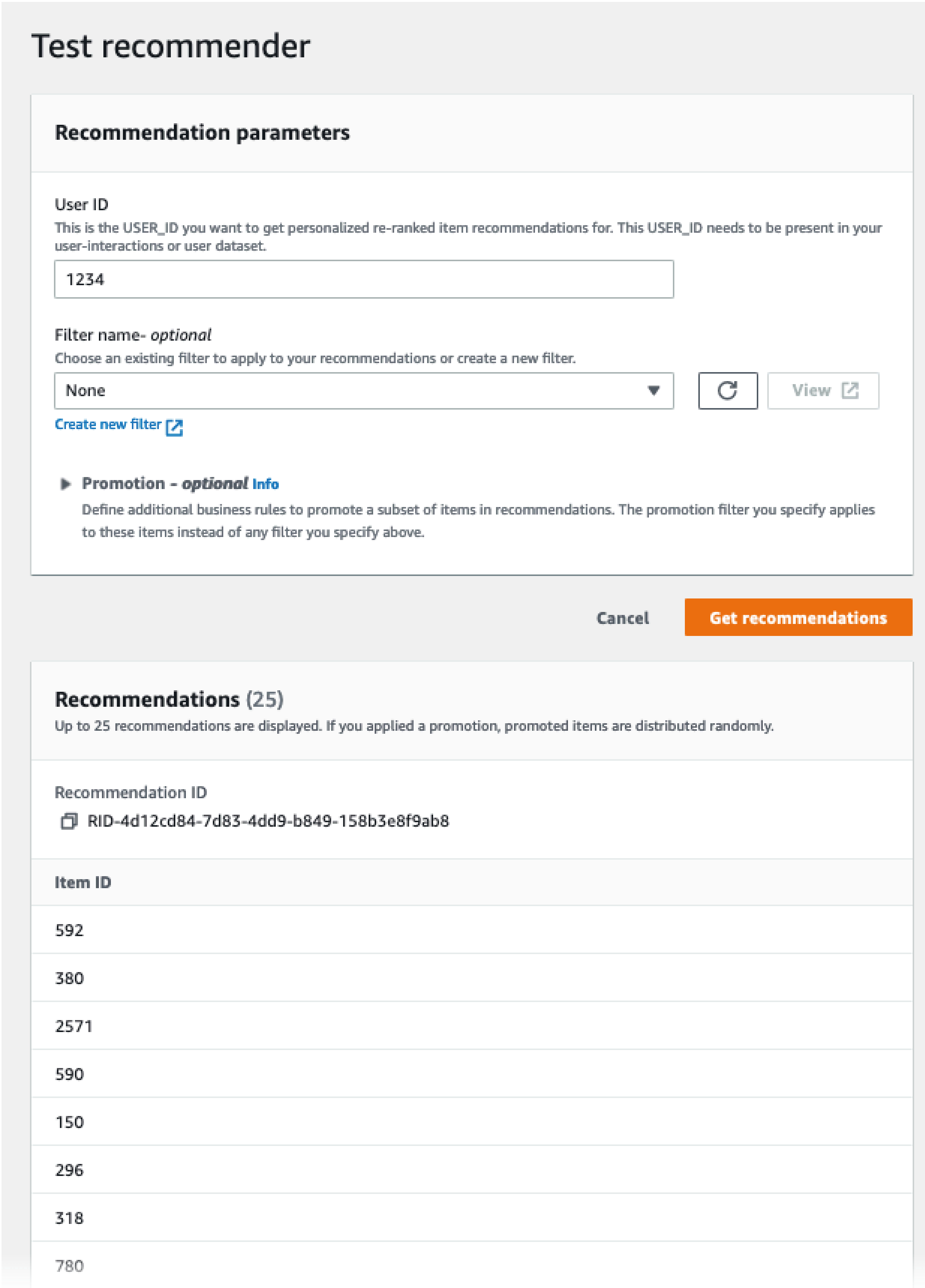
   You can monitor the status of each recommender on the **Recommenders** page. When your recommender status is **Active**, you can use it to get recommendations in Step 4: Get recommendations.

Step 4: Get recommendations

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

To get recommendations

1. On the Overview page for your Domain dataset group, in the navigation pane choose **Recommenders**.
2. On the **Recommenders** page, choose your recommender.
3. At the top right, choose **Test**.
4. In **Recommendation parameters**, enter a user ID. Leave the other fields unchanged.
5. Choose **Get recommendations**. A table containing the user's top 25 recommended items appears. Your screen should look similar to the following.


Test recommender
Recommendation parameters
User ID
This is the USER_ID you want to get personalized re-ranked item recommendations for. This USER_ID needs to be present in your user-interactions or user dataset.
1234
Filter name- optional
Choose an existing filter to apply to your recommendations or create a new filter.
None
View
Create new filter
Promotion - optional Info
Define additional business rules to promote a subset of items in recommendations. The promotion filter you specify applies to these items instead of any filter you specify above.
Cancel
Get recommendations
Recommendations (25)
Up to 25 recommendations are displayed. If you applied a promotion, promoted items are distributed randomly.
Recommendation ID
RID-4d12cd84-7d83-4dd9-b849-158b3e8f9ab8
Item ID
592
380
2571
590
150
296
318
780

Getting started with a Domain dataset group (SDK for Java 2.x)

This tutorial shows you how to use the SDK for Java 2.x to create a Domain dataset group for the VIDEO_ON_DEMAND domain. In this tutorial, you create a recommender for the *Top picks for you* use case.

To avoid incurring unnecessary charges, when you finish the getting started exercise see Cleaning up resources for information on deleting the resources you create in the tutorial.

Prerequisites

The following are prerequisite steps for completing this tutorial:

- Complete the Getting started prerequisites to set up the required permissions and create the training data. If you also completed the Getting started with a Domain dataset group (console), you can reuse the same source data. If you are using your own source data, make sure that your data is formatted like in the prerequisites.
- 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*.

Tutorial

In the following steps, you set up your project to use Amazon Personalize packages and create Amazon Personalize SDK for Java 2.x clients. Then you import data, create a recommender for the *Top picks for you* use case, and get recommendations.

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;
// 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;
// recommender packages
import software.amazon.awssdk.services.personalize.model.CreateRecommenderRequest;
import software.amazon.awssdk.services.personalize.model.CreateRecommenderResponse;
import software.amazon.awssdk.services.personalize.model.DescribeRecommenderRequest;
// 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: Create Amazon Personalize clients

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

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

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. 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 that you created when you completed the Creating the training data (Domain dataset group).

```
{
  "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": "TIMESTAMP",
```

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

2. Use the following `createDomainSchema` method to create a domain schema in Amazon Personalize. Pass the following as parameters: an Amazon Personalize service client, the name for your schema, `VIDEO_ON_DEMAND` for the domain, and the file path for the schema JSON file that 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 createDomainSchema(PersonalizeClient personalizeClient,
 String schemaName, String domain,
            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)
                    .domain(domain)
                    .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 `createDomainDatasetGroup` method to create a domain dataset group. Pass the following as parameters: an Amazon Personalize service client, a name for the dataset group, and pass `VIDEO_ON_DEMAND` for the domain. The method returns the ARN of your new dataset group. Store it for later use.

```
    public static String createDomainDatasetGroup(PersonalizeClient
 personalizeClient,
            String datasetGroupName,
            String domain) {

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

4. Create an Item interactions dataset. Use the following `createDataset` method to create an Item 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, a name for the job, and your Interactions dataset's ARN. Pass the Amazon S3 bucket path (`s3://`*`bucket name`*`/`*`folder name`*`/ratings.csv`) where you stored the training data, and your service role's ARN. You created this role as part of the Getting started prerequisites. 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 "";
    }
```

Step 4: Create a recommender

After your dataset import job completes, you are ready create a recommender. Use the following `createRecommender` method to create a recommender. Pass the following as parameters: an Amazon Personalize service client, a name for the recommender, your dataset group's Amazon Resource Name (ARN), and `arn:aws:personalize:::recipe/aws-vod-top-picks` for the recipe ARN. The method returns the ARN of your new recommender. Store it for later use.

```
    public static String createRecommender(PersonalizeClient personalizeClient,
            String name,
            String datasetGroupArn,
            String recipeArn) {

        long maxTime = 0;
        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String recommenderStatus = "";

        try {
            CreateRecommenderRequest createRecommenderRequest =
CreateRecommenderRequest.builder()
                    .datasetGroupArn(datasetGroupArn)
                    .name(name)
                    .recipeArn(recipeArn)
                    .build();

            CreateRecommenderResponse recommenderResponse = personalizeClient
                    .createRecommender(createRecommenderRequest);
            String recommenderArn = recommenderResponse.recommenderArn();
            System.out.println("The recommender ARN is " + recommenderArn);

            DescribeRecommenderRequest describeRecommenderRequest =
DescribeRecommenderRequest.builder()
                    .recommenderArn(recommenderArn)
                    .build();

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

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

                recommenderStatus =
personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
                        .status();
                System.out.println("Recommender status: " + recommenderStatus);
```

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

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

Step 5: Get recommendations

After you create a recommender, 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 recommender you created in the previous step, and a user ID (for example, `123`). The method prints the list of recommended items to the screen.

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

        try {
            GetRecommendationsRequest recommendationsRequest =
GetRecommendationsRequest.builder()
                    .recommenderArn(recommenderArn)
                    .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 started with a Domain dataset group (SDK for Python (Boto3))

This tutorial shows you how to use the SDK for Python (Boto3) to create a Domain dataset group for the VIDEO_ON_DEMAND domain. In this tutorial, you create a recommender for the *Top picks for you* use case.

To avoid incurring unnecessary charges, when you finish this getting started exercise delete the resources you create in this tutorial. For more information, see Cleaning up resources.

Topics

Prerequisites

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

- Complete the Getting started prerequisites to set up the required permissions and create the training data. If you are using your own source data, make sure that your data is formatted like in the prerequisites.
- Set up your AWS SDK for Python (Boto3) environment as specified in Setting up the AWS SDKs.

Tutorial

In the following steps, you verify your environment and create SDK for Python (Boto3) clients for Amazon Personalize. Then you import data, create a recommender for the *Top picks for you* use case, and get recommendations.

Step 1: Verify your Python environment and create boto3 clients

After you complete the prerequisites, run the following Python example to confirm that your environment is configured correctly. This code also creates the Amazon Personalize boto3 clients that you use in this tutorial. If your environment is configured correctly, a list of the available recipes is displayed and you can run the other examples in this tutorial.

```
import boto3

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

response = personalize.list_recipes()

for recipe in response['recipes']:
    print (recipe)
```

Step 2: Import data

After you create Amazon Personalize boto3 clients and verify your environment, import the historical data you created when you completed the Getting started prerequisites. To import historical data into Amazon Personalize, do the following:

1. Use the following code to create a schema in Amazon Personalize. Replace `gs-domain-interactions-schema` with a name for the schema.

```
import json
schema = {
  "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": "TIMESTAMP",
            "type": "long"
        }
    ],
    "version": "1.0"
}

create_interactions_schema_response = personalize.create_schema(
    name='gs-domain-interactions-schema',
    schema=json.dumps(schema),
    domain='VIDEO_ON_DEMAND'
)

interactions_schema_arn = create_interactions_schema_response['schemaArn']
print(json.dumps(create_interactions_schema_response, indent=2))
```

2. Create a dataset group with the following code. Replace `dataset group name` with a name for the dataset group.

```
response = personalize.create_dataset_group(
  name = 'dataset group name',
  domain = 'VIDEO_ON_DEMAND'
)
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'])
```

3. Create an Item interactions dataset in your new dataset group with the following code. Give the dataset a name and provide the `schema_arn` and `dataset_group_arn` from the previous steps.

```
response = personalize.create_dataset(
    name = 'interactions-dataset-name',
    schemaArn = interactions_schema_arn,
    datasetGroupArn = dsg_arn,
```

```
    datasetType = 'INTERACTIONS'
)

dataset_arn = response['datasetArn']
```

4. Import your data with a dataset import job with the following code. The code uses the describe_dataset_import_job method to track the status of the job.

   Pass the following as parameters: a name for the job, the `dataset_arn` from the previous step, the Amazon S3 bucket path (`s3://bucket name/folder name/ratings.csv`) where you stored the training data, and your IAM service role's ARN. You created this role as part of the [Getting started prerequisites](). Amazon Personalize needs permission to access the bucket. For information on granting access, see [Giving Amazon Personalize access to Amazon S3 resources]().

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

dataset_interactions_import_job_arn = response['datasetImportJobArn']

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

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

max_time = time.time() + 3*60*60 # 3 hours
while time.time() < max_time:
    describe_dataset_import_job_response = personalize.describe_dataset_import_job(
        datasetImportJobArn = dataset_interactions_import_job_arn
    )
    status = describe_dataset_import_job_response["datasetImportJob"]['status']
    print("Interactions DatasetImportJob: {}".format(status))

    if status == "ACTIVE" or status == "CREATE FAILED":
        break
```

```
    time.sleep(60)
```

Step 4: Create a recommender

After your dataset import job completes, you are ready create a recommender. Use the following code to create a recommender. Pass the following as parameters: a name for the recommender, your dataset group's Amazon Resource Name (ARN), and `arn:aws:personalize:::recipe/aws-vod-top-picks` for the recipe ARN. The code uses the describe_recommender method to track the status of the recommender.

```
import time
create_recommender_response = personalize.create_recommender(
  name = 'gs-python-top-picks',
  recipeArn = 'arn:aws:personalize:::recipe/aws-vod-top-picks',
  datasetGroupArn = dsg_arn
)
recommender_arn = create_recommender_response['recommenderArn']

print('Recommender ARN:' + recommender_arn)
max_time = time.time() + 3*60*60 # 3 hours
while time.time() < max_time:

    version_response = personalize.describe_recommender(
        recommenderArn = recommender_arn
    )
    status = version_response["recommender"]["status"]

    if status == "ACTIVE":
        print("Creation succeeded for {}".format(recommender_arn))

    elif status == "CREATE FAILED":
        print("Creation failed for {}".format(recommender_arn))

    if status == "ACTIVE":
        break
    else:
        print("Recommender creation is still in progress")

    time.sleep(60)
```

Step 5: Get recommendations

After you create a recommender, you use it to get recommendations with the following code. Pass as parameters the Amazon Resource Name (ARN) of the recommender you created in the previous step, and a user ID (for example, 123). The method prints the list of recommended items.

```
response = personalizeRt.get_recommendations(
    recommenderArn = "arn:aws:personalize:us-west-2:014025156336:recommender/gs-python-top-picks-89",
    userId = '123'
)
print("Recommended items")
for item in response['itemList']:
    print (item['itemId'])
```

Getting started using Amazon Personalize APIs with Jupyter (iPython) notebooks

To get started creating Domain dataset groups with Jupyter notebooks, clone or download a series of notebooks found in the notebooks_managed_domains folder of the Amazon Personalize samples repository. The notebooks walk you through importing training data, creating a recommender, and getting recommendations with Amazon Personalize.

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

Getting started with a Domain dataset group (SDK for JavaScript v3)

This tutorial shows you how to use the AWS SDK for JavaScript v3 to create a Domain dataset group for the VIDEO_ON_DEMAND domain. In this tutorial, you create a recommender for the *Top picks for you* use case.

To view the code used in this tutorial on GitHub, see Amazon Personalize code examples for SDK for JavaScript v3 in the *AWS SDK Code Examples* repository.

To avoid incurring unnecessary charges, when you finish this getting started exercise delete the resources you create in this tutorial. For more information, see Cleaning up resources.

Topics

- Prerequisites
- Tutorial

Prerequisites

The following are prerequisite steps for completing this tutorial:

- Complete the Getting started prerequisites to set up the required permissions and create the training data. If you also completed the Getting started with a Domain dataset group (console), you can reuse the same source data. If you are using your own source data, make sure that your data is formatted like in the prerequisites.
- Set up the SDK for JavaScript and AWS credentials as specified in the Setting up the SDK for JavaScript procedure in the *AWS SDK for JavaScript Developer Guide*.

Tutorial

In the following steps, you install the required dependencies. Then you create a dataset group, import data, create a recommender for the *Top picks for you* use case, and get recommendations.

If you use Node.js, you can run each code sample by saving the sample as a JavaScript file and then running `node <fileName.js>`.

Step 1: Install Amazon Personalize dependencies

After you complete the prerequisites, install the following Amazon Personalize dependencies:

- @aws-sdk/client-personalize
- @aws-sdk/client-personalize-runtime
- @aws-sdk/client-personalize-events (optional for this tutorial, but required if you want to record events after you create your recommender)

The following is an example of a `package.json` file you can use. To install the dependencies with Node.js, navigate to where you saved the `package.json` file and run `npm install`.

```
{
  "name": "personalize-js-project",
```

```
  "version": "1.0.0",
  "description": "personalize operations",
  "type": "module",
  "author": "Author Name <email@address.com>",
  "license": "ISC",
  "dependencies": {
    "@aws-sdk/client-personalize": "^3.350.0",
    "@aws-sdk/client-personalize-events": "^3.350.0",
    "@aws-sdk/client-personalize-runtime": "^3.350.0",
    "fs": "^0.0.1-security"
  },
  "compilerOptions": {
    "resolveJsonModule": true,
    "esModuleInterop": true
  }
}
```

Step 2: Create Amazon Personalize clients

After you install the dependencies, create your Amazon Personalize clients. In this tutorial, the code samples assume you create the clients in a file named `personalizeClients.js` stored in a directory named `libs`.

The following is an example of a `personalizeClient.js` file.

```
import { PersonalizeClient } from "@aws-sdk/client-personalize";
import { PersonalizeRuntimeClient } from "@aws-sdk/client-personalize-runtime";
import { PersonalizeEventsClient } from "@aws-sdk/client-personalize-events";
// Set your AWS region.
const REGION = "region"; //e.g. "us-east-1"

const personalizeClient = new PersonalizeClient({ region: REGION});
const personalizeEventsClient = new PersonalizeEventsClient({ region: REGION});
const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region: REGION});

export { personalizeClient, personalizeEventsClient, personalizeRuntimeClient };
```

Step 3: Import data

After you create your Amazon Personalize clients, import the historical data you created when you completed the Getting started prerequisites. 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 that you created when you completed the [Creating the training data (Domain dataset group)](#).

```
{
  "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": "TIMESTAMP",
          "type": "long"
      }
  ],
  "version": "1.0"
}
```

2. Create a domain schema in Amazon Personalize with the following `createDomainSchema.js` code. Replace `SCHEMA_PATH` with the path to the schema.json file you just created. Update the `createSchemaParam` to specify a name for the schema, and for `domain` specify `VIDEO_ON_DEMAND`.

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});
```

```
import fs from 'fs';

let schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
  mySchema = 'TEST' // for unit tests.
}

// Set the domain schema parameters.
export const createDomainSchemaParam = {
  name: 'NAME', /* required */
  schema: mySchema, /* required */
  domain: 'DOMAIN'   /* required for a domain dataset group, specify ECOMMERCE or
 VIDEO_ON_DEMAND */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateSchemaCommand(createDomainSchemaParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

3. Create a domain dataset group in Amazon Personalize with the following `createDomainDatasetGroup.js` code. Update the `domainDatasetGroupParams` to specify a name for the dataset group, and for `domain` specify `VIDEO_ON_DEMAND`.

```
// Get service clients module and commands using ES6 syntax.
import {  CreateDatasetGroupCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});
```

```
// Set the domain dataset group parameters.
export const domainDatasetGroupParams = {
  name: 'NAME',  /* required */
  domain: 'DOMAIN'   /* required for a domain dsg, specify ECOMMERCE or
 VIDEO_ON_DEMAND */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateDatasetGroupCommand(domainDatasetGroupParams));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

4. Create an Item interactions dataset in Amazon Personalize with the following `createDataset.js` code. Update the `createDatasetParam` to specify the Amazon Resource Name (ARN) of the dataset group and schema you just created, give the dataset a name, and for `datasetType`, specify `Interactions`.

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset's parameters.
export const createDatasetParam = {
  datasetGroupArn: 'DATASET_GROUP_ARN', /* required */
  datasetType: 'DATASET_TYPE', /* required */
  name: 'NAME', /* required */
  schemaArn: 'SCHEMA_ARN' /* required */
}

export const run = async () => {
  try {
```

```
    const response = await personalizeClient.send(new
 CreateDatasetCommand(createDatasetParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

5. Import your data with the following `createDatasetImportJob.js` code. Update the `datasetImportJobParam` to specify the following:

   - Specify a name for the job and specify your Interactions dataset's ARN.
   - For `dataLocation`, specify the Amazon S3 bucket path (`s3://`*`bucket name`*`/`*`folder name`*`/ratings.csv`) where you stored the training data.
   - For `roleArn` specify the Amazon Resource Name for your Amazon Personalize service role. You created this role as part of the [Getting started prerequisites](#).

```
// Get service clients module and commands using ES6 syntax.
import {CreateDatasetImportJobCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset import job parameters.
export const datasetImportJobParam = {
  datasetArn: 'DATASET_ARN', /* required */
  dataSource: {  /* required */
    dataLocation: 'S3_PATH'
  },
  jobName: 'NAME',/* required */
  roleArn: 'ROLE_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateDatasetImportJobCommand(datasetImportJobParam));
    console.log("Success", response);
    return response; // For unit tests.
```

```
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Step 4: Create a recommender

After your dataset import job completes, you are ready create a recommender. To create a recommender, use the following `createRecommender.js` code. Update the `createRecommenderParam` with the following: Specify a name for the recommender, specify your dataset group's ARN, and for `recipeArn` specify `arn:aws:personalize:::recipe/aws-vod-top-picks`.

```
// Get service clients module and commands using ES6 syntax.
import { CreateRecommenderCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the recommender's parameters.
export const createRecommenderParam = {
  name: 'NAME', /* required */
  recipeArn: 'RECIPE_ARN', /* required */
  datasetGroupArn: 'DATASET_GROUP_ARN'  /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateRecommenderCommand(createRecommenderParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Step 5: Get recommendations

After you create a recommender, you use it to get recommendations. Use the following `getRecommendations.js` code to get recommendations for a user. Update the `getRecommendationsParam` to specify the ARN of the recommender you created in the previous step, and specify a user ID (for example, 123).

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from
  "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region: "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
  recommenderArn: 'RECOMMENDER_ARN', /* required */
  userId: 'USER_ID',       /* required */
  numResults: 15    /* optional */
}

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
 GetRecommendationsCommand(getRecommendationsParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Getting started with a Custom dataset group

This getting started guide shows you how to provide personalized movie recommendations for your users with a Custom dataset group and the User-Personalization recipe. The tutorial uses historical data that consists of 100,000 movie ratings on 9,700 movies from 600 users.

To begin, complete the Getting started prerequisites and then proceed to either Getting started (console), Getting started (AWS CLI), Getting started (SDK for Python (Boto3)), or Getting started (SDK for Java 2.x).

When you finish the getting started exercise, to avoid incurring unnecessary charges, follow the steps in Cleaning up resources to delete the resources you created.

Topics

Getting started (console)

In this exercise, you use the Amazon Personalize console to create a Custom dataset group with a solution that returns movie recommendations for a given user. Before you start this exercise, review the Getting started prerequisites.

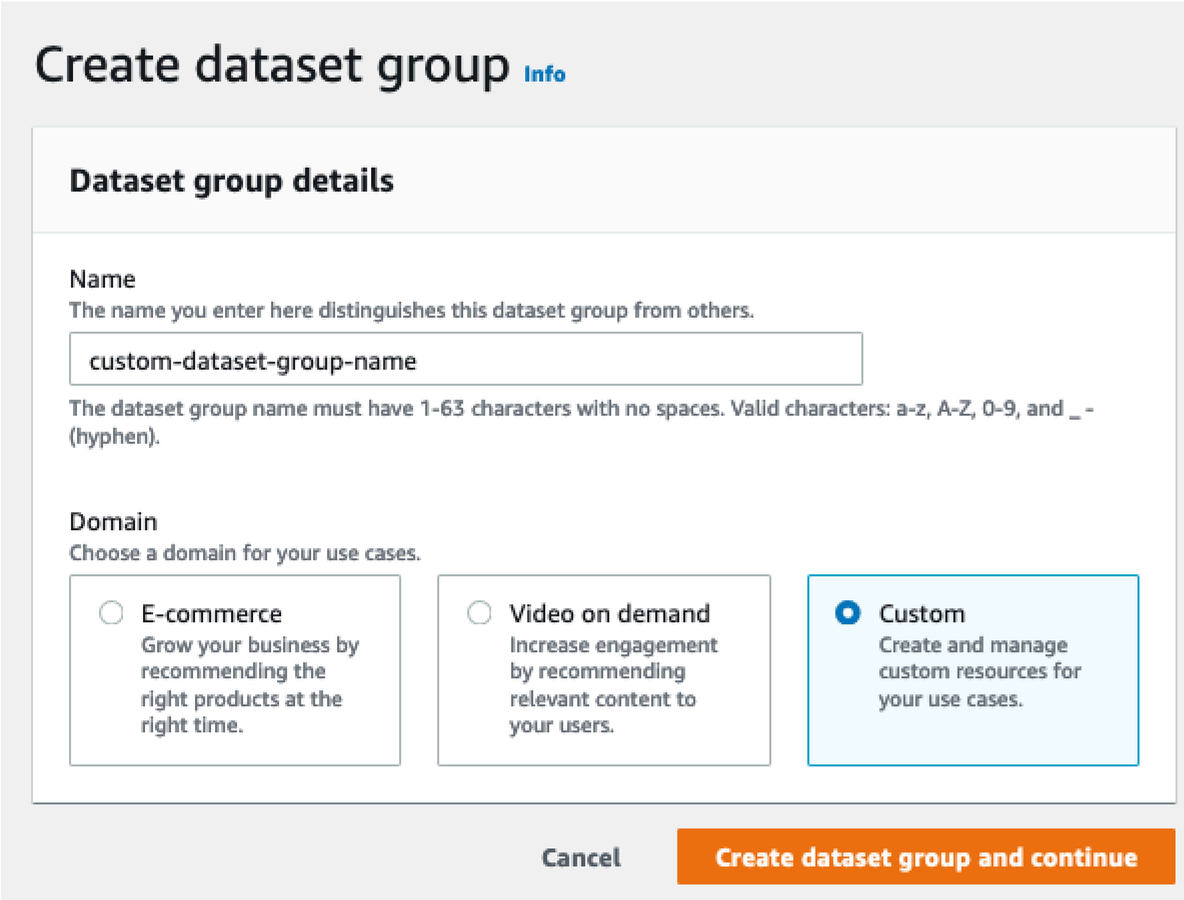
When you finish the getting started exercise, to avoid incurring unnecessary charges, follow the steps in Cleaning up resources to delete the resources you created.

Step 1: Create a dataset group and a dataset

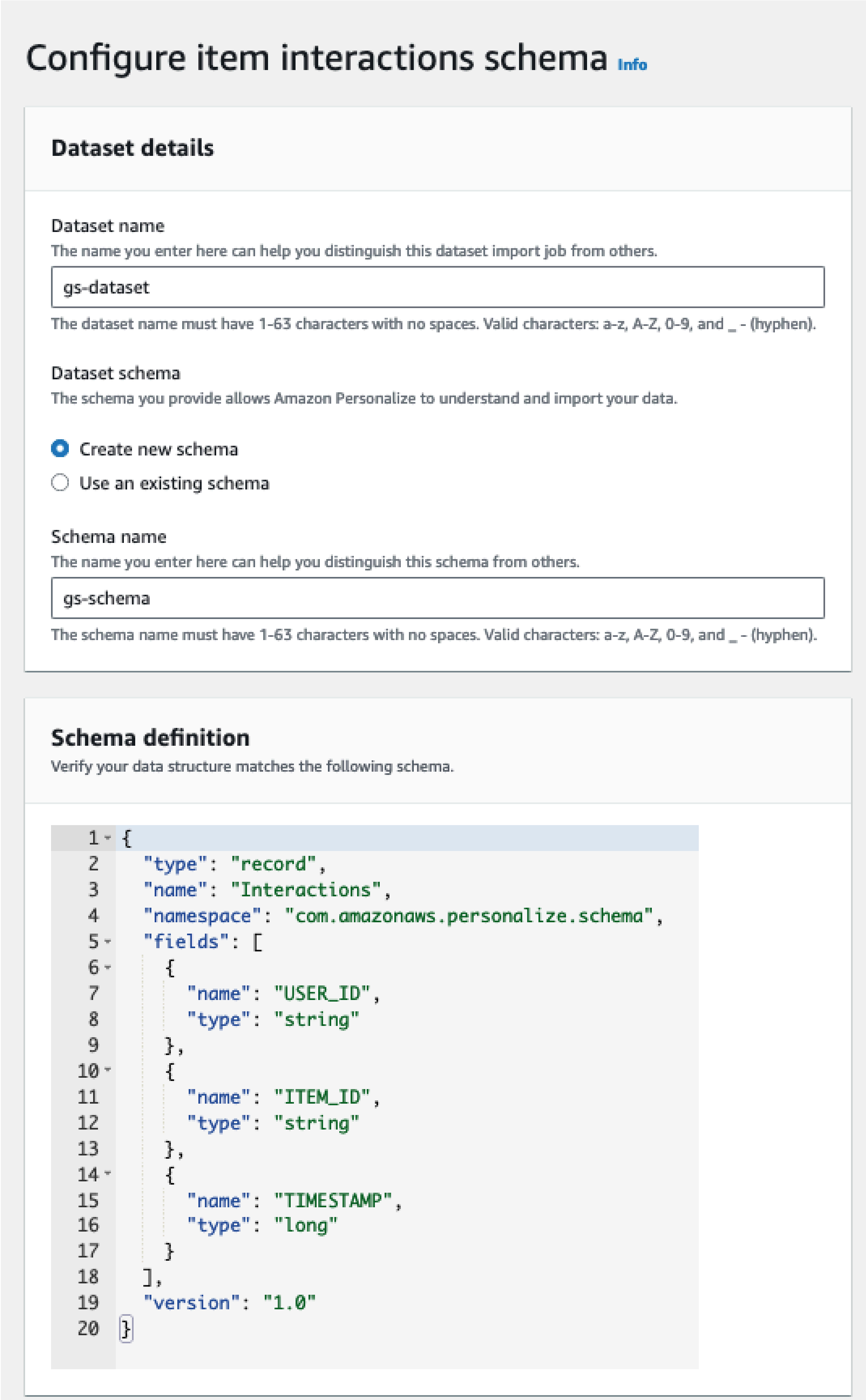
In this procedure, you first create a dataset group. Next, you create an Amazon Personalize *Item interactions dataset* dataset in the dataset group.

To create a dataset group and a dataset

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. In **Dataset group details**, for **Dataset group name**, specify a name for your dataset group.
4. For **Domain** choose **Custom**. Your screen should look similar to the following:



5. Choose **Create dataset group**. The **Overview** page appears.
6. In **Set up datasets**, choose **Create dataset** and choose **Item interactions dataset**.
7. Choose **Import data directly into Amazon Personalize datasets** and choose **Next**.
8. On the **Configure item interactions dataset** page, for **Dataset name**, specify a name for your dataset.
9. For **Dataset schema**, choose **Create new schema**. In the **Schema fields** section, a minimal `Interactions` schema is displayed. The schema matches the headers you previously added to the `ratings.csv` file, so you don't need to make any changes. If you haven't created the training data, see Getting started prerequisites.
10. For **Schema name**, specify a name for the new schema. Your screen should look similar to the following:


Configure item interactions schema Info
Dataset details
Dataset name
The name you enter here can help you distinguish this dataset import job from others.
gs-dataset
The dataset name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).
Dataset schema
The schema you provide allows Amazon Personalize to understand and import your data.
Create new schema
Use an existing schema
Schema name
The name you enter here can help you distinguish this schema from others.
gs-schema
The schema name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).
Schema definition
Verify your data structure matches the following schema.
{
"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"
}

11. Choose **Next**. The **Configure item interactions dataset import job** page appears. Next, complete Step 2: Import item interactions data to import interactions data.

Step 2: Import item interactions data

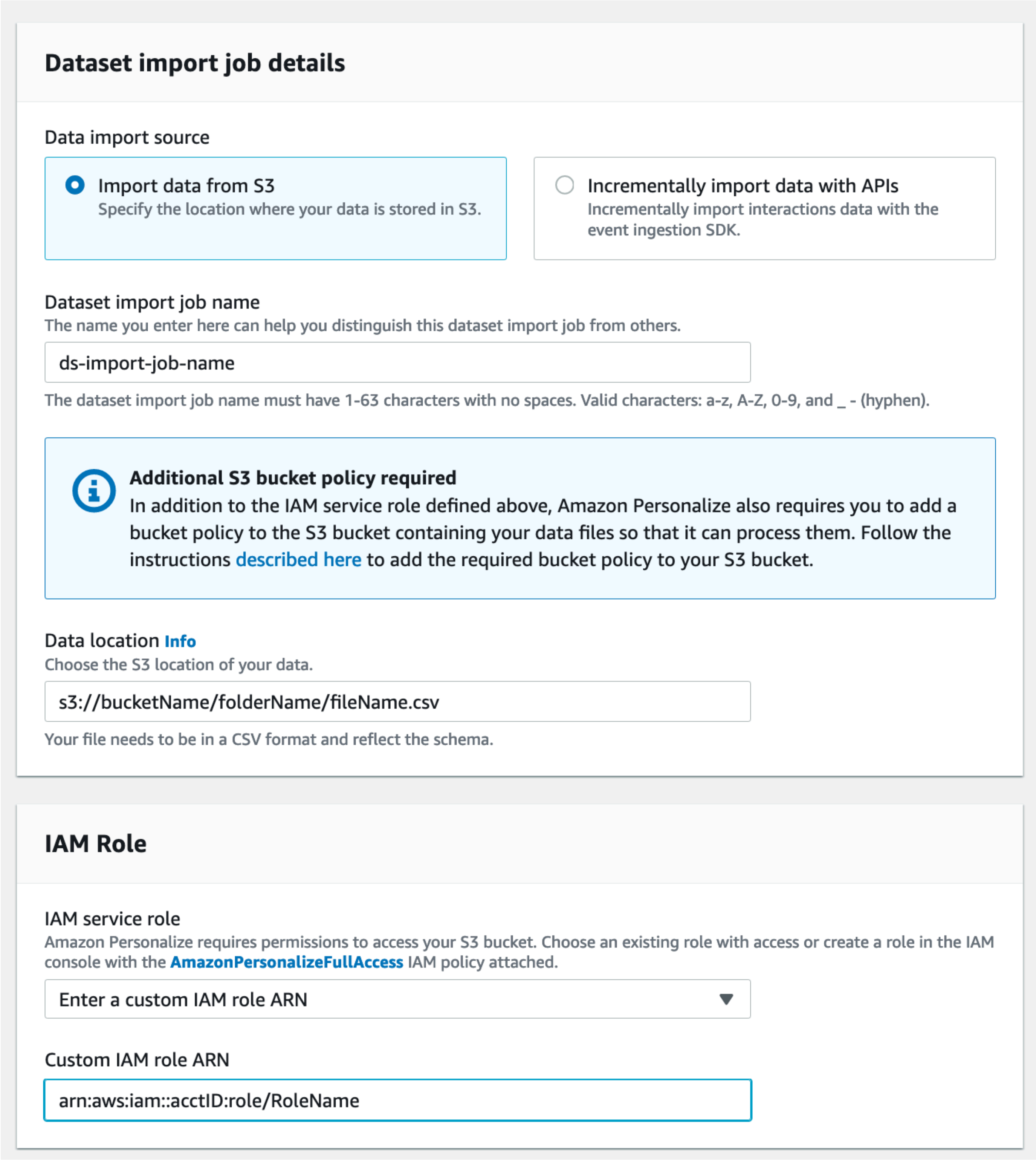
Now that you have created a dataset, it's time to import item interactions data into the dataset.

To import item interactions data

1. On the **Configure item interactions dataset import job** page, for **Data import source** choose **Import data from S3**.
2. For **Dataset import job name**, specify a name for your import job.
3. In the **Additional S3 bucket policy required** dialog box, if you haven't granted Amazon Personalize permissions, follow the instructions to add the required Amazon S3 bucket policy.
4. 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>
   ```
5. In the **IAM Role** section, for **IAM service role**, keep the default selection of **Enter a custom IAM role ARN**.
6. For **Custom IAM role ARN**, specify the role that you created in Creating an IAM role for Amazon Personalize.

   The **Dataset import job details** and **IAM role** sections should be similar to the following:



7. Choose **Finish**. The data import job starts and the **Overview** page is displayed. Initially, the status is **Create pending** (followed by **Create in progress**), and the the **Create solution** button is disabled.

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 status changes to **Active** and the **Create solution** button is enabled.

8. After the import job has finished, choose the **Create solution** button. The **Create solution** page is displayed. Now that you have imported data, you are ready to create a solution in Step 3: Create a solution.

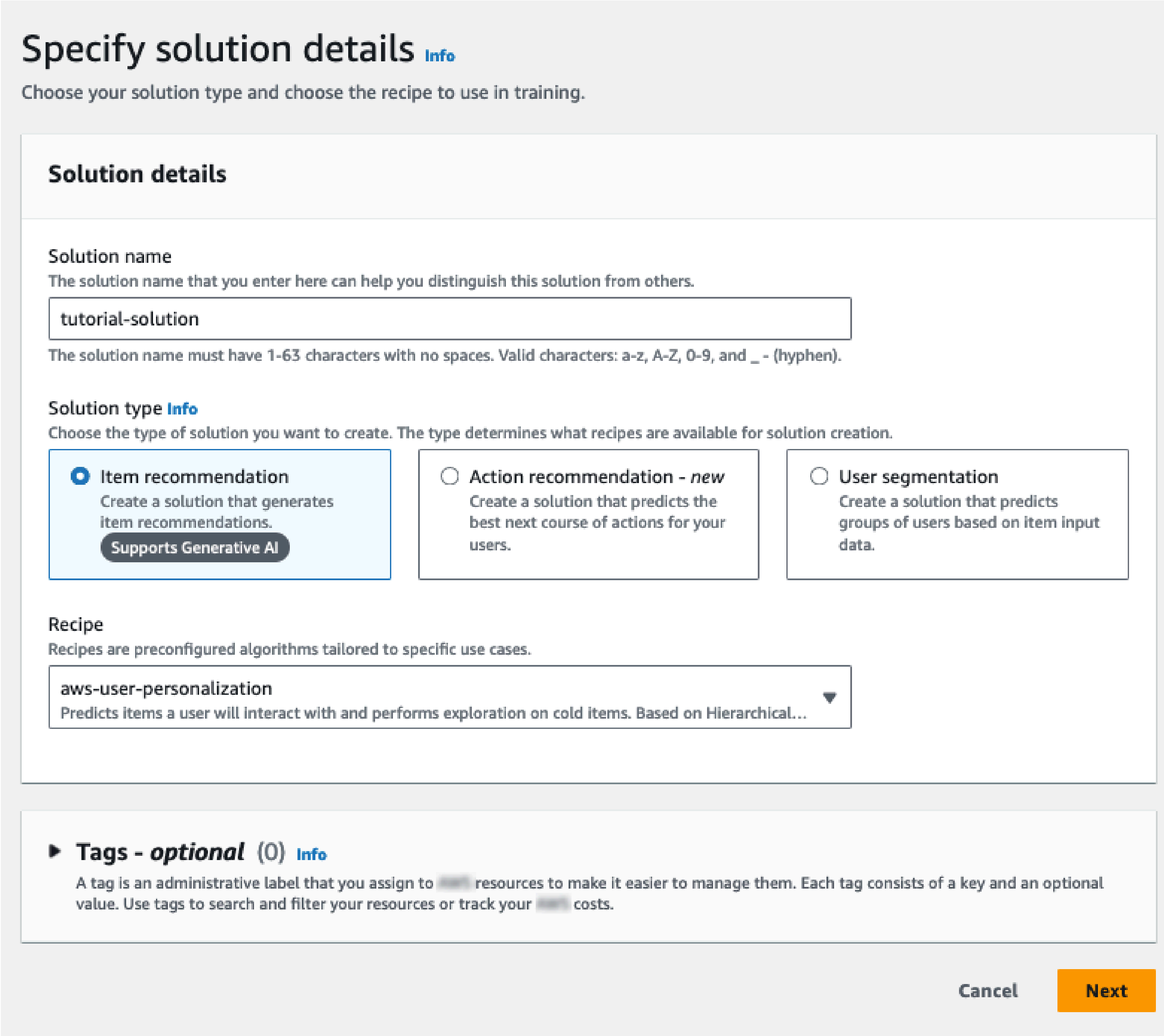
Step 3: Create a solution

In this procedure, you use the dataset that you imported in Step 2: Import item interactions data to train a model. A trained model is referred to as a *solution version*.

To create a solution

1. On the **Overview** page for your dataset group, in **Use custom resources** choose **Create solution**.
2. For **Solution type**, choose **Item recommendation** to get item recommendations for your users.
3. For **Solution name**, specify a name for your solution.
4. For **Solution type** choose **Item recommendations**.
5. For **Recipe**, choose **aws-user-personalization**.

   Your screen should look similar to the following:



6. Choose **Next**. Leave the optional **Advanced configuration** fields unchanged.
7. Choose **Next** and review the details for the solution.
8. Choose **Create solution**. Solution version training starts and the **Overview** page displays.
9. In the navigation pane expand **Custom resources** and choose **Solutions and recipes**.
10. In the **Solutions** section, choose your solution. The details page for the solution page appears. The **Solution versions** page lists the status of your model.

    When the **Solution version status** is *Active*, you are ready to move to [Step 4: Create a campaign](#).

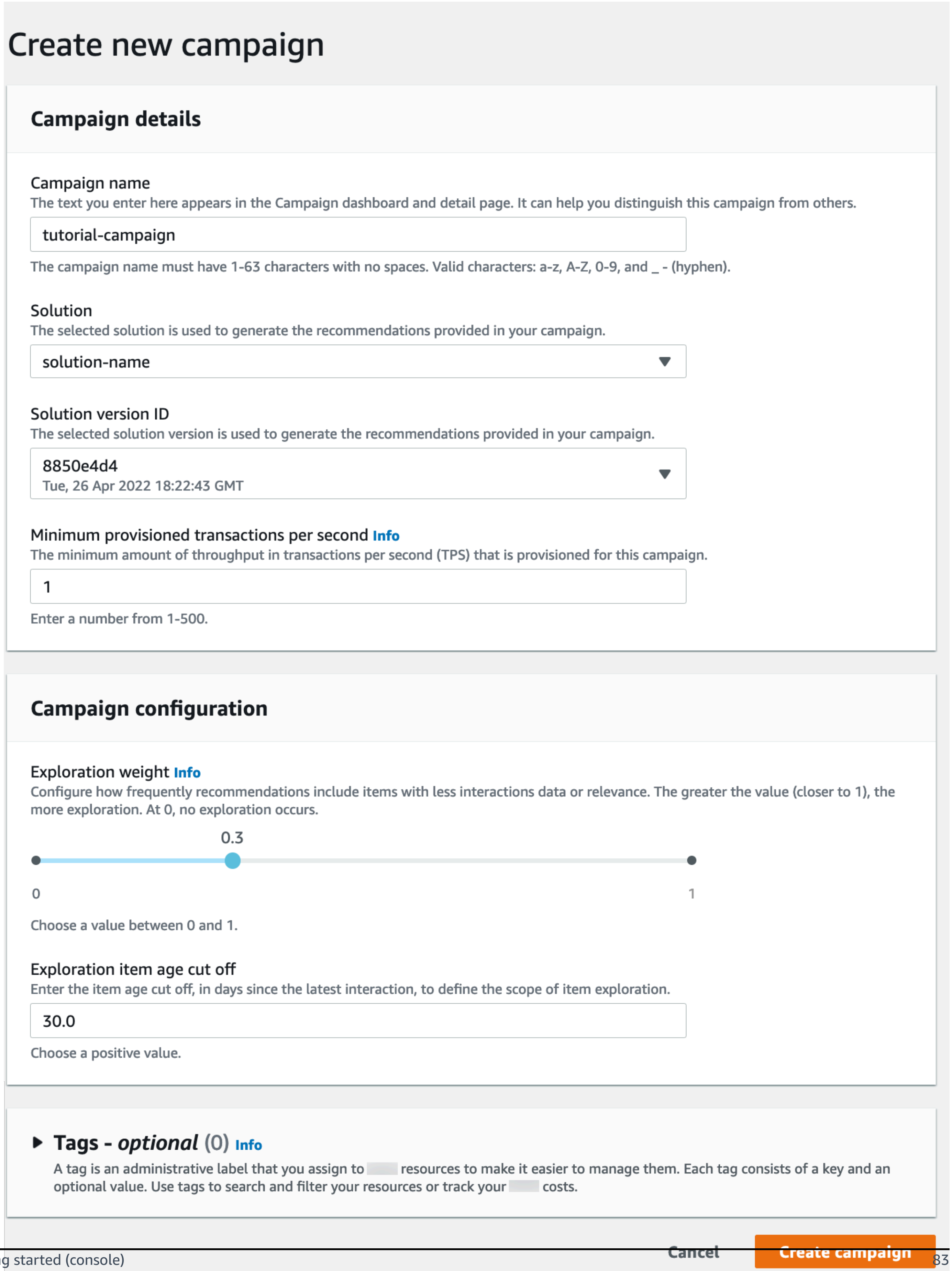
Step 4: Create a campaign

In this procedure, you create a campaign, which deploys the solution version you created in the previous step.

To create a campaign

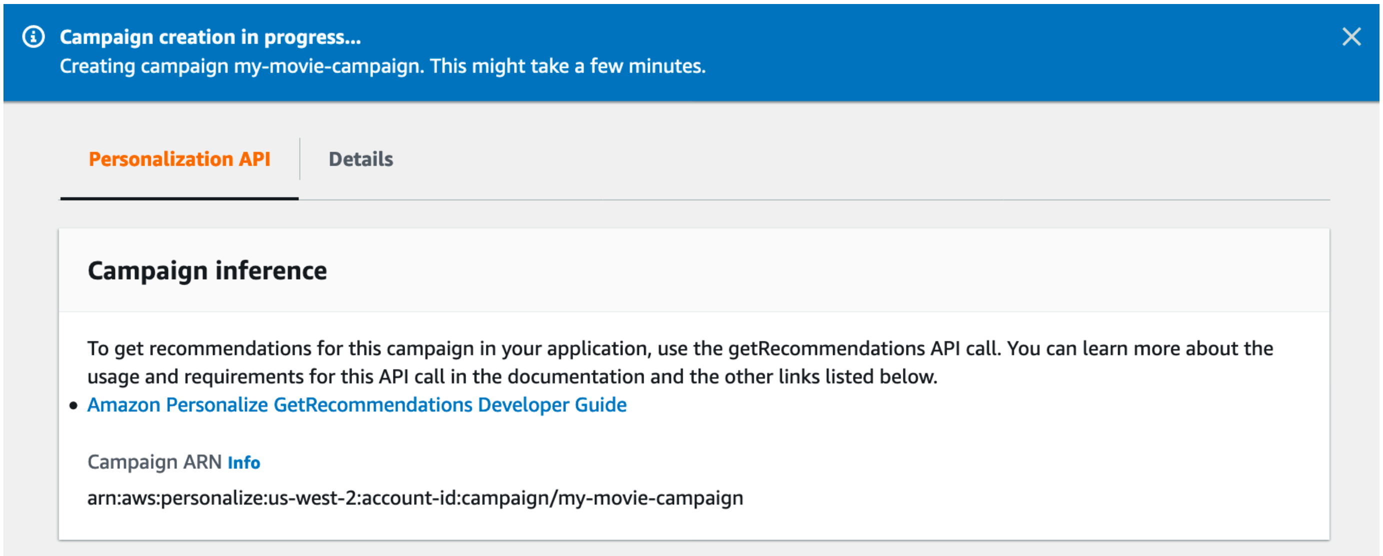
1. In the navigation pane, expand **Custom resources** and choose **Campaigns**.
2. Choose **Create campaign**. The **Create new campaign** page appears.
3. In **Campaign details**, for **Campaign name**, specify a name for your campaign.
4. For **Solution**, choose the solution you created in the previous step and for **Solution version ID** keep the default.
5. 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:

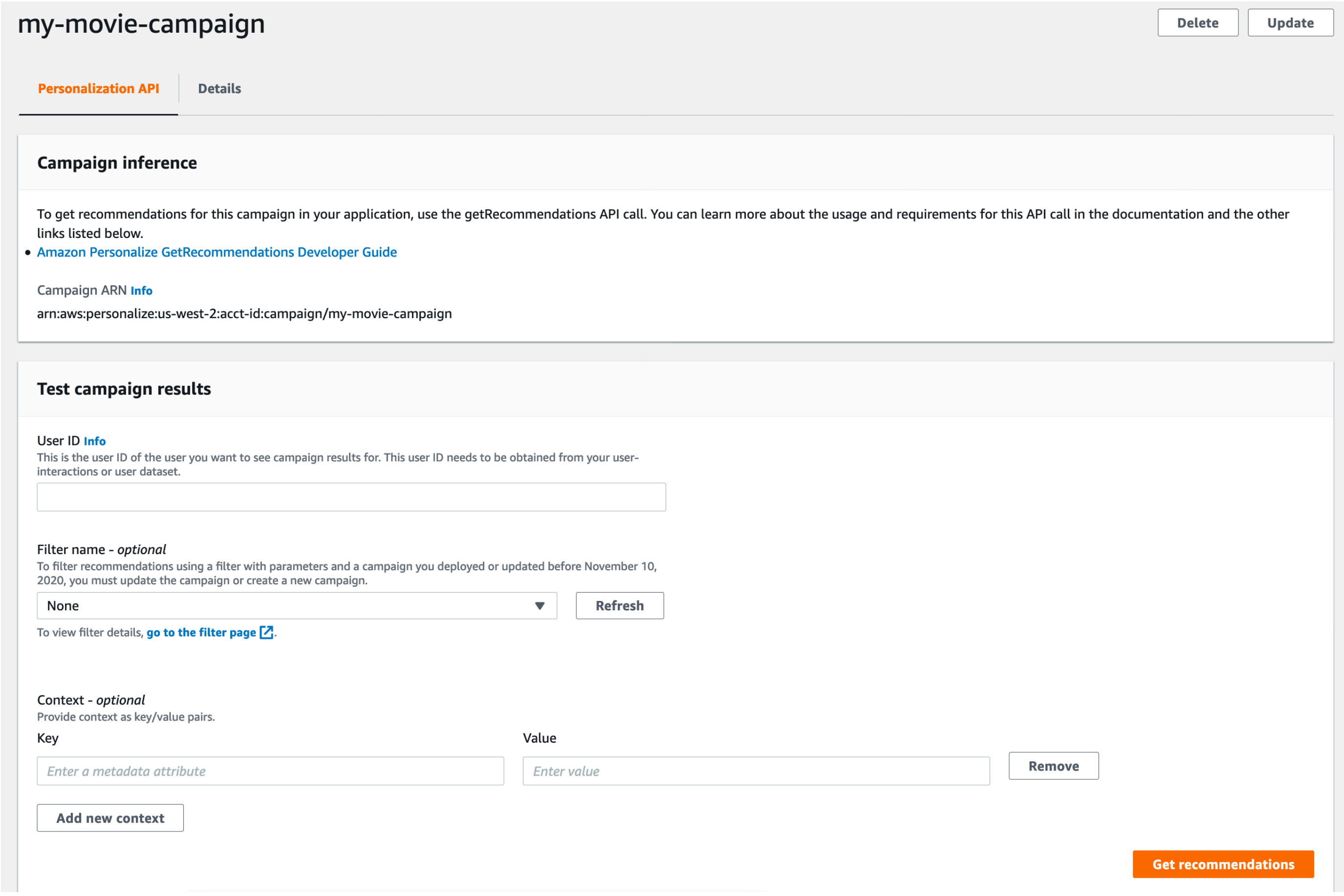

Create new campaign
Campaign details
Campaign name
The text you enter here appears in the Campaign dashboard and detail page. It can help you distinguish this campaign from others.
tutorial-campaign
The campaign name must have 1-63 characters with no spaces. Valid characters: a-z, A-Z, 0-9, and _ - (hyphen).
Solution
The selected solution is used to generate the recommendations provided in your campaign.
solution-name
Solution version ID
The selected solution version is used to generate the recommendations provided in your campaign.
8850e4d4
Tue, 26 Apr 2022 18:22:43 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
0
1
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.
Tags - optional (0) Info
A tag is an administrative label that you assign to resources to make it easier to manage them. Each tag consists of a key and an optional value. Use tags to search and filter your resources or track your costs.
Cancel
Create campaign

6. Choose **Create campaign**. Campaign creation starts and the campaign details pages with the **Personalization API** section displayed.

   Your screen should look similar to the following:

   Creating a campaign can take a couple minutes. 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:



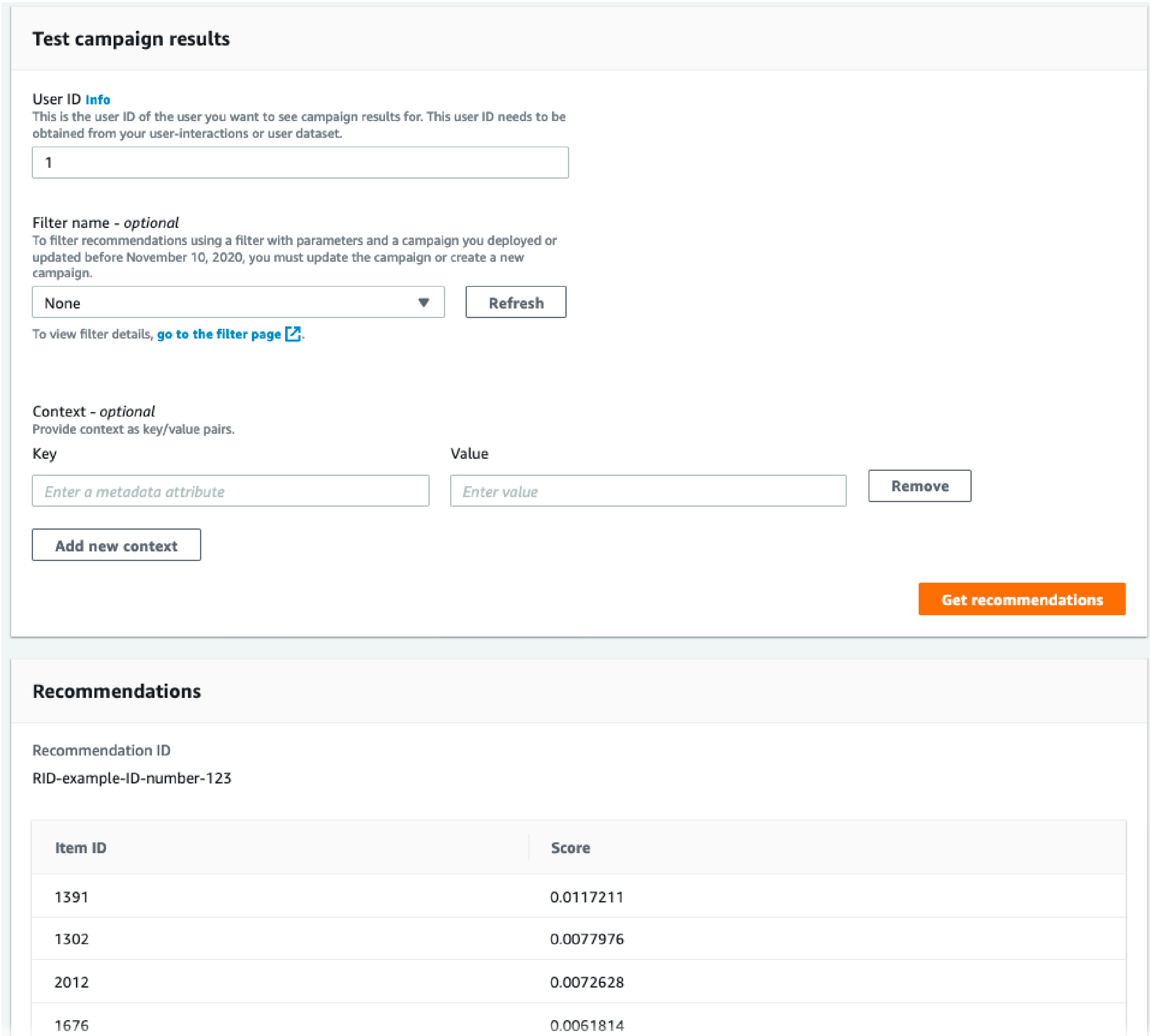
Step 5: Get recommendations

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:



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

When you finish the getting started exercise, to avoid incurring unnecessary charges, follow the steps in Cleaning up resources 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.

```
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 types of datasets recognized by Amazon Personalize. For more information, see [Schemas].

```
{
  "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.

```
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.

```
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 role for Amazon Personalize. For more information about the API, see CreateDatasetImportJob.

   ```
   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.

   ```
   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 operation. Second, you train the model using the CreateSolutionVersion operation.

You train a model using a recipe and your training data. Amazon Personalize provides a set of predefined recipes. For more information, see Choosing a recipe. 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.

```
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.

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

4. 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.

5. You can validate the performance of the solution version by reviewing its metrics. Get the metrics for the solution version by running the following command. Provide the

solution version ARN that was returned previously. For more information about the API, see GetSolutionMetrics.

```
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 API.

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

```
aws personalize create-campaign \
  --name MovieRecommendationCampaign \
  --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.

```
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.

> **Note**
>
> Not all recipes support the `GetRecommendations` API. For more information, see Choosing a recipe.
>
> 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 tutorial shows you how to complete the Amazon Personalize workflow from start to finish with the SDK for Python (Boto3).

To avoid incurring unnecessary charges, when you finish this getting started exercise delete the resources you create in this tutorial. For more information, see Cleaning up resources.

Topics

Prerequisites

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

- Complete the Getting started prerequisites to set up the required permissions and create the training data. If you are using your own source data, make sure that your data is formatted like the prerequisites.
- Set up your AWS SDK for Python (Boto3) environment as specified in Setting up the AWS SDKs.

Tutorial

In the following steps, you verify your environment and create SDK for Python (Boto3) clients for Amazon Personalize. Then you import data, create and deploy a solution version with a campaign, and get recommendations.

Step 1: Verify your Python environment and create boto3 clients

After you complete the prerequisites, run the following Python example to confirm that your environment is configured correctly. This code also creates the Amazon Personalize boto3 clients you use in this tutorial. If your environment is configured correctly, a list of the available recipes is displayed, and you can run the other examples in this tutorial.

```
import boto3
```

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

response = personalize.list_recipes()

for recipe in response['recipes']:
    print (recipe)
```

Step 2: Import data

After you create Amazon Personalize boto3 clients and verify your environment, import the historical data you created when you completed the Getting started prerequisites. To import historical data into Amazon Personalize, do the following:

1. Use the following code to create a schema in Amazon Personalize. Replace `getting-started-schema` with a name for the schema.

```
import json
schema = {
  "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"
}

create_interactions_schema_response = personalize.create_schema(
    name='getting-started-schema',
    schema=json.dumps(schema)
```

```
)

interactions_schema_arn = create_interactions_schema_response['schemaArn']
print(json.dumps(create_interactions_schema_response, indent=2))
```

2. Create a dataset group with the following code. Replace `dataset group name` with a name for the dataset group.

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

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

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

3. Create an Item interactions dataset in your new dataset group with the following code. Give the dataset a name and provide the `schema_arn` and `dataset_group_arn` from the previous steps.

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

dataset_arn = response['datasetArn']
```

4. Import your data with a dataset import job with the following code. The code uses the describe_dataset_import_job method to track the status of the job.

   Pass the following as parameters: a name for the job, the `dataset_arn` from the previous step, the Amazon S3 bucket path (`s3://bucket name/folder name/ratings.csv`) where you stored the training data, and your IAM service role's ARN. You created this role as part of the Getting started prerequisites. Amazon Personalize needs permission to access the bucket. See Giving Amazon Personalize access to Amazon S3 resources.

```
import time
response = personalize.create_dataset_import_job(
```

```
    jobName = 'JobName',
    datasetArn = 'dataset_arn',
    dataSource = {'dataLocation':'s3://bucket/file.csv'},
    roleArn = 'role_arn',
    importMode = 'FULL'
)

dataset_interactions_import_job_arn = response['datasetImportJobArn']

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

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

max_time = time.time() + 3*60*60 # 3 hours
while time.time() < max_time:
    describe_dataset_import_job_response = personalize.describe_dataset_import_job(
        datasetImportJobArn = dataset_interactions_import_job_arn
    )
    status = describe_dataset_import_job_response["datasetImportJob"]['status']
    print("Interactions DatasetImportJob: {}".format(status))

    if status == "ACTIVE" or status == "CREATE FAILED":
        break

    time.sleep(60)
```

Step 3: Create a solution

After importing 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 code. Pass the following as parameters: the `dataset_group_arn` from earlier, a name for the solution, and the ARN for the User-Personalization recipe (`arn:aws:personalize:::recipe/aws-user-personalization`). Store the ARN of your new solution for later use.

```
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)
```

2. Create a solution version with the following code. Pass as a parameter the `solution_arn` from the previous step. The following code creates a solution version. During training, the code uses the [DescribeSolutionVersion](#) 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.

```
import time
import json

create_solution_version_response = personalize.create_solution_version(
    solutionArn = solution_arn
)

solution_version_arn = create_solution_version_response['solutionVersionArn']
print(json.dumps(create_solution_version_response, indent=2))

max_time = time.time() + 3*60*60 # 3 hours
while time.time() < max_time:
    describe_solution_version_response = personalize.describe_solution_version(
        solutionVersionArn = solution_version_arn
    )
    status = describe_solution_version_response["solutionVersion"]["status"]
    print("SolutionVersion: {}".format(status))

    if status == "ACTIVE" or status == "CREATE FAILED":
        break

    time.sleep(60)
```

Step 4: Create a campaign

After you create your solution version, deploy it with an Amazon Personalize campaign. Use the following code to create a campaign that deploys your solution version. Pass the following as parameters: the `solution_version_arn`, and a name for the campaign. The method returns the Amazon Resource Name (ARN) of your new campaign. Store it for later use.

```
response = personalize.create_campaign(
```

```
    name = 'campaign name',
    solutionVersionArn = 'solution version arn'
)

arn = response['campaignArn']

description = personalize.describe_campaign(campaignArn = arn)['campaign']
print('Name: ' + description['name'])
print('ARN: ' + description['campaignArn'])
print('Status: ' + description['status'])
```

Step 5: Get recommendations

After you create a campaign, you can use it to get recommendations. The following code shows how to get recommendations from a campaign and print out each recommended item's ID. Pass the ARN of the campaign you created in the previous step. For user ID, you pass the ID of a user that from the training data, such as 123.

```
response = personalizeRt.get_recommendations(
    campaignArn = 'Campaign ARN',
    userId = '123',
    numResults = 10
)

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

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 to delete the resources you create in the tutorial.

For more examples, see Complete Amazon Personalize project.

Topics

Prerequisites

The following are prerequisite steps for completing this tutorial:

- Complete the Getting started prerequisites, to set up the required permissions and create the training data. You can use the same source data used in the Getting started (console) or Getting started (AWS CLI) exercises. If you are using your own source data, make sure that your data is formatted like in the prerequisites.
- 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*.

Tutorial

In the following steps you set up your project to use Amazon Personalize packages and create Amazon Personalize SDK for Java 2.x clients. Then you import data, create and deploy a solution version with a campaign, and get recommendations.

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;
// 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: Create Amazon Personalize clients

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

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

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. 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.

```
{
```

```
  "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 . Use the following `createDataset` method to create an . 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, a name for the job, your 's ARN, the Amazon S3 bucket path (`s3://`*`bucket name`*`/`*`folder name`*`/ratings.csv`) where you stored the training data, and your service role's ARN (you created this role as part of the Getting started prerequisites). 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 "";
```

```
    }
```

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 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 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 and a solution version. When you create a solution version, you can evaluate its performance before proceeding. For more information, see Evaluating a solution version with metrics.

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 createPersonalCompaign(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());
        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.

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 the Amazon-Personalize-Java-App on GitHub. This project includes training multiple solution versions with different recipes, and recording events with the PutEvents operation.

For additional examples, see code the 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 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.

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?.

Topics

- Cleaning up domain-based resources
- Cleaning up custom resources

Cleaning up domain-based resources

If you created a Domain dataset group, delete resources as follows:

Resource to be deleted	Delete this first	Notes
Recommender		
DatasetImportJob		Can't be deleted.
Dataset		No associated DatasetImportJobs can have a status of CREATE PENDING or IN PROGRESS. No associated Recommenders can have a status of CREATE PENDING or IN PROGRESS.
DatasetSchema	All datasets that reference the schema.	

Resource to be deleted	Delete this first	Notes
DatasetGroup	All associated recommenders All datasets in the dataset group.	

Cleaning up custom resources

If you created a Custom dataset group, delete resources as follows:

Resource to be deleted	Delete this first	Notes
Campaign		
DatasetImportJob		Can not be deleted.
EventTracker		The event-interactions dataset that is associated with the event tracker is not deleted and continues to be used by the solution version.
Dataset		No associated `DatasetImportJob` can have a status of CREATE PENDING or IN PROGRESS. No associated `SolutionVersion` can have a status of CREATE PENDING or IN PROGRESS.
DatasetSchema	All datasets that reference the schema.	
Solution	All campaigns based on the solution version.	No associated `SolutionVersion` can have a status of CREATE PENDING or IN PROGRESS.

Resource to be deleted	Delete this first	Notes
SolutionVersion		Deleted when the associated `Solution` is deleted.
DatasetGroup	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 five types of datasets:

- **Item interactions** – This dataset stores historical and real-time data from interactions between users and items. In Amazon Personalize, an *interaction* is an *event* that you record and then import as training data. For both Domain dataset groups and Custom dataset groups, you must at minimum create an Item interactions dataset.
- **Users** – This dataset stores metadata about your users. This might include information such as age, gender, loyalty membership, or item title.
- **Items** – This dataset stores metadata about your items. This might include information such as price, SKU type, or availability.
- **Actions** – This dataset stores metadata about your actions. An *action* is an engagement activity that you might want to recommend to your customers. Actions might include installing your mobile app, completing a membership profile, joining your loyalty program, or signing up for promotional emails. For the Next-Best-Action recipe, the Actions dataset is required. No other custom recipe or domain use case uses Actions data.
- Action interactions – This dataset stores historical and real-time data from interactions between users and actions. The Next-Best-Action recipe uses this data and the data in your Actions dataset to recommend actions to your users. No other custom recipe or domain use case uses Action-Interactions data.

Each dataset group can have only one of each dataset type. Amazon Personalize stores your data in datasets until you delete the datasets. For all use cases (Domain dataset groups) and recipes (Custom dataset groups), your interactions data must have the following:

- At minimum 1000 item interactions records from users interacting with items in your catalog. These interactions can be from bulk imports, or streamed events, or both.
- At minimum 25 unique user IDs with at least two item interactions for each.

For quality recommendations, we recommend that you have at minimum 50,000 item interactions from at least 1,000 users with two or more item interactions each.

Before you create a dataset, you define a schema for that dataset. 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. After you create a schema, you can't make changes to the schema.

For Domain dataset groups, each dataset type has a default schema with required fields and reserved keywords. Each time you create a dataset, you can either use the existing domain schema or create a new one by modifying the existing default schema. Use the default schema as a guide for what data to import for your domain. Once you define the schema and create the dataset, you can't make changes to the schema.

If you import data in bulk, your data must be stored in comma-separated values (CSV) format. The first row of your CSV file must contain column headers, which must match your schema. For information about how to format your bulk data for Amazon Personalize, Data format guidelines.

Topics

- Datasets
- Schemas
- Data format guidelines

Datasets

The following topics provide detailed information on Amazon Personalize datasets. Each type of dataset has different data requirements. For both Domain dataset groups and Custom dataset groups, your interactions data must have the following before training:

- At minimum 1000 item interactions records from users interacting with items in your catalog. These interactions can be from bulk imports, or streamed events, or both.
- At minimum 25 unique user IDs with at least two item interactions for each.

For quality recommendations, we recommend that you have at minimum 50,000 item interactions from at least 1,000 users with two or more item interactions each.

If you create a Domain dataset group, each dataset has additional requirements depending on domain. If you aren't sure what type of data you need, we recommend creating a Domain dataset group and using the default schemas for your domain as a guide. For more information about dataset and schema requirements see Schemas.

Topics

Item interactions dataset

An *item interaction* is an interaction event between a user and an item in your catalogue. For example, a user watching a movie, viewing a listing, or purchasing a pair of shoes. You import data about your users' interactions with your items into a *Item interactions dataset*. 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, you can have Amazon Personalize 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 item interaction events to an *Item interactions dataset*.

For all use cases (Domain dataset groups) and recipes (custom resources), your item interactions data must have the following:

- At minimum 1000 item interactions records from users interacting with items in your catalog. These interactions can be from bulk imports, or streamed events, or both.
- At minimum 25 unique user IDs with at least two item interactions for each.

For quality recommendations, we recommend that you have at minimum 50,000 item interactions from at least 1,000 users with two or more item interactions each.

To create a recommender or a custom solution, you must at minimum create an *Item interactions dataset*. This section provides information about the following types of item interactions data you can import into Amazon Personalize.

Topics

Event type and event value data

An Item interactions dataset can store event type and event value data for each interaction. Only custom resources use event value data.

Event type data

Amazon Personalize uses event type data, such as *click* or *purchase* data, to identify user intent and interest. If you create domain recommenders, all use cases require event type data. Some use cases require specific event types. You are free to use additional event types. For more information see Choosing a use case.

If you create custom resources, you can choose the events used for training by event type. If your dataset has multiple event types in an EVENT_TYPE column, and you do not provide an event type when you configure a custom solution, Amazon Personalize uses all item interactions data for training with equal weight regardless of type. For more information, see Choosing the item interaction data used for training.

Positive and negative event types

Amazon Personalize assumes any interaction is a positive one. Interactions with a negative event type, such as *dislike*, won't necessarily keep the item from appearing in the user's future recommendations.

The following are ways to have negative events and users' disinterest influence recommendations:

- For all domain use cases and the User-Personalization recipe, Amazon Personalize can use impressions data. When an item appears in impressions data and a user doesn't choose it, the item is less likely to appear in recommendations. For more information, see Impressions data.
- If you use custom resources and import positive and negative event types, you can train on only positive event types and then filter out items the user interacted with negatively. For more information, see Choosing the item interaction data used for training and Filtering recommendations and user segments.

Event value data (custom resources)

Event value data might be the percentage of a movie that a user watched or a rating out of 10. If you create custom solutions and import event value data along with event type data, you can choose records used for training based on type and value. With domain recommenders, Amazon Personalize doesn't use event value data and you can't filter events before training.

To choose records based on type and value, record an event type and event value for each event. The value you choose for each event depends on what data you want to exclude and what event types you are recording. For example, you might match the user activity, such as the percentage of video the user watched for *watch* event types.

When you configure a solution, you 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.

For more information, see Choosing the item interaction data used for training

Contextual metadata

With certain recipes and recommender use cases, Amazon Personalize can use contextual metadata when identifying underlying patterns that reveal the most relevant items for your users. Contextual metadata is interactions data you collect on the user's environment at the time of an event, such as their location or device type.

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 Domain dataset groups, the following recommender use cases can use contextual metadata:

- Recommended for you (ECOMMERCE domain)
- Top picks for you (VIDEO_ON_DEMAND domain)

For custom resources, recipes that use contextual metadata include the following:

- User-Personalization
- Personalized-Ranking

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 create a Domain dataset group for the VIDEO_ON_DEMAND or ECOMMERCE domain, or use the User-Personalization recipe, Amazon Personalize can model impressions data that you upload to an *Item 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. With exploration, recommendations include some items or actions that would be typically less likely to be recommended for the user, such as new items or actions, items or actions with few interactions, or items or actions less relevant for the user based on their previous behavior. The more frequently an item occurs in impressions data, the less likely it is that Amazon Personalize includes the item in exploration. Impression values can have at most 1000 characters (including the vertical bar character).

For more information about exploration see Exploration. Amazon Personalize can model two types of impressions: Implicit impressions and Explicit impressions.

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 and GetPersonalizedRanking operations) as input for future PutEvents 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” 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 API and include the `recommendationId` as a parameter. For a code sample see Recording impressions data.
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.

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” 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 API and list the recommended items that are in stock in the `impression` parameter. For a code sample see Recording impressions data.

   For importing impressions in historical item interactions data, you can list explicit impressions in your csv file and separate each item with a '|' character. The vertical bar character counts towards the 1000 character limit. For an example see Formatting explicit impressions.
5. Amazon Personalize uses the impression data to guide exploration, where future recommendations include new shoes with less interactions data or relevance.

Users dataset

The user data that you can import into Amazon Personalize includes numerical and categorical metadata about your users, such as gender or loyalty membership. You import metadata about your users into an Amazon Personalize *Users dataset*. The maximum number of metadata columns is 25.

This topic provides information about the following types of user data:

Topics

Categorical metadata

With some recipes and both VIDEO_ON_DEMAND and ECOMMERCE domains, 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. You define your own range of values based on your use case. Categorical metadata can be in any language.

With all recipes and domains, 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 and user segments.

Categorical values can have at most 1000 characters. If you have a user with a categorical value with more than 1000 characters, your dataset import job will fail.

For Custom dataset groups and custom solutions, recipes that use categorical metadata include the following:

- User-Personalization
- Personalized-Ranking
- Similar-Items

Items dataset

The item data that you can import into Amazon Personalize includes numerical and categorical metadata such as creation timestamp, price, genre, description, and availability. You import metadata about your items into an Amazon Personalize *Items dataset*.

Amazon Personalize doesn't use non-categorical string item data, such as item titles or author data when training. However, some Amazon Personalize features do use this data to enhance recommendations. For more information, see Non-categorical string data

The maximum number of metadata columns is 100. The maximum number of items that are considered by a model during training is 750,000. Amazon Personalize only considers these items when generating recommendations. Some domains and recipes require an Items dataset. For more information on recipe requirements see Choosing a recipe.

This topic provides information about the following types of item data:

Topics

- Creation timestamp data
- Categorical metadata
- Unstructured text metadata
- Non-categorical string data

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 certain recipes and domains, 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. You define your own range of values based on your use case. Categorical metadata can be in any language.

With all recipes and domains, 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 and user segments.

Categorical values can have a maximum of 1000 characters. If you have an item with a categorical value with more than 1000 characters, your dataset import job will fail.

For Domain dataset groups, both VIDEO_ON_DEMAND and ECOMMERCE domains use categorical metadata. For Custom dataset groups and custom solutions, recipes that use categorical metadata include the following:

- User-Personalization
- Personalized-Ranking
- Similar-Items
- Item-Affinity
- Item-Attribute-Affinity

Unstructured text metadata

With certain recipes and domains, 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 individual 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. You can add at most 1 textual field. For an example of an Items schema with a field for unstructured text data, see Items dataset schema example (custom). Amazon Personalize truncates text fields at the character limit. Make sure that the most relevant information in the text is at the start of the field. For information about importing data into Amazon Personalize, see Step 2: Preparing and importing data.

Before using unstructured text values, Amazon Personalize removes the following from the text:

- HTML and XML tags and entities
- New line, tab, and extra space characters

Unstructured text values can have at most 20,000 characters in all languages except Chinese and Japanese. For Chinese and Japanese, you can have at most 7,000 characters. Amazon Personalize truncates values that exceed the character limit to the character limit.

Text can be in the following languages:

- Chinese (Simplified)
- Chinese (Traditional)
- English
- French
- German
- Japanese
- Portuguese
- Spanish

You can submit unstructured text items in multiple languages, but each item's text should be in only one language.

For Domain dataset groups, both VIDEO_ON_DEMAND and ECOMMERCE domains use textual metadata. For Custom dataset groups and custom solutions, recipes that use textual metadata include the following:

- User-Personalization
- Personalized-Ranking
- Similar-Items
- Item-Affinity
- Item-Attribute-Affinity

Non-categorical string data

Except for item IDs, Amazon Personalize doesn't use non-categorical string data when training, such as item titles or author data. However, Amazon Personalize can use it with the following features:

- Amazon Personalize can include item metadata in recommendations, including non-categorical string values. You might use metadata to enrich recommendations in your user interface, such as adding the director's name to a movie recommendations carousel. For more information, see Metadata with recommendations.

- If you use Similar-Items, you can generate batch recommendations with themes. When you generate batch recommendations with themes, you must specify an item name column in the batch inference job. For more information, see Batch recommendations with themes from Content Generator.
- You can create filters to include or remove items from recommendations based on non-categorical string data. For more information about filters, see Filtering recommendations and user segments.

Actions dataset

An *action* is an engagement or revenue generating activity that you might want to recommend to your users. Actions might include installing your mobile app, completing a membership profile, joining your loyalty program, or signing up for promotional emails. You import data about your actions into an Amazon Personalize *Actions* dataset. Examples of data for an action include an ID for the action, the action's estimated value, or the action's expiration timestamp.

During model training, Amazon Personalize considers a maximum of 1000 actions. If you import more than 1000 actions, Amazon Personalize decides which actions to include in training, with priority given to new actions (actions you recently added with no interactions) and existing actions with recent interactions data.

The maximum number of columns is 10. This topic provides information about the following types of actions data:

Topics

- Action expiration timestamp data
- Repeat frequency data
- Value data
- Creation timestamp data
- Categorical metadata

Action expiration timestamp data

An action expiration timestamp specifies the date at which an action is no longer valid. You provide action expiration timestamp data in Unix epoch time format, in seconds. If an action has expired, Amazon Personalize won't include it in recommendations.

Specify an action expiration timestamp for your actions if you want to limit their appearance in recommendations to a certain time frame. For example, you might have an application that is running a membership drive through a certain month. You might set an expiration timestamp for the *enroll* action for the end of that month. Amazon Personalize automatically stops recommending this action when this date is reached.

If you set the expiration timestamp to a time in the past for a new action, or if you update an actions timestamp to a time in the past, it can take up to 2 hours to remove the action from recommendations.

Repeat frequency data

Repeat frequency data specifies how many days Amazon Personalize should wait to recommend a particular action after a user interacts with it, based on the user's history in your Action interactions dataset. You specify an action's repeat frequency in days, with a maximum of 30.

For example, you might have an ecommerce application where each user creates an account and a profile. If you have a `complete profile` action and you want to wait a week after a user interacts with it before recommending it again, you would specify 7 days as the action's `REPEAT_FREQENCY`. After 7 days, Amazon Personalize starts considering the action for recommendations.

If you don't provide a repeat frequency for an action, Amazon Personalize will not set any limits on the number of times it appears in recommendations.

Value data

Value data is the business value or importance of each action. An action's `value` can be 1 – 10, where 10 is the most valuable action in your dataset.

For example, you might have two actions, one for enrolling in your basic subscription and one for enrolling in your premium service. For the basic service, you might specify a value of `5` and for the premium, a value of `10`.

Amazon Personalize uses value data as one input when determining the best action to recommend to your users. For example, if a user is equally likely to take one action or another, Amazon Personalize ranks the action with the highest value higher in recommendations.

Creation timestamp data

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

If you don't have creation timestamp data, Amazon Personalize infers this information from any action interaction data. It uses the timestamp of the action's oldest interaction data as the action's creation timestamp. If an action 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 action.

Categorical metadata

Amazon Personalize uses categorical metadata about actions, such as seasonality or action exclusivity, when identifying the underlying patterns that reveal the best actions for your users. You define your own range of values based on your use case. Categorical metadata can be in any language.

You can import categorical data and use it to filter recommendations based on an action's attributes. For information about filtering recommendations, see Filtering recommendations and user segments.

Categorical values can have a maximum of 1000 characters. If you have an action with a categorical value with more than 1000 characters, your dataset import job will fail.

Action interactions dataset

An *action interaction* is an interaction involving a user and an action in your Actions dataset. You import action interactions into an Amazon Personalize *Action interactions dataset*. Each action interaction consists of a userID, actionID, timestamp, event type, and any additional data about the interaction, such as categorical metadata.

For example, if you have an *enroll* action in your Actions dataset, and a user takes this action, you would record the user's ID, the action's ID, the timestamp, and for event type, record `TAKEN`. You can import action interaction events in bulk with a dataset import job, or you can stream them in real time with the PutActionInteractions API operation. For more information about importing data, see Step 2: Preparing and importing data.

When you use a PERSONALIZED_ACTIONS custom recipe, Amazon Personalize uses any data in your Action interactions dataset as input to predict the actions your users will most likely take. There is no minimum requirement for action interactions data. We recommend that you import it for quality action recommendations. If you don't have action interaction data, you can create an empty Action interactions dataset and record your customers' interactions with actions by using the PutActionInteractions API operation.

The following topics provide more information about the action interaction data that Amazon Personalize can use.

Topics

- Event type data

Event type data

Amazon Personalize can use patterns in event type data to identify the actions your users will most likely take. For example, if a customer frequently ignores an email subscription action (indicated with the NOT_TAKEN event type), Amazon Personalize might adjust recommendations to feature fewer of this type of action.

You can use only the following event types for action interaction events. Amazon Personalize uses these events to learn about your user and calculate what actions to recommend next.

- Taken – Record *Taken* events when a user takes a recommended action.
- Not Taken – Record *Not Taken* events when your user makes a deliberate choice to not take the action after viewing it. For example, if they choose *No* when you show them the action. *Not Taken* events can indicate the customer isn’t interested in the action.
- Viewed – Record *Viewed* events when you show a user an action before they make a choice to take or not take an action. Amazon Personalize uses *View* events to learn about your users' interests. For example, if a user views an action but doesn't take it, this user might not be interested in this action in the future.

Schemas

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. After you create a schema, you can't make changes to the schema.

For Domain dataset groups, each dataset type has a default schema with required fields and reserved keywords. Each time you create a dataset, you can either use the existing domain schema or create a new one by modifying the existing default schema. Use the default schema as a guide for what data to import for your domain. Once you define the schema and create the dataset, you can't make changes to the schema.

Topics

Schema formatting requirements

When you create a schema for either dataset in a Domain dataset group or Custom dataset group, 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.
- The schema fields can appear in any order, but they must match the order of the corresponding column headers in your CSV file.
- Schemas must be flat JSON files without nested structures. For example, a field cannot be the parent of multiple sub-fields.
- Amazon Personalize schemas don't support complex types such as arrays and maps.
- Schema fields must have unique alphanumeric names. For example, you can't add both a `GENRES_FIELD_1` field and a `GENRESFIELD1` field.
- You must define required fields as their required data types. Reserved categorical string fields must have the `categorical` attribute set to `true`, while reserved string fields can't be categorical. The keywords can't be in your data.

- If you add your own metadata field of type `string` and you want Amazon Personalize to use it when training, it must include the `categorical` attribute or the `textual` attribute (only Items schemas support fields with the textual attribute).
- Amazon Personalize can use non-categorical string columns, such as item name columns, when generating themes, returning metadata in recommendations, and filtering recommendations. For more information, see Non-categorical string data.
- Amazon Personalize doesn't use `boolean` type data when training or filtering recommendations. To have Amazon Personalize use boolean data when training or filtering, use 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`.
- Textual fields must be of the type `string` and must have the `textual` attribute set to `true`. For more information about unstructured text data, see Unstructured text metadata.

Domain dataset group datasets have additional requirements based on both domain and dataset type. Custom dataset group datasets have additional requirements depending on type.

Schema data types

Amazon Personalize schemas support the following Avro types for fields:

- float
- double
- int
- long
- string
- boolean
- null

Some required and reserved fields support null data. Adding a `null` type to a field allows you to use imperfect data (for example, metadata with blank values) to generate recommendations. For information on which fields support null data, see Domain datasets and schemas or Custom datasets and schemas. The following example shows how to add a null type for a GENDER field.

```
{
  "name": "GENDER",
```

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

Domain datasets and schemas

When you create a Domain dataset group, the domain you choose determines your dataset and schema requirements. Each domain has a default schema for each dataset type.

When you create a dataset, you can either use the default schema or create a new one based on the default schema. Use the default schema as a guide for what data to collect and import into each dataset type. The following topics explain dataset and schema requirements for each domain.

For information on the types of data you can import into Amazon Personalize see Types of data Amazon Personalize can use.

For information about general Amazon Personalize schema requirements, such as formatting requirements and available field data types, see Schemas. These requirements apply to all schemas, regardless of domain.

Topics

VIDEO_ON_DEMAND datasets and schemas

When you create a Domain dataset group for the VIDEO_ON_DEMAND domain, each dataset type has a default schema with a set of VIDEO_ON_DEMAND specific required and recommended fields. You can either use the default schema or create a new one based on the default schema. The data you import must match your schema in format and type. Use the default domain schemas listed in the sections below as a guide to determine what data to import to create your VIDEO_ON_DEMAND-based recommender.

You are free to add additional fields. As long as the fields aren't listed as required or reserved, and the data types are listed in Schema data types, the field names and data types are up to you.

For information about general Amazon Personalize schema requirements, such as formatting requirements and available field data types, see Schemas. These requirements apply to all schemas, regardless of domain.

The following topics provide information about each dataset's required and recommended fields for the VIDEO_ON_DEMAND domain. Each dataset section includes the default VIDEO_ON_DEMAND schema in JSON format.

Topics

VIDEO_ON_DEMAND domain dataset and schema requirements

Each dataset type has the following required fields and 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. Reserved categorical string fields must have `categorical` set to `true`, while reserved string fields can't be categorical. The keywords can't be in your data.

Dataset type	Required fields	Reserved keywords
Item interactions (default schema)	USER_ID (`string`) ITEM_ID (`string`) TIMESTAMP (`long`) EVENT_TYPE (`string` and depending on use case, `Watch` and `Click` event types)	EVENT_VALUE (`float`, `null`) IMPRESSION (`string`, `null`) RECOMMENDATION_ID (`string`, `null`) EVENT_ATTRIBUTION_SOURCE (`string`, `null`)
Users (default schema)	USER_ID (`string`)	SUBSCRIPTION_MODEL (categorical `string`, `null`)

Dataset type	Required fields	Reserved keywords
	1 metadata field (categorical `string` or numerical)	
Items ([default schema]())	ITEM_ID (`string`) CREATION_TIMESTAMP (`long`) GENRES (categorical `string`)	PRICE (`float`, `null`) DURATION (`float`, `null`) GENRE_L2 (categorical `string`, `null`) GENRE_L3 (categorical `string`, `null`) AVERAGE_RATING (`float`, `null`) PRODUCT_DESCRIPTION (textual `string`, `null`) CONTENT_OWNER (categorical `string`, `null`) CONTENT_CLASSIFICATION (categorical `string`, `null`)

Item interactions dataset requirements (VIDEO_ON_DEMAND domain)

An *Item interactions dataset* stores historical and real-time data from interactions between users and items in your VIDEO_ON_DEMAND catalog. For more information about the types of data you can store in an interactions dataset, see [Item interactions dataset]().

You must have an Item interactions dataset for all use cases and your schema must have the following fields:

- USER_ID (`string`)
- ITEM_ID `string`
- TIMESTAMP (`long`)

- EVENT_TYPE (`string` and depending on use case, `Watch` and `Click` event types)

Your schema can also include the following reserved keywords:

- EVENT_VALUE (`float`, `null`)
- IMPRESSION (`string`, `null`)
- RECOMMENDATION_ID (`string`, `null`)

You are free to add additional fields depending on your use case and your data. As long as the fields aren't listed as required or reserved, and the data types are listed in Schema data types, the field names and data types are up to you. For an example of the default schema for Item interactions datasets for VIDEO_ON_DEMAND domains, see Default Interactions schema (VIDEO_ON_DEMAND domain).

Optionally add the reserved keyword EVENT_VALUE if you have value data for events, such as the percentage of a video watched. Optionally add the reserved keyword IMPRESSION if you want to include explicit and implicit impressions data. For more information about recording impressions data see Impressions data.

The maximum total number of optional metadata fields you can add to an Item interactions dataset, combined with total number of *distinct* event types in your Item interaction 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. If you have an interaction with a categorical value with more than 1000, your dataset import job will fail.

For more information on minimum requirements and maximum data limits for an Item interactions dataset for the VIDEO_ON_DEMAND domain, see Service quotas. For information about general Amazon Personalize schema requirements, such as formatting requirements and available field data types, see Schemas. These requirements apply to all schemas, regardless of domain.

Default Interactions schema (VIDEO_ON_DEMAND domain)

The following is the default VIDEO_ON_DEMAND domain schema for Item interactions datasets.

```
{
```

```
  "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": "TIMESTAMP",
          "type": "long"
      }
  ],
  "version": "1.0"
}
```

Users dataset requirements (VIDEO_ON_DEMAND domain)

A *Users dataset* stores metadata about your users. This might include information such as age, gender, and loyalty membership for each item. For information on the types of user data you can import into Amazon Personalize, see Users dataset. For information about general Amazon Personalize schema requirements, such as formatting requirements and available field data types, see Schemas. These requirements apply to all schemas, regardless of domain.

A Users dataset is optional for all VIDEO_ON_DEMAND use cases. If you have user data, we recommend creating one to get the most relevant recommendations. If you create a Users dataset, your schema must include the following fields.

- USER_ID
- 1 metadata field (categorical `string` or numerical)

You are free to add additional fields depending on your use case and your data. As long as the fields aren't listed as required or reserved, and the data types are listed in Schema data types, the field names and data types are up to you. For an example of the default schema for Users datasets for VIDEO_ON_DEMAND domains, see Default Users schema (VIDEO_ON_DEMAND domain).

A `SUBSCRIPTION_MODEL` field is included in the default schema. This field is an optional reserved keyword and must have a type of `string` with categorical set to `true`. To get the best recommendations, we recommend that you keep this field in your schema if you have subscription model information about each of your users in your data. The data you import must match your schema.

Using categorical data

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 individual item imports. For users with multiple categories, separate each value using the vertical bar, '|'. For example, for a SUBSCRIPTION_MODEL field, your data for a user might be student|monthly|discount.

Categorical values can have at most 1000 characters. If you have a user with a categorical value with more than 1000 characters, your dataset import job will fail.

Default Users schema (VIDEO_ON_DEMAND domain)

The following is the default VIDEO_ON_DEMAND domain schema for Users datasets.

```
{

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

Items dataset requirements (VIDEO_ON_DEMAND domain)

An *Items dataset* stores metadata about your items in your catalogue. This might include information such as price, genre, and availability for each item. For information about the types of item data you can import into Amazon Personalize, see Items dataset. For information about general Amazon Personalize schema requirements, such as formatting requirements and available field data types, see Schemas. These requirements apply to all schemas, regardless of domain.

An Items dataset is required for some use cases (see VIDEO_ON_DEMAND use cases). When optional, we still recommend creating one to get the most relevant recommendations. If you create an Items dataset, your schema must include the following fields:

- ITEM_ID
- GENRES (categorical `string`)
- CREATION_TIMESTAMP (in Unix epoch time format)

Your schema can also include the following reserved keywords. Each keyword lists its required data type and whether it supports null data. Adding the null type is optional.

- PRICE (float)
- DURATION (float)
- GENRE_L2 (categorical `string`, `null`)
- GENRE_L3 (categorical `string`, `null`)
- AVERAGE_RATING (`float`, `null`)
- PRODUCT_DESCRIPTION (textual `string`, `null`)
- CONTENT_OWNER (categorical `string`, `null`): The company that owns the video. For example, values might be HBO, Paramount, and NBC.
- CONTENT_CLASSIFICATION (categorical `string`, `null`): The content's rating. For example, values might be G, PG, PG-13, R, NC-17, and unrated.

To get the best recommendations, we recommend that you keep these as many of these fields in your schema as you have data. The data you import must match your schema. The maximum number of metadata columns is 100. You are free to add additional fields depending on your use case and your data. As long as the fields aren't listed as required or reserved, and the data types are listed in Schema data types, the field names and data types are up to you.

Use reserved keywords GENRE_L2 and GENRE_L3 for items with multiple multi-level categories. For more information, see Using categorical data. For information on textual and categorical metadata see Items dataset. For an example of the default schema for Items datasets for ECOMMERCE domains, see Default Items schema (VIDEO_ON_DEMAND domain).

Using categorical data

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 individual item imports. Categorical values can have at most 1000 characters. If you have an item with a categorical value with more than 1000 characters, your dataset import job will fail.

For items with multiple categories, separate each value with the vertical bar, '|'. For example, for a GENRES field your data for an item might be `Action|Crime|Biopic`. If you have a multiple levels of categorical data and some items have multiple categories for each level in the hierarchy, add a field for each level and append a level indicator after each field name: GENRES, GENRE_L2, GENRE_L3. This allows you filter recommendations based on sub-categories, even if an item belongs to multiple multi-level categories. For example, a video might have the following data for each category level:

- GENRES: Action|Adventure
- GENRE_L2: Crime|Western
- GENRE_L3: biopic

In this example, the video is in the action > crime > biopic hierarchy *and* the adventure > western > biopic hierarchy. We recommend only using up to L3 but you can use more levels if necessary. For information on creating and using filters, see Filtering recommendations and user segments.

Default Items schema (VIDEO_ON_DEMAND domain)

The following is the default schema for Items datasets for the VIDEO_ON_DEMAND domain.

```
{
  "type": "record",
  "name": "Items",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
    {
      "name": "ITEM_ID",
      "type": "string"
```

```
    },
    {
      "name": "GENRES",
      "type": [
        "string"
      ],
      "categorical": true
    },
    {
      "name": "CREATION_TIMESTAMP",
      "type": "long"
    }
  ],
  "version": "1.0"
}
```

ECOMMERCE datasets and schemas

When you create a Domain dataset group for the ECOMMERCE domain, each dataset type has a default schema with a set of ECOMMERCE-specific required and recommended fields. You can use the default schema or create a new one based on the default schema. The data you import must match your schema in format and type. Use the default domain schemas listed in the sections below as a guide to determine what data to import to create your ECOMMERCE-based recommender.

You are free to add additional fields. As long as the fields aren't listed as required or reserved, and the data types are listed in Schema data types, the field names and data types are up to you.

For information about general Amazon Personalize schema requirements, such as formatting requirements and available field data types, see Schemas. These requirements apply to all schemas, regardless of domain.

The following topics provide information about each dataset's required and recommended fields for the ECOMMERCE domain. Each dataset section includes the default ECOMMERCE schema in JSON format.

Topics

- Items dataset requirements (ECOMMERCE domain)

ECOMMERCE domain dataset and schema requirements

Each dataset type has the following required fields and 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. Reserved categorical string fields must have `categorical` set to `true`, while reserved string fields can't be categorical. The keywords can't be in your data.

Dataset type	Required fields	Reserved keywords
Item interactions (default schema)	USER_ID (`string`) ITEM_ID (`string`) TIMESTAMP (`long`) EVENT_TYPE (`string` and depending on use case, `Purchase` and `View` event types)	EVENT_VALUE (`float`, `null`) IMPRESSION (`string`, `null`) RECOMMENDATION_ID (`string`, `null`) EVENT_ATTRIBUTION_SOURCE (`string`, `null`)
Users (default schema)	USER_ID (`string`) 1 metadata field (categorical `string` or numerical)	
Items (default schema)	ITEM_ID (`string`) PRICE (`float`) CATEGORY_L1 (categorical `string`)	CATEGORY_L2 (categorical `string`, `null`) CATEGORY_L3 (categorical `string`, `null`) PRODUCT_DESCRIPTION (textual `string`, `null`) CREATION_TIMESTAMP (`long`)

Dataset type	Required fields	Reserved keywords
		AGE_GROUP (categorical `string`, `null`) ADULT (categorical `string`, `null`) GENDER (categorical `string`, `null`)

Item interactions dataset requirements (ECOMMERCE domain)

An *Item interactions dataset* stores historical and real-time data from interactions between users and items in your ECOMMERCE catalog. For more information about the types of data you can store in an interactions dataset, see Item interactions dataset. For information about general Amazon Personalize schema requirements, such as formatting requirements and available field data types, see Schemas. These requirements apply to all schemas, regardless of domain.

You must at minimum create an Item interactions dataset and your schema must have the following fields:

- USER_ID (`string`)
- ITEM_ID (`string`)
- TIMESTAMP (`long`)
- EVENT_TYPE (`string` and depending on use case, `Purchase` and `View` event types)

Your schema can also include the following reserved keywords:

- EVENT_VALUE (`float`, `null`)
- IMPRESSION (`string`, `null`)
- RECOMMENDATION_ID (`string`, `null`)

The data you import must match your schema. You are free to add additional fields depending on your use case and your data. As long as the fields aren't listed as required or reserved, and the data types are listed in Schema data types, the field names and data types are up to you. For an example of the default schema for Item interactions datasets for ECOMMERCE domains, see Default Interactions schema (ECOMMERCE domain).

Optionally add the reserved keyword EVENT_VALUE if you have value data for events. Optionally add the reserved keyword IMPRESSION if you want to include explicit and implicit impressions data. For more information about recording impressions data see Impressions data.

The maximum total number of optional metadata fields you can add to an Item interactions dataset, combined with total number of *distinct* event types in your Item interaction 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. If you have an interaction with a categorical value with more than 1000, your dataset import job will fail.

For more information on minimum requirements and maximum data limits for an Item interactions dataset for the ECOMMERCE domain, see Service quotas.

Default Interactions schema (ECOMMERCE domain)

The following is the default ECOMMERCE domain schema for Item interactions datasets.

```
{

  "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": "TIMESTAMP",
```

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

Users dataset requirements (ECOMMERCE domain)

A *Users dataset* stores metadata about your users. This might include information such as age, gender, and loyalty membership for each user. For more information on the types of user data you can import into Amazon Personalize, see Users dataset. For information about general Amazon Personalize schema requirements, such as formatting requirements and available field data types, see Schemas. These requirements apply to all schemas, regardless of domain.

A Users dataset is optional for all ECOMMERCE use cases. If you have user data, we recommend creating one to get the most relevant recommendations. If you create a Users dataset, your schema must include the following fields.

- USER_ID
- 1 metadata field (categorical `string` or numerical)

The data you import must match your schema. You are free to add additional fields depending on your use case and your data. As long as the fields aren't listed as required or reserved, and the data types are listed in Schema data types, the field names and data types are up to you. For an example of the default schema for Users datasets for ECOMMERCE domains, see Default Users schema (ECOMMERCE domain).

For more information on minimum requirements and maximum data limits for a Users dataset, see Service quotas.

Using categorical data

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 individual item imports. For users with multiple categories, separate each value using the vertical bar, '|'. For example, for a SUBSCRIPTION_MODEL field, your data for a user might be student|monthly|discount.

Categorical values can have at most 1000 characters. If you have a user with a categorical value with more than 1000 characters, your dataset import job will fail.

Default Users schema (ECOMMERCE domain)

The following is the default ECOMMERCE domain schema for Users datasets with a CATEGORY field as the required metadata field.

```
{

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

Items dataset requirements (ECOMMERCE domain)

An *Items dataset* stores metadata about your ECOMMERCE items. This might include information such as price, category, and product description for each item. For more information on the types of item data you can import into Amazon Personalize, see Items dataset. For information about general Amazon Personalize schema requirements, such as formatting requirements and available field data types, see Schemas. These requirements apply to all schemas, regardless of domain.

An Items dataset is optional for all ECOMMERCE use cases. If you have items data, we recommend creating one to get the most relevant recommendations. If you create an items dataset, your schema must include the following fields:

- ITEM_ID
- PRICE (`float`)
- CATEGORY_L1 (categorical `string`)

Your schema can also include the following reserved keywords. For categorical fields, you can define your own range of values based on your use case.

- CATEGORY_L2 (categorical `string`, `null`)
- CATEGORY_L3 (categorical `string`, `null`)
- PRODUCT_DESCRIPTION (textual `string`, `null`)
- CREATION_TIMESTAMP (`float`)
- AGE_GROUP (categorical `string`, `null`): The age group the item is for. Values might be newborns, infants, children, and adults.
- ADULT (categorical `string`, `null`): Whether the item is restricted to only adults, such as alcohol. Values might be yes or no.
- GENDER (categorical `string`, `null`): The gender the item is for. Values might be male, female, and unisex.

To get the best recommendations, we recommend that you keep these as many of these fields in your schema as you have data. The data you import must match your schema. The data you import must match your schema. The maximum number of metadata columns is 100. You are free to add additional fields depending on your use case and your data. As long as the fields aren't listed as required or reserved, and the data types are listed in Schema data types, the field names and data types are up to you.

Use reserved keywords CATEGORY_L2 and CATEGORY_L3 for items with multiple multi-level categories. For more information, see Using categorical data. For information on textual and categorical metadata see Unstructured text metadata. For an example of the default schema for Items datasets for ECOMMERCE domains, see Default Items schema (ECOMMERCE domain).

Using categorical data

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 individual item imports. You can define your own range of values based on your use case. Categorical values can have at most 1000 characters. If you have an item with a categorical value with more than 1000 characters, your dataset import job will fail.

For items with multiple categories, separate each value with the vertical bar, '|'. For example, for a CATEGORY_L1 field your data for an item might be `Electronics|Productivity|Mouse`. If you have a multiple levels of categorical data and some items have multiple categories for each level in the hierarchy, add a field for each level and append a level indicator after each field name: CATEGORY_L1, CATEGORY_L2, CATEGORY_L3. This allows you filter recommendations based on sub-categories, even if an item belongs to multiple multi-level categories. For example, an item might have the following data for each category level:

- CATEGORY_L1: Electronics|Productivity
- CATEGORY_L2: Productivity|Computers
- CATEGORY_L3: Mouse

In this example, the item is in the electronics > productivity > mouse hierarchy *and* the productivity > computers > mouse hierarchy. We recommend only using up to L3 but you can use more levels if necessary. For information on creating and using filters see Filtering recommendations and user segments.

Default Items schema (ECOMMERCE domain)

The following is the default schema for Items datasets for the ECOMMERCE domain with only the required fields.

```
{
  "type": "record",
  "name": "Items",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
    {
      "name": "ITEM_ID",
      "type": "string"
    },
    {
      "name": "PRICE",
      "type": "float"
    },
    {
      "name": "CATEGORY_L1",
      "type": [
        "string"
      ],
      "categorical": true
    }
```

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

Custom datasets and schemas

When you create a Custom dataset group, you create your own schemas from scratch. Custom dataset group datasets and schemas have fewer required fields and more flexibility. The following topics explain the schema and data requirements for datasets a Custom dataset group. Each dataset section lists the required data for the dataset type and provides a JSON example of a schema.

For information on the types of data you can import into Amazon Personalize see Datasets. For information about general Amazon Personalize schema requirements, such as formatting requirements and available field data types, see Schemas. These requirements apply to all Amazon Personalize schemas.

Topics

Custom dataset and schema requirements

When you create a dataset for a Custom dataset group, each dataset type has the following required fields and reserved keywords with required data types.

Dataset type	Required fields	Reserved keywords
Item interactions (schema example)	USER_ID (`string`) ITEM_ID (`string`) TIMESTAMP (`long`)	EVENT_TYPE (`string`) EVENT_VALUE (`float`, `null`) IMPRESSION (`string`, `null`)

Dataset type	Required fields	Reserved keywords
		RECOMMENDATION_ID (string, null) EVENT_ATTRIBUTION_SOURCE (string, null)
Users (schema example)	USER_ID (string) 1 metadata field (categorical string or numerical)	
Items (schema example)	ITEM_ID (string) 1 metadata field (categorical or textual string field or numerical field)	CREATION_TIMESTAMP (long)
Actions (schema example)	ACTION_ID (string)	CREATION_TIMESTAMP (long) VALUE (long, null) TYPE (string, null) EXPIRATION_TIMESTAMP (long, null) REPEAT_FREQUENCY (long, null)
Action interactions (schema example)	USER_ID (string) ACTION_ID (string) EVENT_TYPE (string) TIMESTAMP (long)	IMPRESSION (string, null) RECOMMENDATION_ID (string, null)

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 25 metadata fields for a Users schema, 100 metadata fields for an Items schema, and 10 metadata fields for an Actions 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. Reserved categorical string fields must have `categorical` set to `true`, while reserved string fields can't be categorical. The following are reserved keywords:

- EVENT_TYPE: For Item 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` and can't be set as categorical.
- EVENT_VALUE: For Item 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 or Actions 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](#).
- IMPRESSION: For Item interactions datasets with explicit impressions data, use an `IMPRESSION` field with type `String` and optionally type `null`. 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](#).
- RECOMMENDATION_ID: For Item 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 operation without it. For more information see Impressions data.

- VALUE: For Actions datasets, if you have value you data for some or all of your actions, add a `VALUE` field to your schema. For its type, use `long` and optionally type `null`. For more information about actions and their value, see Value data.
- ACTION_EXPIRATION_TIMESTAMP: For Actions datasets, if you have an expiration timestamp for some or all of your actions, add a `ACTION_EXPIRATION_TIMESTAMP` field to your schema. For its type, use `long` and optionally type `null`. For more information about expiration timestamps, see Action expiration timestamp data.
- REPEAT_FREQUENCY: For Actions datasets, if you have repeat frequency data for some or all of your actions, add a `REPEAT_FREQUENCY` field to your schema. For its type, use `long` and optionally type `null`. For more information about repeat frequency data, see Repeat frequency data.

Item interactions dataset schema requirements (custom)

An *Item interactions dataset* stores historical and real-time data from interactions between users and items in your catalog. For information on the types of interactions data Amazon Personalize can use, see Item interactions dataset.

The 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)

You are free to add additional fields depending on your use case and your data. As long as the fields aren't listed as required or reserved, and the data types are listed in Schema data types, the field names and data types are up to you.

The maximum total number of optional metadata fields you can add to an Item interactions dataset, combined with total number of *distinct* event types in your Item interaction 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. If you have an interaction with a categorical value with more than 1000, your dataset import job will fail.

For more information on minimum requirements and maximum data limits for an Item interactions dataset, see Service quotas.

Interactions schema example (custom)

The following example shows a schema for an Item 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. EVENT_TYPE must of type string and can't be categorical. `LOCATION` and `DEVICE` are optional contextual metadata fields. For information on schema requirements see Custom dataset and schema requirements.

```
{

  "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",
                "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 schema requirements (custom)

A *Users dataset* stores metadata about your users. This might include information such as age, gender, and loyalty membership for each item. For information on the types of user data you can import into Amazon Personalize, see Users dataset.

The 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. Your Users schema must have minimum one metadata field, but if you add a `null` type, this value can be null for the user. You are free to add additional fields depending on your use case and your data. As long as the fields aren't listed as required or reserved, and the data types are listed in Schema data types, the field names and data types are up to you.

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 individual item imports. For users with multiple categories, separate each value using the vertical bar, '|'. For example, for a SUBSCRIPTION_MODEL field, your data for a user might be student|monthly|discount.

Categorical values can have at most 1000 characters. If you have a user with a categorical value with more than 1000 characters, your dataset import job will fail.

For more information on minimum requirements and maximum data limits for a Users dataset, see Service quotas.

Users schema example (custom)

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 25 metadata fields. For information about schema requirements see Custom dataset and schema requirements.

```
{
  "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 schema requirements (custom)

An *Items dataset* stores metadata about your items in your catalogue. This might include information such as price, genre, and availability for each item. For information about the types of item data you can import into Amazon Personalize, see Items dataset.

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. Your schema must have minimum one metadata field, but if you add a `null` type, this value can be null for the item. You are free to add additional fields depending on your use case and your data. As long as the fields aren't listed as required or reserved, and the data types are listed in Schema data types, the field names and data types are up to you.

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 individual item imports. Categorical values can have at most 1000 characters. If you have an item with a categorical value with more than 1000 characters, your dataset import job will fail.

For items with multiple categories, separate each value with the vertical bar, '|'. For example, for a GENRES field your data for an item might be `Action|Crime|Biopic`. If you have a multiple levels of categorical data and some items have multiple categories for each level in the hierarchy, add a field for each level and append a level indicator after each field name: GENRES, GENRE_L2, GENRE_L3. This allows you filter recommendations based on sub-categories, even if an item belongs to multiple multi-level categories (for information on creating and using filters see Filtering recommendations and user segments). For example, a video might have the following data for each category level:

- GENRES: Action|Adventure
- GENRE_L2: Crime|Western
- GENRE_L3: Biopic

In this example, the video is in the action > crime > biopic hierarchy *and* the adventure > western > biopic hierarchy. We recommend only using up to L3 but you can use more levels if necessary.

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.

Items dataset schema example (custom)

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 100 metadata fields. The `CREATION_TIMESTAMP` field is a reserved keyword. For information about schema requirements, see Custom dataset and schema requirements.

```
{
  "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,"This is 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."
...
...
```

Actions dataset schema requirements (custom)

An *action* is an engagement activity that you might want to recommend to your customers. Actions might include installing your mobile app, completing a membership profile, joining your loyalty program, or signing up for promotional emails. An *Actions dataset* stores data about your actions. For information about the types of action data you can import into Amazon Personalize, see Actions dataset.

The data you provide for each action must match your Actions dataset schema. Depending on your schema, action metadata can include empty/null values. At minimum, your schema must have an ACTION_ID field and you must provide an ID for each action.

You can add additional fields depending on your use case and your data. You can choose the field names and data types unless the fields are listed as required or reserved, and the data types are listed in Schema data types.

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 individual action imports. Categorical values can have at most 1000 characters. If you have an action with a categorical value with more than 1000 characters, your dataset import job will fail.

For actions with multiple categories, separate each value with the vertical bar, '|'. For example, for a MEMBERSHIP_LEVEL field, your data for an action might be `Premium|Deluxe|Exclusive`.

During model training, Amazon Personalize considers a maximum of 1000 actions. If you import more than 1000 actions, Amazon Personalize decides which actions to include in training, with priority given to new actions (actions you recently added with no interactions) and existing actions with recent interactions data.

For more information on minimum requirements and maximum data limits for an Actions dataset, see Service quotas.

Actions dataset schema example (custom)

The following example shows how to structure an Actions schema. The `ACTION_ID` field is required. The `MEMBERSHIP_LEVEL` field is a categorical string field. The `VALUE`, `CREATION_TIMESTAMP`, and `REPEAT_FREQUENCY` fields are reserved keywords with the required types. You can add a maximum of 10 columns. For information about schema requirements, see Custom dataset and schema requirements.

```
{
```

```
  "type": "record",
  "name": "Actions",
  "namespace": "com.amazonaws.personalize.schema",
  "fields": [
    {
      "name": "ACTION_ID",
      "type": "string"
    },
    {
      "name": "VALUE",
      "type": [
        "null",
        "long"
      ]
    },

    {
      "name": "MEMBERSHIP_LEVEL",
      "type": [
        "null",
        "string"
      ],
      "categorical": true
    },

    {
      "name": "CREATION_TIMESTAMP",
      "type": "long"
    },
    {
      "name": "REPEAT_FREQUENCY",
      "type": [
        "long",
        "null"
      ]
    }
  ],
  "version": "1.0"
}
```

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

```
ACTION_ID,VALUE,MEMBERSHIP_LEVEL,CREATION_TIMESTAMP,REPEAT_FREQUENCY
```

```
1,10,Deluxe|Premium,1510003267,7
2,5,Basic,1580003267,7
3,5,Preview,1590003267,3
4,10,Deluxe|Platinum,1560003267,4
...
...
```

Action interactions dataset schema requirements (custom)

An *Action interactions dataset* stores historical and real-time data from interactions between users and actions in your *Actions dataset*. For information on the types of data Amazon Personalize can use, see Action interactions dataset.

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

- USER_ID
- ACTION_ID
- TIMESTAMP
- EVENT_TYPE

You can add additional fields depending on your use case and your data. You can choose the field names and data types unless the fields are listed as required or reserved, and the data types are listed in Schema data types.

For more information about minimum requirements and maximum data limits for an Action interactions dataset, see Service quotas.

Action interactions dataset schema example (custom)

The following example shows a schema for an Action interactions dataset with only the required fields. For information about general schema formatting requirements, see Schema formatting requirements.

```
{

  "type": "record",
  "name": "ActionInteractions",
  "namespace": "com.amazonaws.personalize.schema",
```

```
  "fields": [
      {
          "name": "USER_ID",
          "type": "string"
      },
      {
          "name": "ACTION_ID",
          "type": "string"
      },
      {
          "name": "EVENT_TYPE",
          "type": "string"
      },
      {
          "name": "TIMESTAMP",
          "type": "long"
      }
  ],
  "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 IMPRESSION are null.

```
USER_ID,ACTION_ID,EVENT_TYPE,TIMESTAMP
35,73,Viewed,1586731606
54,35,Not taken,1586731609
9,33,Viewed,1586735158
23,10,Taken,1586735697
27,11,Taken,1586735763
...
...
```

Creating a schema with SDK for Python (Boto3)

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 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 Data format guidelines.

Data format guidelines

When you import data into Amazon Personalize datasets, you can choose to import records in bulk, individually, or both.

- Bulk imports involve importing a large number of historical records at once. You can prepare and import your bulk data with SageMaker Data Wrangler and multiple data sources. Or you can prepare bulk data yourself, and import it directly into Amazon Personalize from a CSV file in Amazon S3.
- With individual imports, you import individual records with the Amazon Personalize console and API operations. Or you can stream interactions data from live events in real time. For more information about individual imports, see Importing individual records.

Before you import your bulk data, make sure it's properly formatted. The following sections can help you format your bulk data. If you're not sure how to format your data, you can use Amazon SageMaker Data Wrangler (Data Wrangler) to prepare your data. For more information, see Preparing and importing data using Amazon SageMaker Data Wrangler.

Topics

Bulk data format guidelines and requirements

The following guidelines and requirements can help you make sure your bulk data is formatted correctly.

- Your input data must be in a CSV (comma-separated values) file.
- The first row of your CSV file must contain your column headers. Don't enclose headers in quotation marks (").
- Make sure you have the required fields for your dataset type and make sure that their names align with Amazon Personalize requirements. For example, your Items data might have a column called `ITEM_IDENTIFICATION_NUMBER` with IDs for each of your items. To use this column as an ITEM_ID field, rename the column to `ITEM_ID`. If you use Data Wrangler to format your data, you can use the **Map columns for Amazon Personalize** Data Wrangler transform to make sure your columns are named correctly.

  For information about required fields, see Schemas. For information about using Data Wrangler to prepare your data, see Preparing and importing data using Amazon SageMaker Data Wrangler.
- The column header names in your CSV file must map to your schema.
- Each record in your CSV file must be on a single line.
- The data types in each column must map to your schema. If you use Data Wrangler to format your data, you can use the Data Wrangler transform Parse Value as Type to convert the data types.
- `TIMESTAMP` and `CREATION_TIMESTAMP` data must be in *UNIX epoch* time format. For more information, see Timestamp data.
- If your data includes any non-ASCII encoded characters, your CSV file must be encoded in UTF-8 format.
- Makes sure you format any textual data as described in Unstructured text metadata.
- Make sure you format impression data and categorical data as described in Formatting explicit impressions and Formatting categorical data.

Interactions data example

The following interactions data represents historical user activity from a website that sells movie tickets. You might use the data to train a model that provides movie recommendations based on users' interaction data.

```
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
```

Here's associated Interactions schema:.

```
{
  "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 Item 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 recipe, you can record and upload impressions data. Impressions are lists of items that were visible to a user when interacting with a particular item (for example, clicked or watched). To upload impressions data in a bulk data import, manually record each item ID. Be sure to separate the values with a vertical bar, '|', character as part of your historical interactions data. The vertical bar character counts toward the 1000 character limit for impressions data. For more information on impressions data, see Impressions data.

The following is a short excerpt from an Item 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

EVENT_TYPE	IMPRESSION	ITEM_ID	TIMESTAMP	USER_ID
...	...	...	...	...

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.

Formatting 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, for an item that has 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.

Domain use cases and custom recipes

Amazon Personalize provides different domain use cases and custom recipes for training models:

- When you create a recommender in a Domain dataset group, you specify a use case. Amazon Personalize trains the models backing the recommender with the best configurations for the use case.
- When you create a custom solution in a Custom dataset group or Domain dataset group, you specify a recipe and configure training parameters. When you create a solution version for the solution, Amazon Personalize trains the models backing the solution version based on the recipe and training configuration.

Topics

- Use case and recipe features
- Choosing a use case
- Choosing a recipe

Use case and recipe features

With some use case and recipes, Amazon Personalize uses the following features to generate more relevant recommendations and improve item discovery and engagement.

Topics

- Real-time personalization
- Exploration
- Automatic updates

Real-time personalization

With some use cases and recipes, Amazon Personalize uses real-time personalization to update and adapt recommendations according to a user's evolving interest. It updates recommendations for a user as you record their interactions with items or actions present at the latest full training. You record these interactions with an event tracker and the PutEvents operation or, for interactions with actions, the PutActionInteractions operation.

For more information about recording events, see Recording events. For information about new data influences real-time recommendations, including real-time personalization, see How new data influences real-time recommendations.

The following use cases and recipes support real-time personalization:

- Recommended for you (ECOMMERCE use case)
- Top picks for you (VIDEO_ON_DEMAND use case)
- User-Personalization recipe
- Personalized-Ranking recipe
- Next-Best-Action recipe

Exploration

For some domain use cases and custom recipes, Amazon Personalize uses exploration when recommending items. With exploration, recommendations include some items or actions that would be typically less likely to be recommended for the user, such as new items or actions, items or actions with few interactions, or items or actions less relevant for the user based on their previous behavior. 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 because they are fresh.

If your use case or recipe uses exploration, when you create a recommender or custom campaign, or when you create a batch inference job (custom resources), you can configure exploration with the following fields:

- Emphasis on exploring less relevant items (exploration weight) – Configure how much to explore. Specify a decimal value between 0 to 1. The default is 0.3. The closer the value is to 1, the more exploration. With more exploration, recommendations include more items with less item interactions data or relevance based on previous behavior. At zero, no exploration occurs and recommendations are based on current data (relevance).
- Exploration item age cutoff – Specify the maximum item age in days since the latest interaction across all items in the Item interactions dataset. This defines the scope of item exploration based on item age. Amazon Personalize determines item age based on its creation timestamp or, if creation timestamp data is missing, item interactions data. For more information how Amazon Personalize determines item age, see Creation timestamp data.

To increase the items Amazon Personalize considers during exploration, enter a greater value. The minimum is 1 day and the default is 30 days. Recommendations might include items that are older than the item age cut off you specify. This is because these items are relevant to the user and exploration didn't identify them.

Use cases and recipes that use exploration

For more information about each use case or recipe that uses exploration, see the following:

- Recommended for you (ECOMMERCE use case)
- Top picks for you (VIDEO_ON_DEMAND use case)
- User-Personalization recipe
- Next-Best-Action recipe

Automatic updates

For some use cases and custom recipes, Amazon Personalize automatically updates your recommender or solution version to consider new items or actions for recommendations. There is no cost for automatic updates. For a list of use cases and recipes with automatic updates, see Domain use cases and custom recipes with automatic updates.

Automatic updates work as follows:

- When Amazon Personalize automatically updates your solution version or recommender depends on how you get recommendations:
  - For real-time recommendations, Amazon Personalize updates the solution version or recommender every two hours.
  - For batch item recommendations, when you create a batch inference job and specify the latest fully trained solution version for your solution, Amazon Personalize automatically updates the solution version to consider new items during exploration. If you don't specify the latest solution version, no update occurs.
- With each update, Amazon Personalize starts including new items in recommendations using Exploration. When considering a new item or action, Amazon Personalize considers any metadata for the item. However, this data will have a greater effect on recommendations only after you record interactions for the item and fully retrain.

- For an update to occur, you must provide new action, item, or interactions data since the last automatic update or retraining.
- Amazon Personalize considers new items until you import 750,000 items. This is the maximum number of items considered during training.

Additional guidelines and requirements for custom resources

If you use custom resources, the following are guidelines and requirements for auto updates:

- Your solution version must be deployed in a campaign. Your campaign automatically uses the updated solution version.
- Automatic updates aren't the same as automatic training. An automatic update doesn't create a completely new solution version. And the model doesn't learn from your latest data. To maintain your solution, your automatic training frequency should still be at least weekly.
- After your solution automatically creates a new solution version or you manually create a new one, Amazon Personalize will not automatically update older solution versions, even if you deployed them in a campaign.
- If every two hours is not frequent enough, with User-Personalization 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. The manually updated solution version won't be automatically updated in the future. If your solution uses automatic training, auto updates will resume for the next solution version. If not, manually create a new solution with training mode set to `FULL` and deploy it in a campaign.

Domain use cases and custom recipes with automatic updates

For more information about each use case or recipe that features automatic updates, see the following:

- Recommended for you (ECOMMERCE use case)
- Top picks for you (VIDEO_ON_DEMAND use case)
- User-Personalization recipe
- Next-Best-Action recipe

Choosing a use case

When you create a recommender in a Domain dataset group, you specify a use case. Amazon Personalize trains the models backing the recommender with the best configurations for the use case. Each domain has different use cases. For example, if you specify *VIDEO_ON_DEMAND* for your Domain dataset group, only VIDEO_ON_DEMAND use cases are available. Each use case has different requirements for getting recommendations. Some use cases require specific event types. You are free to include additional event types.

For all use cases, your interactions data must have the following:

- At minimum 1000 item interactions records from users interacting with items in your catalog. These interactions can be from bulk imports, or streamed events, or both.
- At minimum 25 unique user IDs with at least two item interactions for each.

For quality recommendations, we recommend that you have at minimum 50,000 item interactions from at least 1,000 users with two or more item interactions each.

Topics

- VIDEO_ON_DEMAND use cases
- ECOMMERCE use cases

VIDEO_ON_DEMAND use cases

The following sections list the requirements and Amazon Resource Name (ARN) for each VIDEO_ON_DEMAND use case. For all use cases, your interactions data must have the following:

- At minimum 1000 item interactions records from users interacting with items in your catalog. These interactions can be from bulk imports, or streamed events, or both.
- At minimum 25 unique user IDs with at least two item interactions for each.

For quality recommendations, we recommend that you have at minimum 50,000 item interactions from at least 1,000 users with two or more item interactions each.

> **Note**
>
> If you use the CreateRecommender API, provide the ARN listed here for the recipe ARN.

Topics

Because you watched X

Get recommendations for videos that other users also watched based on a video that you specify. With this use case, Amazon Personalize automatically filters videos the user watched based on the userId you specify and `Watch` events. If you apply your own filter, your filter is applied after the videos the user watched are filtered out.

When filtering, Amazon Personalize considers at most 100 item interactions per user per event type. This applies to any automatic or custom filters. You can use the Service Quotas console to request an increase for this limit. For more information, see the Requesting a quota increase section of the *Service Quotas User Guide*.

- **Recipe ARN:** `arn:aws:personalize:::recipe/aws-vod-because-you-watched-x`
- **GetRecommendations API requirements:**

  `userId`: Required

  `itemId`: Required
- **Datasets used when training:** Only Item interactions dataset (required)
- **Required event types:** At minimum, 1000 `Watch` events.

More like X

Get recommendations for videos that are similar to a video that you specify. With this use case, Amazon Personalize automatically filters videos the user watched based on the userId that you specify and `Watch` events. If you apply your own filter, your filter is applied after the videos the user watched are filtered out.

When filtering, Amazon Personalize considers at most 100 item interactions per user per event type. This applies to any automatic or custom filters. You can use the [Service Quotas console](#) to request an increase for this limit. For more information, see the [Requesting a quota increase](#) section of the *Service Quotas User Guide*.

- **Recipe ARN:** `arn:aws:personalize:::recipe/aws-vod-more-like-x`
- **GetRecommendations API requirements:**

  `userId`: Required

  `itemId`: Required
- **Datasets used when training:**
  - Interactions (required)
  - Items (required)
- **Required number of events:** At minimum, 1000 events of any type.
- **Recommended event types:** `Watch` and `Click` events.

Most popular

Get recommendations for videos that have been watched by the most users.

- **Recipe ARN:** `arn:aws:personalize:::recipe/aws-vod-most-popular`
- **GetRecommendations requirements:**

  `userId`: Required

  `itemId`: Not used
- **Datasets used when training:** Only Item interactions dataset (required)
- **Required event types:** At minimum, 1000 `Watch` events.

Trending now

Get recommendations for videos that are currently trending. Trending videos are items that are rapidly becoming more popular with your users. Every two hours, Amazon Personalize automatically evaluates your interactions data and identifies trending items.

- **Recipe ARN:** `arn:aws:personalize:::recipe/aws-vod-trending-now`
- **GetRecommendations API requirements:**

  `userId`: Required only if you filter by CurrentUser or by items a user has interacted with

  `itemId`: Not used
- **Datasets used when training:** Only Item interactions dataset (required)
- **Required number of events:** At minimum, 1000 events of any type.

Top picks for you

Get personalized content recommendations for a user that you specify. With this use case, Amazon Personalize automatically filters videos the user watched based on the userId that you specify and `Watch` events. If you apply your own filter, your filter is applied after the videos the user watched are filtered out.

When filtering, Amazon Personalize considers at most 100 item interactions per user per event type. This applies to any automatic or custom filters. You can use the Service Quotas console to request an increase for this limit. For more information, see the Requesting a quota increase section of the *Service Quotas User Guide*.

When recommending items, this use case uses real-time-personalization and exploration. And it uses automatic updates to consider new items for recommendations.

- **Recipe ARN:** `arn:aws:personalize:::recipe/aws-vod-top-picks`
- **GetRecommendations requirements:**

  `userId`: Required

  `itemId`: Not used
- **Datasets used when training:**
  - Interactions (required)

  - Items (optional)
  - Users (optional)
- **Required number of events:** At minimum, 1000 events.
- **Recommended event types:** `Click` and `Watch` events.
- **Exploration configuration parameters:** When you create a recommender, you can configure exploration with the following.
  - Emphasis on exploring less relevant items (exploration weight) – Configure how much to explore. Specify a decimal value between 0 to 1. The default is 0.3. The closer the value is to 1, the more exploration. With more exploration, recommendations include more items with less item interactions data or relevance based on previous behavior. At zero, no exploration occurs and recommendations are based on current data (relevance).
  - Exploration item age cutoff – Specify the maximum item age in days since the latest interaction across all items in the Item interactions dataset. This defines the scope of item exploration based on item age. Amazon Personalize determines item age based on its creation timestamp or, if creation timestamp data is missing, item interactions data. For more information how Amazon Personalize determines item age, see Creation timestamp data.

    To increase the items Amazon Personalize considers during exploration, enter a greater value. The minimum is 1 day and the default is 30 days. Recommendations might include items that are older than the item age cut off you specify. This is because these items are relevant to the user and exploration didn't identify them.

ECOMMERCE use cases

The following sections list the requirements and Amazon Resource Name (ARN) for each ECOMMERCE use case. For all use cases, your interactions data must have the following:

- At minimum 1000 item interactions records from users interacting with items in your catalog. These interactions can be from bulk imports, or streamed events, or both.
- At minimum 25 unique user IDs with at least two item interactions for each.

For quality recommendations, we recommend that you have at minimum 50,000 item interactions from at least 1,000 users with two or more item interactions each.

> **Note**
>
> If you use the CreateRecommender API, provide the ARN listed here for the recipe ARN.

Topics

- Most viewed
- Best sellers
- Frequently bought together
- Customers who viewed X also viewed
- Recommended for you

Most viewed

Get recommendations for popular items based on how many times your customers viewed an item.

- **Recipe ARN:** `arn:aws:personalize:::recipe/aws-ecomm-popular-items-by-views`
- **GetRecommendations requirements:**

  `userId`: Required

  `itemId`: Not used

  `inputList`: NA
- **Datasets used when training:** Only Item interactions dataset (required)
- **Required event types:** At minimum, 1000 `View` events.

Best sellers

Get recommendations for popular items based on how many times your customers purchased an item.

- **Recipe ARN:** `arn:aws:personalize:::recipe/aws-ecomm-popular-items-by-purchases`
- **GetRecommendations requirements:**

  `userId`: Required

`itemId`: Not used

`inputList`: NA

- **Datasets used when training:** Only Item interactions dataset (required)
- **Required event types:** At minimum, 1000 `Purchase` events.

Frequently bought together

Get recommendations for items that customers frequently buy together along with an item that you specify.

- **Recipe ARN:** `arn:aws:personalize:::recipe/aws-ecomm-frequently-bought-together`
- **GetRecommendations requirements:**

  `userId`: Required only if you filter by CurrentUser

  `itemId`: Required

  `inputList`: NA
- **Datasets used when training:** Only Item interactions dataset (required)
- **Required event types:** At minimum, 1000 `Purchase` events.

Customers who viewed X also viewed

Get recommendations for items that customers also viewed based on an item that you specify. With this use case, Amazon Personalize automatically filters items the user purchased based on the userId that you specify and `Purchase` events. If you apply your own filter, your filter is applied after the items the user already purchased are filtered out.

When filtering, Amazon Personalize considers at most 100 item interactions per user per event type. This applies to any automatic or custom filters. You can use the Service Quotas console to request an increase for this limit. For more information, see the Requesting a quota increase section of the *Service Quotas User Guide*.

- **Recipe ARN:** `arn:aws:personalize:::recipe/aws-ecomm-customers-who-viewed-x-also-viewed`

- **GetRecommendations requirements:**

  `userId`: Required

  `itemId`: Required

  `inputList`: NA
- **Datasets used when training:** Only Item interactions dataset (required)
- **Required event types:** At minimum, 1000 `View` events.
- **Recommended event types:** `Purchase` events.

Recommended for you

Get personalized recommendations for items based on a user that you specify. With this use case, Amazon Personalize automatically filters out items the user purchased based on the userId that you specify and `Purchase` events. If you apply your own filter, your filter is applied after the items the user already purchased are filtered out.

When filtering, Amazon Personalize considers at most 100 item interactions per user per event type. This applies to any automatic or custom filters. You can use the Service Quotas console to request an increase for this limit. For more information, see the Requesting a quota increase section of the *Service Quotas User Guide*.

When recommending items, this use case uses real-time-personalization and exploration. And it uses automatic updates to consider new items for recommendations.

- **Recipe ARN:** `arn:aws:personalize:::recipe/aws-ecomm-recommended-for-you`
- **GetRecommendations requirements:**

  `userId`: Required

  `itemId`: Not used

  `inputList`: NA
- **Datasets used when training:**
  - Interactions (required)
  - Items (optional)
  - Users (optional)

- **Required number of events:** At minimum, 1000 events.
- **Recommended event types:** `View` and `Purchase` events.
- **Exploration configuration parameters:** When you create a recommender, you can configure exploration with the following.
  - Emphasis on exploring less relevant items (exploration weight) – Configure how much to explore. Specify a decimal value between 0 to 1. The default is 0.3. The closer the value is to 1, the more exploration. With more exploration, recommendations include more items with less item interactions data or relevance based on previous behavior. At zero, no exploration occurs and recommendations are based on current data (relevance).
  - Exploration item age cutoff – Specify the maximum item age in days since the latest interaction across all items in the Item interactions dataset. This defines the scope of item exploration based on item age. Amazon Personalize determines item age based on its creation timestamp or, if creation timestamp data is missing, item interactions data. For more information how Amazon Personalize determines item age, see Creation timestamp data.

    To increase the items Amazon Personalize considers during exploration, enter a greater value. The minimum is 1 day and the default is 30 days. Recommendations might include items that are older than the item age cut off you specify. This is because these items are relevant to the user and exploration didn't identify them.

Choosing a recipe

When you create a custom solution, you specify a recipe and configure training parameters. *Recipes* are Amazon Personalize algorithms that are prepared for specific use cases. Amazon Personalize provides recipes, based on common use cases, for training models. When you create a solution version for the solution, Amazon Personalize trains the models backing the solution version based on the recipe and training configuration.

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.

Topics

Amazon Personalize recipe types by use case

To choose your recipe, first choose your use case from the following and note its corresponding recipe type.

- Recommending items for users (USER_PERSONALIZATION recipes)

  To provide personalized recommendations for your users, train your model with a USER_PERSONALIZATION recipe. Personalized recommendations help drive better engagement and conversion.
- 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 trending or popular items (POPULAR_ITEMS recipes)

  To recommend trending or popular items use a POPULAR_ITEMS recipe. You might use a POPULAR_ITEMS if your customers highly value what other users are interacting with. Common uses include recommending viral social media content, breaking news articles, or recent sports videos.

- 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.
- Recommending the next best action (PERSONALIZED_ACTIONS recipes)

  To recommend the next best action for your users in real time, such as signing up for your loyalty program or applying for a credit card, you should use a PERSONALIZED_ACTIONS recipe. Recommending the next best action can increase customer loyalty, generate more revenue, and improve your users' experience.
- Getting user segments (USER_SEGMENTATION recipes)

  To get segments of users based on item input data, such as users who will most likely interact with items with a certain attribute, you should use a USER_SEGMENTATION recipe. Getting user segments can help you create advanced marketing campaigns that promote different items to different user segments based on the likelihood that they will take an action.

Amazon Personalize recipes

Amazon Personalize provides the following 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	User-Personalization HRNN recipe (legacy) HRNN-Metadata recipe (legacy) HRNN-Coldstart recipe (legacy)	GetRecommendations	`userId`: Required `itemId`: Not used `inputList` : NA

Recipe type	Recipes	API	API requirements
POPULAR_ITEMS	Trending-Now Popularity-Count	GetRecommendations	`userId`: Required only if you apply a filter that requires it `itemId`: Not used `inputList` : NA
PERSONALIZED_RANKING	Personalized-Ranking	GetPersonalizedRanking	`userId`: Required `itemId`: NA `inputList` : list of itemId's
RELATED_ITEMS	Similar-Items SIMS	GetRecommendations	`userId`: Required only if you apply a filter that requires it `itemId`: Required `inputList` : NA

Recipe type	Recipes	API	API requirements
PERSONALIZED_ACTIONS	Next-Best-Action	GetActionRecommendations	`userId`: Required `actionId`: Not used `itemId`: Not used `inputList` : NA
USER_SEGMENTATION	Item-Affinity Item-Attribute-Affinity	CreateBatchSegmentJob	For batch workflow requirements, see Creating a batch segment job.

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 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 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

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
- Legacy user personalization recipes

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 most likely interact with. You might use User-Personalization to generate personalized movie recommendations for a streaming app or personalized product recommendations for a retail app.

With User-Personalization, Amazon Personalize generates recommendations primarily based on user item interaction data in an Item interactions dataset. It can also use any item and user metadata in your Items and Users datasets. For more information about the data it uses, see Required and optional datasets.

Topics

- Recipe features
- Required and optional datasets
- Properties and hyperparameters
- Training with the User-Personalization recipe (console)
- Training with the User-Personalization recipe (Python SDK)
- Getting recommendations and recording impressions (SDK for Python (Boto3))
- Sample Jupyter notebook

Recipe features

User-Personalization uses the following Amazon Personalize recipe features when generating item recommendations:

- Real-time personalization – With real-time personalization, Amazon Personalize updates and adapts item recommendations according to a user's evolving interest. For more information, see Real-time personalization.
- Exploration – With exploration, recommendations include new items or items with less interactions data. 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. For more information about exploration, see Exploration.
- Automatic updates – With automatic updates, Amazon Personalize automatically updates the latest model (solution version) every two hours to consider new items for recommendations. For more information, see Automatic updates.

Required and optional datasets

To use the User-Personalization, you must create an Item interactions dataset and import at minimum 1000 item interactions. Amazon Personalize generates recommendations primarily based on item interaction data.

With User-Personalization, Amazon Personalize can use Item interactions data that includes the following:

- Event type and event value data – Amazon Personalize uses event type data, such as click or watch event types, to identify user intent and interest through any patterns in their behavior. Also, you can use event type and event value data to filter records before training. For more information, see Event type and event value data.
- Contextual metadata – Contextual metadata is interactions data you collect on the user's environment at the time of an event, such as their location or device type. For more information, see Contextual metadata.
- 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. For more information, see Impressions data.

The following datasets are optional and can improve recommendations:

- Users dataset – Amazon Personalize can use data in your Users dataset to better understand your users and their interests. You can also use data in a Users dataset to filter recommendations. For information about the user data you can import, see Users dataset.
- Items dataset – Amazon Personalize can use data in your Items dataset to identify connections and patterns in their behavior. This helps Amazon Personalize understand your users and their interests. You can also use data in a Items dataset to filter recommendations. For information about the item data you can import, see Items dataset.

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 Choosing a recipe.

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.

The table 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	

Name	Description
hidden_dimension	The number of hidden variables used in the model. *Hidden variables* recreate users' purchase history and item statistics to generate ranking scores. Specify a greater number of hidden dimensions when your Item interactions dataset includes more complicated patterns. Using more hidden dimension s requires a larger dataset and more time to process. To decide on the best value, use HPO. To use HPO, set performHPO to true when you call CreateSolution and CreateSolutionVersion operations. Default value: 149 Range: [32, 256] Value type: Integer HPO tunable: Yes

Name	Description
`bptt`	Determines whether to use the back-propagation through time technique. *Back-propagation through time* is a technique that updates weights in recurrent neural network-based algorithms. Use `bptt` for long-term credits to connect delayed rewards to early events. For example, a delayed reward can be a purchase made after several clicks. An early event can be an initial click. Even within the same event types, such as a click, it’s a good idea to consider long-term effects and maximize the total rewards. To consider long-term effects, use larger `bptt` values. Using a larger `bptt` value requires larger datasets and more time to process. Default value: 32 Range: [2, 32] Value type: Integer HPO tunable: Yes

Name	Description
recency_mask	Determines whether the model should consider the latest popularity trends in the Item interactions dataset. Latest popularity trends might include sudden changes in the underlying patterns of interaction events. To train a model that places more weight on recent events, set `recency_mask` to `true`. To train a model that equally weighs all past interactions, set `recency_mask` to `false`. To get good recommendations using an equal weight, you might need a larger training dataset. Default value: `True` Range: `True` or `False` Value type: Boolean HPO tunable: Yes
Featurization hyperparameters	

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

Name	Description
`max_user_history_length_percentile`	The maximum percentile of user history lengths to include in model training. *History length* is the total amount of data about a user. Use `max_user_history_length_percentile` to exclude a percentage of users with long history lengths because data for these users tend to contain noise. For example, a robot might have a long list of automated interactions. Removing these users limits noise in training. Choose an appropriate value after you review user history lengths using a histogram or similar tool. We recommend setting a value that retains the majority of users but removes the edge cases. For example, setting `min_user_history_length_percentile to 0.05` and `max_user_history_length_percentile to 0.95` includes all users except those with history lengths at the bottom or top 5%. Default value: 0.99 Range: [0.0, 1.0] Value type: Float HPO tunable: No
Item exploration campaign configuration hyperparameters	

Name	Description
exploration_weight	Determines how frequently recommendations include items with less item interaction data or relevance . The closer the value is to 1.0, the more explorati on. At zero, no exploration occurs and recommend ations are based on current data (relevance). For more information see the section called "Campaign Config". Default value: 0.3 Range: [0.0, 1.0] Value type: Float HPO tunable: No

Name	Description
`exploration_item_age_cut_off`	Specify the maximum item age in days since the latest interaction across all items in the Item interactions dataset. This defines the scope of item exploration based on item age. Amazon Personalize determines an item's age based on its creation timestamp or, if creation timestamp data is missing, item interaction data. For more information how Amazon Personalize determines an item's age, see [Creation timestamp data](). To increase the items Amazon Personalize considers during exploration, enter a greater value. The minimum is 1 day and the default is 30 days. Recommendations might include items that are older than the item age cut off you specify. This is because these items are relevant to the user and exploration didn't identify them. Default value: 30.0 Range: Positive floats Value type: Float HPO tunable: No

Training with the User-Personalization recipe (console)

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

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

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

2. Create a Custom dataset group with a new schema and upload your dataset with impressions data. Optionally include CREATION_TIMESTAMP and Unstructured text metadata 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 Step 2: Preparing and importing data.
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 **Solution type**, choose **Item recommendation** to get item recommendations for your users.
7. For **Recipe**, choose **aws-user-personalization**. The **Solution configuration** section appears providing several configuration options.
8. In **Additional configuration**, if your Item 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 item interactions data that Amazon Personalize uses when training the model. For more information see Choosing the item interaction data used for training.
9. Optionally configure hyperparameters for your solution. For a list of User-Personalization recipe properties and hyperparameters, see Properties and hyperparameters.
10. Choose **Create and train solution** to start training. The **Dashboard** page displays.

    You can navigate to the solution details page to 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 operation.

3. For **Campaign configuration**, provide the following information:

   - **Exploration weight:** Configure how much to explore, where recommendations include items with less item interaction 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 item interaction data from the 10 days since the latest interaction in the dataset are considered during exploration.

     > **Note**
     >
     > Recommendations might include items without item interaction 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 5: Get recommendations 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.

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

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.

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 item interaction 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.

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

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.

Legacy user personalization recipes

> **Note**
>
> Legacy HRNN recipes are no longer available. This documentation is for reference purposes. 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.

The following are legacy USER_PERSONALIZATION recipes.

- HRNN recipe (legacy)
- HRNN-Coldstart recipe (legacy)
- HRNN-Metadata recipe (legacy)

HRNN recipe (legacy)

> **Note**
>
> Legacy HRNN recipes are no longer available. This documentation is for reference purposes. 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.

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.

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`	The number of hidden variables used in the model. *Hidden variables* recreate users' purchase history and item statistics to generate ranking scores. Specify a greater number of hidden dimensions when your Item 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

Name	Description
	performHPO to true when you call CreateSolution and CreateSolutionVersion operations. Default value: 43 Range: [32, 256] Value type: Integer HPO tunable: Yes
bptt	Determines whether to use the back-propagation through time technique. *Back-propagation through time* is a technique that updates weights in recurrent neural network-based algorithms. Use bptt for long-term credits to connect delayed rewards to early events. For example, a delayed reward can be a purchase made after several clicks. An early event can be an initial click. Even within the same event types, such as a click, it’s a good idea to consider long-term effects and maximize the total rewards. To consider long-term effects, use larger bptt values. Using a larger bptt value requires larger datasets and more time to process. Default value: 32 Range: [2, 32] Value type: Integer HPO tunable: Yes

Name	Description
`recency_mask`	Determines whether the model should consider the latest popularity trends in the Item interactions dataset. Latest popularity trends might include sudden changes in the underlying patterns of interaction events. To train a model that places more weight on recent events, set `recency_mask` to `true`. To train a model that equally weighs all past interactions, set `recency_mask` to `false`. To get good recommendations using an equal weight, you might need a larger training dataset. Default value: `True` Range: `True` or `False` Value type: Boolean HPO tunable: Yes
Featurization hyperparameters	

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

Name	Description
`max_user_history_length_percentile`	The maximum percentile of user history lengths to include in model training. *History length* is the total amount of data about a user. Use `max_user_history_length_percentile` to exclude a percentage of users with long history lengths because data for these users tend to contain noise. For example, a robot might have a long list of automated interactions. Removing these users limits noise in training. Choose an appropriate value after you review user history lengths using a histogram or similar tool. We recommend setting a value that retains the majority of users but removes the edge cases. For example, setting `min__user_history_length_percentile to 0.05` and `max_user_history_length_percentile to 0.95` includes all users except those with history lengths at the bottom or top 5%. Default value: 0.99 Range: [0.0, 1.0] Value type: Float HPO tunable: No

HRNN-Metadata recipe (legacy)

Note

Legacy HRNN recipes are no longer available. This documentation is for reference purposes.

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.

The HRNN-Metadata recipe predicts the items that a user will interact with. It is similar to the HRNN 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.

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`	The number of hidden variables used in the model. *Hidden variables* recreate users' purchase history and item statistics to generate ranking scores. Specify a greater number of hidden dimensions when your Item interactions dataset includes more complicated patterns. Using more hidden dimensions requires a larger dataset and more time to process. To decide on the optimal value, use HPO. To use HPO, set `performHPO` to `true` when you call CreateSolution and CreateSolutionVersion operations. Default value: 43 Range: [32, 256] Value type: Integer HPO tunable: Yes
`bptt`	Determines whether to use the back-propagation through time technique. *Back-propagation through time* is a technique that updates weights in recurrent neural network-based algorithms. Use `bptt` for long-term credits to connect delayed rewards to early events. For example, a delayed reward can be a purchase made after several clicks. An early event can be an initial click. Even within the same event types, such as a click, it’s a good idea to consider long-term effects and maximize the total rewards. To consider long-term effects, use larger `bptt` values. Using a larger `bptt` value requires larger datasets and more time to process. Default value: 32

Name	Description
	Range: [2, 32] Value type: Integer HPO tunable: Yes
`recency_mask`	Determines whether the model should consider the latest popularity trends in the Item interactions dataset. Latest popularity trends might include sudden changes in the underlying patterns of interaction events. To train a model that places more weight on recent events, set `recency_mask` to `true`. To train a model that equally weighs all past interactions, set `recency_mask` to `false`. To get good recommendations using an equal weight, you might need a larger training dataset. Default value: `True` Range: `True` or `False` Value type: Boolean HPO tunable: Yes
Featurization hyperparameters	

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

Name	Description
`max_user_history_length_percentile`	The maximum percentile of user history lengths to include in model training. *History length* is the total amount of data about a user. Use `max_user_history_length_percentile` to exclude a percentage of users with long history lengths because data for these users tend to contain noise. For example, a robot might have a long list of automated interactions. Removing these users limits noise in training. Choose an appropriate value after you review user history lengths using a histogram or similar tool. We recommend setting a value that retains the majority of users but removes the edge cases. For example, setting `min__user_history_length_percentile to 0.05` and `max_user_history_length_percentile to 0.95` includes all users except those with history lengths at the bottom or top 5%. Default value: 0.99 Range: [0.0, 1.0] Value type: Float HPO tunable: No

HRNN-Coldstart recipe (legacy)

> **Note**
>
> Legacy HRNN recipes are no longer available. This documentation is for reference purposes.

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.

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 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 Item 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 Choosing a recipe.

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.

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`	The number of hidden variables used in the model. *Hidden variables* recreate users' purchase history and item statistics to generate ranking scores. Specify a greater number of hidden dimensions when your Item interactions dataset includes more complicated patterns. Using more hidden dimensions requires a larger dataset and more time to process. To decide on the optimal value, use HPO. To use HPO, set `performHPO` to `true` when you call CreateSolution and CreateSolutionVersion operations.

Name	Description
	Default value: 149 Range: [32, 256] Value type: Integer HPO tunable: Yes
`bptt`	Determines whether to use the back-propagation through time technique. *Back-propagation through time* is a technique that updates weights in recurrent neural network-based algorithms. Use `bptt` for long-term credits to connect delayed rewards to early events. For example, a delayed reward can be a purchase made after several clicks. An early event can be an initial click. Even within the same event types, such as a click, it's a good idea to consider long-term effects and maximize the total rewards. To consider long-term effects, use larger `bptt` values. Using a larger `bptt` value requires larger datasets and more time to process. Default value: 32 Range: [2, 32] Value type: Integer HPO tunable: Yes

Name	Description
recency_mask	Determines whether the model should consider the latest popularity trends in the Item interactions dataset. Latest popularity trends might include sudden changes in the underlying patterns of interaction events. To train a model that places more weight on recent events, set `recency_mask` to `true`. To train a model that equally weighs all past interactions, set `recency_mask` to `false`. To get good recommendations using an equal weight, you might need a larger training dataset. Default value: `True` Range: `True` or `False` Value type: Boolean HPO tunable: Yes
Featurization hyperparameters	
cold_start_max_interactions	The maximum number of user-item interactions an item can have to be considered a cold item. Default value: 15 Range: Positive integers Value type: Integer HPO tunable: No

Name	Description
`cold_start_max_duration`	The maximum duration in days relative to the starting point for a user-item interaction to be considered a cold start item. To set the starting point of the user-item interaction, set the `cold_start_relative_from` hyperparameter. Default value: 5.0 Range: Positive floats Value type: Float HPO tunable: No
`cold_start_relative_from`	Determines the starting point for the HRNN-Cold start recipe to calculate `cold_start_max_duration`. To calculate from the current time, choose `currentTime`. To calculate `cold_start_max_duration` from the timestamp of the latest item in the Item interactions dataset, choose `latestItem`. This setting is useful if you frequently add new items. Default value: `latestItem` Range: `currentTime`, `latestItem` Value type: String HPO tunable: No

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

Name	Description
max_user_history_length_percentile	The maximum percentile of user history lengths to include in model training. *History length* is the total amount of data about a user. Use max_user_history_length_percentile to exclude a percentage of users with long history lengths because data for these users tend to contain noise. For example, a robot might have a long list of automated interactions. Removing these users limits noise in training. Choose an appropriate value after you review user history lengths using a histogram or similar tool. We recommend setting a value that retains the majority of users but removes the edge cases. For example, setting min__user_history_length_percentile to 0.05 and max_user_history_length_percentile to 0.95 includes all users except those with history lengths at the bottom or top 5%. Default value: 0.99 Range: [0.0, 1.0] Value type: Float HPO tunable: No

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 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.

performAutoML	recipeArn	solutionConfig	Result
true	omit	omitted	Amazon Personalize chooses the recipe
true	omit	`autoMLConfig : metricName` and `recipeList` specified	Amazon Personalize chooses a recipe from the list that optimizes the metric
omit	specified	omitted	You specify the recipe
omit	specified	specified	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`.

POPULAR_ITEMS

To recommend trending or popular items, such as breaking news articles or popular social media content, use a POPULAR_ITEMS recipe. To generate recommendations for items that are rapidly becoming more popular with your users, use the Trending-Now recipe recipe. To generate a baseline for comparison purposes, we recommend using the Popularity-Count recipe. This POPULAR_ITEMS recipe recommends the most popular items based on counting interactions.

POPULAR_ITEMS recipes are as follows:

- Trending-Now recipe
- Popularity-Count recipe

Trending-Now recipe

The Trending-Now recipe (aws-trending-now) generates recommendations for items that are rapidly becoming more popular with your users. You might use the Trending-Now recipe if items gaining in popularity are more relevant to your customers. For example, your customers might highly value what other users are interacting with. Common uses include recommending viral social media content, breaking news articles, or recent sports videos.

Trending-Now automatically identifies the top trending items by calculating the increase in interactions that each item has over configurable intervals of time. The items with highest rate of increase are considered trending items. The time is based on timestamp data in your Item interactions dataset. The items considered come from the interactions data you imported in bulk and incrementally. You don't have to manually create a new solution version for Trending-Now to consider new items in interactions data.

You can specify the time interval by providing a `Trend discovery frequency` when you create your solution. For example, if you specify `30 minutes` for `Trend discovery frequency`, for every 30 minutes of data, Amazon Personalize identifies the items with the greatest rate of increase in interactions since the last evaluation. Possible frequencies include 30 minutes, 1 hour, 3 hours, and 1 day. Choose a frequency that aligns with the distribution of your interactions data. Missing data over the interval you choose can reduce recommendation accuracy. If you import zero interactions over the last two time intervals, Amazon Personalize recommends only popular items instead of trending items.

With Trending-Now, you call the GetRecommendations operation or get recommendations on the **Test campaign** page of the Amazon Personalize console. Amazon Personalize returns the top trending items. You pass a `userId` in your request only if you apply a filter that requires it. With the GetRecommendations API, you can configure the number of trending items returned with the `numResults` parameter. You can't get batch recommendations with the Trending-Now recipe.

To use Trending-Now, you must create an Item interactions dataset with at least 1000 unique historical and event interactions combined (after filtering by eventType and eventValueThreshold, if provided). When generating trending item recommendations, Trending-Now doesn't use data in Items or Users datasets. However, you can still filter recommendations based on data in these datasets. For more information, see Filtering recommendations and user segments.

Topics

- Properties and hyperparameters
- Creating a solution (SDK for Python (Boto3))
- Sample Jupyter notebook

Properties and hyperparameters

The Trending-Now recipe has the following properties:

- **Name** – `aws-trending-now`
- **Recipe Amazon Resource Name (ARN)** – `arn:aws:personalize:::recipe/aws-trending-now`
- **Algorithm ARN** – `arn:aws:personalize:::algorithm/aws-trending-now-custom`

For more information, see Choosing a recipe.

The following table describes the hyperparameters for the Trending-Now 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.

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
Feature transformation hyperparameters	
`Trend discovery frequency`	Specify how often Amazon Personalize evaluates your interactions data and identifies trending items. For example, if you specify `30 minutes` for `Trend discovery frequency` , every 30 minutes Amazon Personalize identifies items with the greatest rate of increase in interactions over 30-minute intervals. Available frequencies include 30 minutes, 1 hour, 3 hours, and 1 day. Choose a frequency that aligns with the distribution of your interactions data. Missing data over the interval you choose can reduce recommendation accuracy. If you don't specify a value, the default is every 2 hours. Default value: 2 hours Possible values: 30 minutes, 1 hour, 3 hour, and 1 day. Value type: String HPO tunable: No

Creating a solution (SDK for Python (Boto3))

The following code shows how to create a solution with the Trending-Now recipe using the SDK for Python (Boto3). Possible values for `trend_discovery_frequency` are `30 minutes`, `1 hour`, `3 hours`, and `1 day`. For information about creating a solution with the console, see Creating a solution (console).

```
import boto3

personalize = boto3.client("personalize")

create_solution_response = personalize_client.create_solution(
    name="solution name",
```

```
    recipeArn="arn:aws:personalize:::recipe/aws-trending-now",
    datasetGroupArn="dataset group ARN",
    solutionConfig={
        "featureTransformationParameters": {
            "trend_discovery_frequency": "1 hour"
         }
    }
)
print(create_solution_response['solutionArn'])
```

Sample Jupyter notebook

For a sample Jupyter notebook that shows how to use the Trending-Now recipe, see trending_now_example.ipynb in the Amazon Personalize samples GitHub repository.

Popularity-Count recipe

Popularity-Count recommends the most popular items based your interactions data. The most popular items are the items with the most interactions data from 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 Evaluating a solution version with metrics.

After you create a solution version, make sure you keep your solution version and data up to date. With Popularity-Count, you must manually create a new solution version (retrain the model) for Amazon Personalize to consider new items for recommendations and update the model with your user's most recent behavior. Then you must update any campaign using the solution version. For more information, see Maintaining recommendation relevance.

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.

PERSONALIZED_RANKING

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

Personalized-Ranking

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 Item 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. For example, with Personalized-Ranking, Amazon Personalize can re-rank search results that you generate with OpenSearch.

To train a model, the Personalized-Ranking recipe uses the data in your Item interactions dataset, and if you created them, the Items dataset and Users dataset in your dataset group (these datasets are optional). With Personalized-Ranking, your Items dataset can include Unstructured text metadata and your Item interactions dataset can include Contextual metadata. To get a personalized ranking, use the GetPersonalizedRanking API.

After you create a solution version, make sure you keep your solution version and data up to date. With Personalized-Ranking, you must manually create a new solution version (retrain the model) for Amazon Personalize to consider new items for recommendations and update the model with your user's most recent behavior. Then you must update any campaign using the solution version. For more information, see Maintaining recommendation relevance.

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.

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`	The number of hidden variables used in the model. *Hidden variables* recreate users' purchase history and item statistics to generate ranking scores. Specify a greater number of hidden dimensions when your Item interactions dataset includes more complicated patterns. Using more hidden dimensions requires a larger dataset and more time to process. To decide

Name	Description
	on the optimal value, use HPO. To use HPO, set `performHPO` to `true` when you call CreateSolution and CreateSolutionVersion operations. Default value: 149 Range: [32, 256] Value type: Integer HPO tunable: Yes
`bptt`	Determines whether to use the back-propagation through time technique. *Back-propagation through time* is a technique that updates weights in recurrent neural network-based algorithms. Use `bptt` for long-term credits to connect delayed rewards to early events. For example, a delayed reward can be a purchase made after several clicks. An early event can be an initial click. Even within the same event types, such as a click, it's a good idea to consider long-term effects and maximize the total rewards. To consider long-term effects, use larger `bptt` values. Using a larger `bptt` value requires larger datasets and more time to process. Default value: 32 Range: [2, 32] Value type: Integer HPO tunable: Yes

Name	Description
recency_mask	Determines whether the model should consider the latest popularity trends in the Item interactions dataset. Latest popularity trends might include sudden changes in the underlying patterns of interaction events. To train a model that places more weight on recent events, set `recency_mask` to `true`. To train a model that equally weighs all past interactions, set `recency_mask` to `false`. To get good recommendations using an equal weight, you might need a larger training dataset. Default value: `True` Range: `True` or `False` Value type: Boolean HPO tunable: Yes
Featurization hyperparameters	

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

Name	Description
`max_user_history_length_percentile`	The maximum percentile of user history lengths to include in model training. *History length* is the total amount of data about a user. Use `max_user_history_length_percentile` to exclude a percentage of users with long history lengths because data for these users tend to contain noise. For example, a robot might have a long list of automated interactions. Removing these users limits noise in training. Choose an appropriate value after you review user history lengths using a histogram or similar tool. We recommend setting a value that retains the majority of users but removes the edge cases. For example, setting `min__user_history_length_percentile to 0.05` and `max_user_history_length_percentile to 0.95` includes all users except those with history lengths at the bottom or top 5%. Default value: 0.99 Range: [0.0, 1.0] Value type: Float HPO tunable: No

Personalized-Ranking sample notebook

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

RELATED_ITEMS

> **Note**
>
> All RELATED_ITEMS recipes use interactions data. Choose the Similar-Items recipe if you have also have item metadata and want Amazon Personalize to use it to find similar items. Or choose the SIMS recipe if you want to configure more hyperparameters for the model.

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
- SIMS recipe

Similar-Items

The Similar-Items recipe generates recommendations for items that are similar to an item you specify. It calculates similarity based on both interactions data and, if you provide it, item metadata. If Amazon Personalize can't find the item ID that you specify in your recommendation request, the recipe returns popular items as recommendations. Similar-Items doesn't use data in a Users dataset when generating recommendations. However, you can still filter recommendations based on data in a Users dataset. For more information, see Filtering recommendations and user segments.

SIMS

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 Item 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.

Similar-Items recipe

> **Note**
>
> All RELATED_ITEMS recipes use interactions data. Choose Similar-Items if you have also have item metadata and want Amazon Personalize to use it to find similar items. Or choose the SIMS recipe if you want to configure more hyperparameters for the model.

The Similar-Items (aws-similar-items) recipe generates recommendations for items that are similar to an item you specify. Use Similar-Items to help customers discover new items in your catalog based on their previous behavior and item metadata. Recommending similar items can increase user engagement, click-through rate, and conversion rate for your application.

Similar-Items calculates similarity based on interactions data and any item metadata you provide. It takes into account the co-occurrence of the item in user histories in your Interaction dataset, and any item metadata similarities. For example, with Similar-Items, Amazon Personalize could recommend items customers frequently bought together with a similar style (Categorical metadata), or movies that different users also watched with a similar description (Unstructured text metadata).

With Similar-Items, you provide an item ID in a GetRecommendations 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 Batch recommendations and user segments (custom resources)). When you get similar items, you can filter the items based on an attribute of the item you specify in your request. You do this by adding a `CurrentItem.attribute` element to your filter. For an example, see item data filter examples.

To use Similar-Items, you must create an Item interactions dataset with at least 1000 unique historical and event interactions (combined). For more accurate predictions, we recommend that you also create an Items dataset and import metadata about items in your catalog. Similar-Items doesn't use data in a Users dataset when generating recommendations. You can still filter recommendations based on data in a Users dataset. For more information, see Filtering recommendations and user segments.

If you have an Items dataset with textual data and item title data, you can generate themes for related items in batch recommendations. For more information, see Batch recommendations with themes from Content Generator

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.

After you create a solution version, make sure you keep your solution version and data up to date. With Similar-Items, you must manually create a new solution version (retrain the model) for Amazon Personalize to consider new items for recommendations and update the model with your user's most recent behavior. Then you must update any campaign using the solution version. For more information, see Maintaining recommendation relevance.

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 Choosing a recipe.

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.

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	

Name	Description
popularity_discount_factor	Configure how popularity influences recommend ations. Specify a value closer to zero to include more popular items. Specify a value closer to one for less emphasis on popularity. Default value: 0.0 Range: [0.0, 1.0] Value type: Float HPO tunable: No
item_id_hidden_dim	The number of hidden variables Amazon Personalize uses to model item ID embeddings based on interacti ons data. *Hidden variables* recreate users' purchase history and item statistics to generate ranking scores. To use item_id_hidden_dim , 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 Item interacti ons dataset. Using a greater maximum value requires more time to process. To use HPO, set performHPO to true when you call the CreateSolution operation. Default value: 100 Range: [30, 200] Value type: Integer HPO tunable: Yes

Name	Description
`item_metadata_hidden_dim`	The number of hidden variables Amazon Personalize uses to model item metadata. To use `item_metadata_hidden_dim` , 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 Item interactions dataset. Using a greater maximum requires more time to process. To use HPO, set `performHPO` to `true` when you call the CreateSolution operation. Default value: 100 Range: [30, 200] Value type: Integer HPO tunable: Yes

SIMS recipe

> **Note**
>
> All RELATED_ITEMS recipes use interactions data. Choose SIMS if you want to configure more hyperparameters for the model. Choose the Similar-Items recipe if you have item metadata and want Amazon Personalize to use it to find similar items.

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 Item 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.

When you get similar item recommendations, you can filter the items based on an attribute of the item you specify in your request. You do this by adding a `CurrentItem.attribute` element to your filter. For an example, see item data filter examples.

To use SIMS, you must create an Item interactions dataset with at least 1000 unique historical and event interactions (combined). SIMS doesn't use data in a Users or Items dataset when generating recommendations. You can still filter recommendations based on data in a these datasets. For more information, see Filtering recommendations and user segments.

If there isn't sufficient user behavior data for an item or the item ID you provide isn't found, SIMS recommends popular items. After you create a solution version, make sure you keep your solution version and data up to date. With SIMS, you must manually create a new solution version (retrain the model) for Amazon Personalize to consider new items for recommendations and update the model with your user's most recent behavior. Then you must update any campaign using the solution version. For more information, see Maintaining recommendation relevance.

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.

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	
`popularity_discount_factor`	Configure how popularity influences recommendations. Specify a value closer to zero to include more popular items. Specify a value closer to one for less emphasis on popularity. Default value: 0.5 Range: [0.0, 1.0] Value type: Float HPO tunable: Yes
`min_cointeraction_count`	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. Default value: 3 Range: [0, 10] Value type: Integer HPO tunable: Yes
Featurization hyperparameters	
`min_user_history_length_percentile`	The minimum percentile of user history lengths to include in model training. *History length* is the total amount of available data on a user. Use `min_user_history_length_percentile` to exclude a percentage of users with short history lengths. Users with a short history often show patterns based on item popularity instead of the user's personal

Name	Description
	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. Default value: 0.005 Range: [0.0, 1.0] Value type: Float HPO tunable: No

Name	Description
max_user_history_length_percentile	The maximum percentile of user history lengths to include in model training. History length is the total amount of available data on a user. Use max_user_history_length_percentile to exclude a percentage of users with long history lengths. Users with a long history tend to contain noise. For example, a robot might have a long list of automated interactions. Removing these users limits noise in training. Choose an appropriate value after you review user history lengths using a histogram or similar tool. We recommend setting a value that retains the majority of users but removes the edge cases. For example, min_hist_length_percentile = 0.05 and max_hist_length_percentile = 0.95 includes all users except ones with history lengths at the bottom or top 5%. Default value: 0.995 Range: [0.0, 1.0] Value type: Float HPO tunable: No

Name	Description
`min_item_interaction_count_percentile`	The minimum percentile of item interaction counts to include in model training. Use `min_item_interaction_count_percentile` to exclude a percentage of items with a short history of interactions. Items with a short history often are new items. Removing them can train models with more focus on items with a known history. Choose an appropriate value after you review user history lengths, using a histogram or similar tool. We recommend setting a value that retains the majority of items, but removes the edge cases. Default value: 0.01 Range: [0.0, 1.0] Value type: Float HPO tunable: No

Name	Description
`max_item_interaction_count_percentile`	The maximum percentile of item interaction counts to include in model training. Use `max_item_interaction_count_percentile` 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. For example, `min_item_interaction_count_percentile = 0.05` and `max_item_interaction_count_percentile = 0.95` includes all items except ones with an interaction count at the bottom or top 5%. Default value: 0.9 Range: [0.0, 1.0] Value type: Float HPO tunable: No

SIMS sample notebook

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

PERSONALIZED_ACTIONS

To recommend the next best action for your users in real time, such as signing up for your loyalty program, downloading your app, or applying for a credit card, use a PERSONALIZED_ACTIONS recipe. Recommending the next best action can increase customer loyalty, generate more revenue, and improve your users' experience.

PERSONALIZED_ACTIONS recipes are as follows:

- Next-Best-Action recipe

Next-Best-Action recipe

The Next-Best-Action (aws-next-best-action) recipe generates real-time recommendations for the next best actions for your users. The next best action for a user is the action that they will most likely take. For example, enrolling in your loyalty program, downloading your app, or applying for a credit card.

With Next-Best-Action, you can provide personalized action recommendations for your users as they use your application. Suggesting the right action for a user can result in more users taking your actions. Depending on the actions you want to recommend, you can increase customer loyalty, generate more revenue, and improve the user experience of your application. For a use case example that describes how personalized action recommendations can benefit an ecommerce application, see Use case example.

Amazon Personalize predicts the next best action from the actions you import into your Actions dataset. It identifies the actions that a user will most likely take based on their interactions with actions and items. If your action data includes the value of the action, Amazon Personalize accounts for the action's value. If a user is equally likely to take two different actions, Amazon Personalize ranks the action with the greater value higher. For more information about the data the Next-Best-Action recipe uses, see Required and optional datasets.

When you get real-time action recommendations for a user, Amazon Personalize returns a list of actions that the user will most likely take within a configurable period of time (the `action optimization period`). For example, the actions they will most likely take in the next 14 days. The list is sorted in descending order by propensity score. This score represents the likelihood that the user will take the action.

When you create a solution with the Next-Best-Action recipe, you can configure the window of time Amazon Personalize uses when predicting actions by using the `action optimization period` featurization hyperparameter. For more information, see Properties and hyperparameters.

Topics

Use case example

Suggesting the right action for a user can result in more users taking your actions. Depending on the actions you want to recommend, you can potentially increase customer loyalty, generate more revenue, and improve the user experience of your application.

For example, you might have an ecommerce application that suggests the following different actions:

- Subscribe to loyalty program
- Download mobile app
- Purchase in *Jewelry* category
- Purchase in *Beauty and grooming* category

You might have a user who frequently shops at your site and has repeatedly taken the *Jewelry* and *Beauty and grooming* purchase actions. For this user, Amazon Personalize action recommendations and their scores might include the following:

- Subscribe to loyalty program

  Propensity score – 1.00
- Purchase in *Jewelry* category

  Propensity score – 0.86
- Purchase in *Beauty and grooming* category

  Propensity score – 0.85

With these action recommendations, you know to prompt the user to enroll in your loyalty program. This action has the highest propensity score and it is the action the user will most likely take. This is because the user frequently shops at your store and is likely to engage with the benefits from your loyalty program.

Recipe features

The Next-Best-Action recipe uses the following Amazon Personalize recipe features when generating action recommendations:

- Real-time personalization: Amazon Personalize uses real-time personalization to update and adapt action recommendations according to a user's evolving interest. For more information, see Real-time personalization.
- Exploration: With exploration, recommendations include new actions or actions with less interactions data. For more information about exploration, see Exploration.
- Automatic updates: With automatic updates, Amazon Personalize automatically updates the latest model (solution version) every two hours to include new actions in recommendations through exploration. For more information, see Automatic updates.

Required and optional datasets

To use the Next-Best-Action recipe, you must create the following datasets:

- Actions: You import data about your actions, such as their value, into an Amazon Personalize Actions dataset.

  In your actions data, you can provide an EXPIRATION_TIMESTAMP for each action. If an action has expired, Amazon Personalize won't include it in recommendations. You can also provide a REPEAT_FREQUENCY for each action. This indicates how long Amazon Personalize should wait before recommending an action again after a user interacts with it. For information about the data an Actions dataset can store, see Actions dataset.
- Item interactions: Your Item interactions dataset must have at minimum 1000 item interactions. Amazon Personalize uses item interactions to understand your users' current state and their interests. For information about the item interactions data, see Item interactions dataset.

The following datasets are optional:

- Action interactions dataset: An *action interaction* is an interaction involving a user and an action in your Actions dataset. You can import Taken, Not taken, and Viewed action interactions. Although this data is optional, we recommend that you import action interaction data for quality recommendations. If you don't have action interaction data, you can create an empty Action interactions dataset and record your customers' interactions with actions by using the PutActionInteractions API operation.

Until you import action interaction data, Amazon Personalize recommends actions in your Actions dataset without personalization, and propensity scores are 0.0.

For information about the action interactions data you can import, see Action interactions dataset. For information about recording action interaction events, see Recording action interaction events.

> **Note**
>
> With Next-Best-Action, Amazon Personalize doesn't use impressions data or contextual metadata in an Action interactions dataset.

- Users: Amazon Personalize uses any data in your Users dataset to better understand your users and their interests. You can also use data in a Users dataset to filter action recommendations. For information about the user data you can import, see Users dataset.
- Items: Amazon Personalize uses any data in your Items dataset along with your Item interactions dataset to identify connections and patterns in their behavior. This helps Amazon Personalize understand your users and their interests. For information about the item data you can import, see Items dataset.

Properties and hyperparameters

The Next-Best-Action recipe doesn't support hyperparameter optimization. The Next-Best-Action recipe has the following properties:

- **Name** – `aws-next-best-action`
- **Recipe Amazon Resource Name (ARN)** – `arn:aws:personalize:::recipe/aws-next-best-action`
- **Algorithm ARN** – `arn:aws:personalize:::algorithm/aws-next-best-action`

The following table describes the featurization hyperparameters for the aws-next-best-action recipe. A *hyperparameter* is an algorithm parameter that you can adjust to improve model performance. Featurization hyperparameters control how to filter the data to use in training.

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**: Whether the parameter can participate in HPO

Name	Description
Featurization hyperparameters	
`action_optimization_period`	The window of time Amazon Personalize uses when predicting the next best actions for a user. For example, the actions the user will most likely take in the next 14 days. If you don't have much action interaction data, specify a larger value. If you aren't sure what value to specify, use the default. Default value: 14 Range: [7, 28] Value type: Integer HPO tunable: No

USER_SEGMENTATION

USER_SEGMENTATION recipes generate segments of users based on item input data. Each user segment is sorted in descending order based on the probability that each user will interact with items in your inventory. Use a USER_SEGMENTATION recipe to create segments of users who will most likely interact with your catalog based on their item or item attribute preferences. For example, you might want to create a targeted marketing campaign for users that would most likely watch a particular movie or purchase a particular products by brand.

Item-Affinity

The Item-Affinity (aws-item-affinity) recipe is a USER_SEGMENTATION recipe that creates a user segment for each item that you specify.

To train a model, the Item-Affinity recipe uses the Interactions and Items datasets in your dataset group. To create user segments, you train a solution version with the Item-Affinity recipe, and then create a batch segment job.

Item-Attribute-Affinity

The Item-Attribute-Affinity (aws-item-attribute-affinity) recipe is a USER_SEGMENTATION recipe that creates a user segment for each item attribute that you specify.

To train a model, the Item-Attribute-Affinity recipe uses the Interactions dataset and Item dataset from a dataset group. To create user segments, you train a solution version with the Item-Attribute-Affinity recipe, and then create a batch segment job.

Item-Affinity recipe

The Item-Affinity (aws-item-affinity) recipe is a USER_SEGMENTATION recipe that creates a user segment (group of users) for each item that you specify. These are the users Amazon Personalize predicts will most likely interact with each item. Use Item-Affinity to learn more about your users and take actions based on their respective user segments.

For example, you might want to create a marketing campaign for your retail application based on user preferences for items in your catalog. Item-Affinity would create a user segment for each item based on data in your Interactions and Items datasets. You could use this to promote different items to different user segments based on the likelihood that they will take an action (for example, click an item or purchase an item). Other uses might include cross-selling products to different sets of users or identifying prospective job applicants.

To get user segments based on items, you create a solution and a solution version with the Item-Affinity recipe, then add a list of items in JSON format to an Amazon S3 bucket and create a batch segment job. Amazon Personalize outputs a user segment for each item to your output location in Amazon S3. Your input data can have a maximum of 500 items to get user segments for. For information about preparing input data for a batch segment job, see Preparing input data for batch recommendations.

You must have an Item interactions dataset to use Item-Affinity. Items and Users datasets are optional. You can get user segments with batch segment jobs. For more information, see [Batch recommendations and user segments (custom resources)](#).

After you create a solution version, make sure you keep your solution version and data up to date. With Item-Affinity, you must create a new solution version for Amazon Personalize to consider new users for user segments and update the model with your users' most recent behavior. To get a user segment for an item, the item must have been present when you created the solution version.

The Item-Affinity recipe has the following properties:

- **Name** – `aws-item-affinity`
- **Recipe Amazon Resource Name (ARN)** – `arn:aws:personalize:::recipe/aws-item-affinity`
- **Algorithm ARN** – `arn:aws:personalize:::algorithm/aws-item-affinity`
- **Feature transformation ARN** – `arn:aws:personalize:::feature-transformation/item-affinity`
- **Recipe type** – `USER_SEGMENTATION`

The following table describes the hyperparameters for the Item-Affinity recipe. A *hyperparameter* is an algorithm parameter that you adjust to improve model performance. Algorithm hyperparameters control how the model performs. You can't use hyperparameter optimization (HPO) with the Item-Affinity recipe.

The table also provides the following information for each hyperparameter:

- **Range**: [lower bound, upper bound]
- **Value type**: Integer, Continuous (float), Categorical (Boolean, list, string)

Name	Description
Algorithm hyperparameters	
`hidden_dimension`	The number of hidden variables used in the model. *Hidden variables* recreate users' purchase history and item statistics to generate ranking scores. Specify a greater number of hidden dimensions when your

Name	Description
	Interactions dataset includes more complicated patterns. Using more hidden dimensions requires a larger dataset and more time to process. Default value: 149 Range: [32, 256] Value type: Integer

Item-Attribute-Affinity recipe

The Item-Attribute-Affinity (aws-item-attribute-affinity) recipe is a USER_SEGMENTATION recipe that creates a user segment (group of users) for each item attribute that you specify. These are the users Amazon Personalize predicts will most likely interact with items with the particular attribute. Use Item-Attribute-Affinity to learn more about your users and take actions based on their respective user segments.

For example, you might want to create a marketing campaign for your retail application based on user preferences for shoe types in your catalog. Item-Attribute-Affinity would create a user segment for each shoe type based data in your Interactions and Items datasets. You could use this to promote different shoes to different user segments based on the likelihood that they will take an action (for example, click a shoe or purchase a shoe). Other uses might include promoting different movie genres to different users or identifying prospective job applicant based on job type.

To get user segments based on item attributes, you create a solution and a solution version with the Item-Attribute-Affinity recipe, then add a list of item attributes in JSON format to an Amazon S3 bucket and create a batch segment job. Amazon Personalize outputs a user segment for each item to your output location in Amazon S3. Your input data can have a maximum of 10 queries, where each query is one or more item attributes. For information about preparing input data for a batch segment job, see Preparing input data for batch recommendations.

You must have an Item interactions dataset and an Items dataset to use Item-Attribute-Affinity. Your Items dataset must have at least one column that is a non-textual, non-reserved metadata column. You can get user segments with batch segment jobs. For more information, see Batch recommendations and user segments (custom resources).

After you create a solution version, make sure you keep your solution version and data up to date. With Item-Attribute-Affinity, you must create a new solution version for Amazon Personalize to consider new users for user segments and update the model with your users' most recent behavior. To get a user segment for an item attribute, the item attribute must have been present when you created the solution version.

The Item-Attribute-Affinity recipe has the following properties:

- **Name** – `aws-item-attribute-affinity`
- **Recipe Amazon Resource Name (ARN)** – `arn:aws:personalize:::recipe/aws-item-attribute-affinity`
- **Algorithm ARN** – `arn:aws:personalize:::algorithm/aws-item-attribute-affinity`
- **Feature transformation ARN** – `arn:aws:personalize:::feature-transformation/item-attribute-affinity`
- **Recipe type** – `USER_SEGMENTATION`

The following table describes the hyperparameters for the Item-Attribute-Affinity recipe. A *hyperparameter* is an algorithm parameter that you can adjust to improve model performance. Algorithm hyperparameters control how the model performs. You can't use hyperparameter optimization (HPO) with the Item-Attribute-Affinity recipe.

The table also provides the following information for each hyperparameter:

- **Range**: [lower bound, upper bound]
- **Value type**: Integer, Continuous (float), Categorical (Boolean, list, string)

Name	Description
Algorithm hyperparameters	
`hidden_dimension`	The number of hidden variables used in the model. *Hidden variables* recreate users' purchase history and item statistics to generate ranking scores. Specify a greater number of hidden dimensions when your Interactions dataset includes more complicated

Name	Description
	patterns. Using more hidden dimensions requires a larger dataset and more time to process. Default value: 149 Range: [32, 256] Value type: Integer

Readiness checklist

After you review how Amazon Personalize works and complete the getting started exercise, you can start getting ready to use Amazon Personalize with your own data. This checklist provides lists of Amazon Personalize features, requirements, and data guidance. It can help you plan, or you can use it as a reference as you create resources in Amazon Personalize.

Topics

Have you matched your use cases to Amazon Personalize resources?

Amazon Personalize recommendations can address the following use cases:

- Generating personalized recommendations for a user
- Recommending similar or related items
- Recommending trending or popular items
- Recommending the next best actions for a user
- Re-ordering by relevance (only with custom resources)
- Generating user segments (only with custom resources)

Amazon Personalize features domain based resources and custom resources configured for these use cases. You start by creating a Domain dataset group or a Custom dataset group:

- With a *Domain dataset group*, you create resources that are pre-configured and optimized for for the VIDEO_ON_DEMAND or ECOMMERCE domains.

If you have a streaming video or e-commerce application, we recommend that you start with a Domain dataset group. You can still add custom resources, such as solutions and solution versions trained for custom use cases. And you can still use custom resources to get batch recommendations.

- With a *Custom dataset group*, you choose a recipe that matches your use case. You then train and deploy only configurable solutions and solution versions (trained Amazon Personalize recommendation models). When ready, you can deploy the solution version in a campaign for real-time recommendations. Or you can get batch recommendations without a campaign.

  If you don't have a streaming video or e-commerce application, we recommend that you create a Custom dataset group. Otherwise, start with a Domain dataset group and adding custom resources as necessary.

For information on the use cases and custom recipes available in Amazon Personalize, see Domain use cases and custom recipes.

Do you have enough item interaction data?

For all use cases and recipes, you must have at minimum 1,000 item interactions for 25 unique users with at least two interactions each. For quality recommendations, we recommend that you have at minimum 50,000 item interactions from at least 1,000 users with two or more item interactions each.

If you aren't sure if you have enough data, you can import and analyze it with the Amazon Personalize console. For more information, see Analyzing data in datasets.

Do you have a real-time event streaming architecture in place?

If you don't have enough item interaction data, you can use Amazon Personalize to collect additional real-time event data. With some recipes and use cases, Amazon Personalize can learn from your user's most recent activity and update recommendations as they use your application.

For information about recording events, including how events impact recommendations, a list of third-party event tracking services, and sample implementations, see Recording events.

Is your data optimized for Amazon Personalize?

We recommend you check for the following in your data:

- Check for missing values. We recommend that a minimum of 70% of your records have data for every attribute. We recommend columns that allow null values be at least 70% complete.
- Fix any inaccuracies or issues in your data, such as inconsistent naming conventions, duplicate categories for an item, mismatched IDs across datasets, or duplicate IDs. These issues can negatively impact recommendations or lead to unexpected behavior. For example, you might have both “N/A” and “Not Applicable” in your data, but filter out recommendations based on only “N/A”. Items marked "Not Applicable" would not be removed by the filter.
- If an item, user, or action can have multiple categories, such as a movie with multiple genres, combine the categorical values into one attribute and separate each value with the | operator. For example, a movie’s GENRES data might be Action | Adventure | Thriller.
- Avoid having more than 1000 possible categories for a column (unless the column contains data for only filtering purposes).

For a complete list of data recommendations, and instructions on how you can use Amazon Personalize to identify issues, see Analyzing data in datasets.

Do you collect optional data that can improve recommendations?

The following data can help improve your recommendation relevance.

- Event type (required for all Domain dataset group use cases)
- Event value
- Contextual metadata
- Item and user metadata
- Action interaction data (used by only PERSONALIZED_ACTIONS recipes)

For more information on the types of data Amazon Personalize can use, see Types of data Amazon Personalize can use.

Do you have a plan to test your recommendations?

You can use A/B testing to compare the results of different groups of users interacting with recommendations from different models. A/B testing can help you compare different recommendation strategies and see if recommendations are helping you achieve your business goals. For more information, see Measuring recommendation impact with A/B testing.

Do you have additional business goals?

In some cases, you might have goals in addition to generating relevant recommendations for your users. For example, you might want to maximize revenue, or promote certain types of items from a certain category. The following Amazon Personalize features can help:

- Promotions: You can use promotions to make sure a certain percentage of items satisfy your business requirements. For more information, see Promoting items in recommendations.
- Optimizing for business objective: For some Custom dataset group recipes, you can optimize a solution for a custom objective, such as maximizing streaming minutes or increasing revenue. For more information, see Optimizing a solution for an additional objective.
- Filtering recommendations. Use filters to apply business rules to recommendations. You can use filters to include or exclude certain types of items from recommendations. For more information, see Filtering recommendations and user segments.

Amazon Personalize workflow

After you review the Readiness checklist, you are ready to start completing the Amazon Personalize workflow:

1. **Create a dataset group**

   A dataset group is a container for Amazon Personalize resources. The type of dataset group you create determines the resources you can create in step 3 of the Amazon Personalize workflow.

   - With a *Domain dataset group*, you can create recommenders configured for VIDEO_ON_DEMAND or ECOMMERCE domain use cases. You use the recommenders to get recommendations. Amazon Personalize manages their configuration, training, and updates. If you start with a Domain dataset group, you can still add custom resources.
   - With a *Custom dataset group*, you can create only custom resources. These including solutions, solution versions, and campaigns. For these resources, you have more control over configurations, updates, and retraining.

2. **Prepare and import data**

   You import item interaction, action interaction, item, user, and action records into *datasets* (Amazon Personalize containers for data). You can import records in bulk or individually. When you import bulk data, you can use Amazon SageMaker Data Wrangler to import data from 40+ sources and prepare it for Amazon Personalize. For more information, see Preparing and importing data using Amazon SageMaker Data Wrangler.

   After you import data into an Amazon Personalize dataset, you can analyze it, export it to an Amazon S3 bucket, update it, or delete it by deleting the dataset. For more information, see Managing data.

3. **Create domain recommenders or custom resources**

   After you import your data, create domain recommenders (for Domain dataset groups) or custom resources (for Custom dataset group) to train a model on your data. You use these resources to generate recommendations.

4. **Get recommendations**

   Use your recommender or custom campaign to get recommendations. With a Custom dataset group, you can also get batch recommendations or user segments.

After you complete the Amazon Personalize workflow the first time, keep data current, and regularly re-train any custom solutions. 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](#).

Step 1: Creating a dataset group

When you start using Amazon Personalize, you create a dataset group. A *dataset group* is a container for Amazon Personalize resources, including datasets, domain recommenders, and custom resources. A dataset group organizes your resources into independent collections, where resources from one dataset group can't influence resources in any other dataset group.

You create a dataset group for each of your business domains. For example, you might have an application that provides recommendations for streaming video and another that provides recommendations for audio books. In Amazon Personalize, you would create a dataset group for each application. This way, the data from one application does not influence the recommendations Amazon Personalize generates for the other application.

You can create a Domain dataset group or a Custom dataset group:

- With a *Domain dataset group*, you create resources that are pre-configured and optimized for different use cases. When you create a dataset group, you make it a Domain dataset group by specifying a domain of VIDEO_ON_DEMAND or ECOMMERCE.

  If you have a streaming video or e-commerce application, we recommend that you create a Domain dataset group. You can still add custom resources, such as solutions and solution versions trained for custom use cases.
- A *Custom dataset group* includes only custom resources that you configure depending on your use case. With custom resources, you train and deploy configurable solutions and solution versions (a trained Amazon Personalize recommendation model) based on your business needs. If don't have a VIDEO_ON_DEMAND or ECOMMERCE application, we recommend that you create a Custom dataset group. Otherwise, we recommend starting with a Domain dataset group and adding custom resources as necessary.

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)

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 your **Domain**:
   - Choose **E-commerce** to create an ECOMMERCE Domain dataset group.
   - Choose **Video on demand** to create a VIDEO_ON_DEMAND Domain dataset group.
   - Choose **Custom** to create a Custom dataset group with only custom resources, such as solutions, campaigns, and batch inference jobs.
6. For **Tags**, optionally add any tags. For more information about tagging Amazon Personalize resources, see Tagging Amazon Personalize resources.
7. Choose **Create dataset group**. The **Overview** page displays. You are now ready to import data. See Step 2: Preparing and importing data.

Creating a dataset group (AWS CLI)

To create a dataset group, use the `create-dataset-group` operation. To create a Domain dataset group, for domain specify `ECOMMERCE` or `VIDEO_ON_DEMAND`. To create a Custom dataset group, don't specify a domain. You can use the Tags parameter to optionally tag resources in Amazon Personalize. For a sample see Adding tags (AWS CLI).

The following code creates a Domain dataset group for the `VIDEO_ON_DEMAND` domain.

```
aws personalize create-dataset-group \
--name dataset-group-name \
--domain VIDEO_ON_DEMAND
```

If successful, the dataset group Amazon Resource Name (ARN) display as follows.

```
{
  "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 display as follows.

```
{
    "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, you are ready to import data. For more information, see Step 2: Preparing and importing data.

Creating a dataset group (AWS SDKs)

Use the following code to create a Domain dataset group. Give the Domain dataset group a name, and for `domain`, specify either `ECOMMERCE` or `VIDEO_ON_DEMAND`. To create a Custom dataset group, modify the code to remove the domain parameter.

For more information about the API operation, see CreateDatasetGroup in the API reference section. You can use the Tags parameter to optionally tag resources in Amazon Personalize. For a sample see Adding tags (AWS SDKs).

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_dataset_group(
  name = 'dataset group name',
  domain = 'business domain'
)
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 String createDomainDatasetGroup(PersonalizeClient personalizeClient,
                                              String datasetGroupName,
                                              String domain) {

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

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import {  CreateDatasetGroupCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the domain dataset group parameters.
export const domainDatasetGroupParams = {
  name: 'NAME',  /* required */
  domain: 'DOMAIN'   /* required for a domain dsg, specify ECOMMERCE or
 VIDEO_ON_DEMAND */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateDatasetGroupCommand(domainDatasetGroupParams));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

The DescribeDatasetGroup operation returns the `datasetGroupArn` and the status of the operation. When the dataset group's `status` is ACTIVE, you are ready to import data. For more information, see Step 2: Preparing and importing data.

Step 2: Preparing and importing data

Amazon Personalize uses your data to generate recommendations for your users and user segments. Amazon Personalize stores your data in datasets until you delete the datasets. For all use cases (Domain dataset groups) and recipes (custom resources), your interactions data must have the following:

- At minimum 1000 item interactions records from users interacting with items in your catalog. These interactions can be from bulk imports, or streamed events, or both.
- At minimum 25 unique user IDs with at least two item interactions for each.

For quality recommendations, we recommend that you have at minimum 50,000 item interactions from at least 1,000 users with two or more item interactions each.

When you import data, you can choose to import records in bulk, individually, or both.

- Bulk imports involve importing a large number of historical records at once. You can prepare and import your item interaction, user, and item bulk data with SageMaker Data Wrangler and multiple data sources. Or you can prepare bulk data yourself, and import it directly into Amazon Personalize from a CSV file in Amazon S3. For information about how to format your bulk data for Amazon Personalize, see Data format guidelines.
- With individual imports, you import individual records with the Amazon Personalize console and API operations. Or you can import interactions data from live events in real time.

After you import data into an Amazon Personalize dataset, you can analyze it, export it to an Amazon S3 bucket, update it, or delete it by deleting the dataset. For more information, see Managing data.

As your catalog grows, update your historical data with additional bulk, or individual data, import operations. For real-time recommendations, keep your Item interactions dataset up to date with your users' behavior. You do this by recording real-time interaction *events* with an event tracker and the PutEvents operation. For more information, see Recording events

Topics

- Preparing and importing bulk data
- Importing individual records

Preparing and importing bulk data

When you have created a dataset, you are ready to start importing your bulk historical data into Amazon Personalize. You have two choices for importing your bulk records:

- For Item interactions, Users, and Items datasets, you can use Amazon SageMaker Data Wrangler to import your data from 40+ sources, generate visualizations and Amazon Personalize specific insights, and transform it to meet Amazon Personalize requirements.
- For all dataset types, you can import bulk data directly into datasets. When you import directly, you manually format your data to meet Amazon Personalize requirements and upload it to Amazon S3. Then you create a schema and a dataset, and import the data directly into the dataset with a dataset import job.

The following guidelines can help you make sure your bulk data is formatted correctly.

- Your input data must be in a CSV (comma-separated values) file.
- The first row of your CSV file must contain your column headers. Don't enclose headers in quotation marks (").
- Make sure you have the required fields for your dataset type and make sure that their names align with Amazon Personalize requirements. For example, your Items data might have a column called `ITEM_IDENTIFICATION_NUMBER` with IDs for each of your items. To use this column as an ITEM_ID field, rename the column to `ITEM_ID`. If you use Data Wrangler to format your data, you can use the **Map columns for Amazon Personalize** Data Wrangler transform to make sure your columns are named correctly.

  For information about required fields, see Schemas. For information about using Data Wrangler to prepare your data, see Preparing and importing data using Amazon SageMaker Data Wrangler.
- The column header names in your CSV file must map to your schema.
- Each record in your CSV file must be on a single line.
- The data types in each column must map to your schema. If you use Data Wrangler to format your data, you can use the Data Wrangler transform Parse Value as Type to convert the data types.
- `TIMESTAMP` and `CREATION_TIMESTAMP` data must be in *UNIX epoch* time format. For more information, see Timestamp data.
- If your data includes any non-ASCII encoded characters, your CSV file must be encoded in UTF-8 format.
- Makes sure you format any textual data as described in Unstructured text metadata.
- Make sure you format impression data and categorical data as described in Formatting explicit impressions and Formatting categorical data.

For more information about bulk data formatting requirements for Amazon Personalize, see Data format guidelines.

After you import data into an Amazon Personalize dataset, you can analyze it, export it to an Amazon S3 bucket, update it, or delete it by deleting the dataset. For more information, see Managing data.

If you already created a recommender or deployed a custom solution version with a campaign, how new bulk records influence recommendations depends on the domain use case or recipe that you use. For more information, see How new data influences real-time recommendations.

Filter updates for bulk records

Within 20 minutes of completing a bulk import, Amazon Personalize updates any filters you created in the dataset group with your new item and user data. This update allows Amazon Personalize to use the most recent data when filtering recommendations for your users.

Topics

- Preparing and importing data using Amazon SageMaker Data Wrangler
- Importing data directly into Amazon Personalize datasets

Preparing and importing data using Amazon SageMaker Data Wrangler

> **Important**
>
> As you use Data Wrangler, you incur SageMaker costs. For a complete list of charges and prices, see the Data Wrangler tab of Amazon SageMaker pricing. To avoid incurring additional fees, when you are finished, shut down your Data Wrangler instance. For more information, see Shut Down Data Wrangler.

After you create a dataset group, you can use Amazon SageMaker Data Wrangler (Data Wrangler) to import data from 40+ sources into an Amazon Personalize dataset. Data Wrangler is a feature of Amazon SageMaker Studio Classic that provides an end-to-end solution to import, prepare, transform, and analyze data. You can't use Data Wrangler to prepare and import data into an Actions dataset or Action interactions dataset.

When you use Data Wrangler to prepare and import data, you use a data flow. A *data flow* defines a series of machine learning data prep steps, starting with importing data. Each time you add a step to your flow, Data Wrangler takes an action on your data, such as transforming it or generating a visualization.

The following are some of the steps that you can add to your flow to prepare data for Amazon Personalize:

- **Insights:** You can add Amazon Personalize specific insight steps to your flow. These insights can help you learn about your data and what actions you can take to improve it.
- **Visualizations:** You can add visualization steps to generate graphs such as histograms and scatter plots. Graphs can help you discover issues in your data, such as outliers or missing values.
- **Transformations:** You can use Amazon Personalize specific and general transformation steps to make sure your data meets Amazon Personalize requirements. The Amazon Personalize transformation helps you map your data columns to required columns depending on the Amazon Personalize dataset type.

If you need to leave Data Wrangler before importing data into Amazon Personalize, you can return to where you left off by choosing the same dataset type when you launch Data Wrangler from the Amazon Personalize console. Or you can access Data Wrangler directly through SageMaker Studio Classic.

We recommend you import data from Data Wrangler into Amazon Personalize as follows. The transformation, visualization and analysis steps are optional, repeatable, and can be completed in any order.

1. **Set up permissions** - Set up permissions for Amazon Personalize and SageMaker service roles. And set up permissions for your users.
2. **Launch Data Wrangler in SageMaker Studio Classic from the Amazon Personalize console** - Use the Amazon Personalize console to configure a SageMaker domain and launch Data Wrangler in SageMaker Studio Classic.
3. **Import your data into Data Wrangler** - Import data from 40+ sources into Data Wrangler. Sources include AWS services, such as Amazon Redshift, Amazon EMR, or Amazon Athena, and 3rd parties such as Snowflake or DataBricks.
4. **Transform your data** - Use Data Wrangler to transform your data to meet Amazon Personalize requirements.
5. **Visualize and analyze your data** - Use Data Wrangler to visualize your data and analyze it through Amazon Personalize specific insights.

6. **Process and import data into Amazon Personalize** - Use a SageMaker Studio Classic Jupyter notebook to import your processed data into Amazon Personalize.

Additional information

The following resources provide additional information about using Amazon SageMaker Data Wrangler and Amazon Personalize.

- For a tutorial that walks you through processing and transforming a sample dataset, see Demo: Data Wrangler Titanic Dataset Walkthrough in the *Amazon SageMaker Developer Guide*. This tutorial introduces the fields and functions of Data Wrangler.
- For information on onboarding to Amazon SageMaker domains, see Quick onboard to Amazon SageMaker Domain in the *Amazon SageMaker Developer Guide*.
- For information on Amazon Personalize data requirements, see Data format guidelines and Schemas.

Setting up permissions

To prepare data with Data Wrangler, you must set up the following permissions:

- **Create a service role for Amazon Personalize:** If you haven't already, complete the instructions in Setting up Amazon Personalize to create an IAM service role for Amazon Personalize. This role must have `GetObject` and `ListBucket` permissions for the Amazon S3 buckets that store your processed data. And it must have permission to use any AWS KMS keys.

  For information about granting Amazon Personalize access to your Amazon S3 buckets, see Giving Amazon Personalize access to Amazon S3 resources. For information about granting Amazon Personalize access to your AWS KMS keys, see Giving Amazon Personalize permission to use your AWS KMS key.
- **Create an administrative user with SageMaker permissions:** Your administrator must have full access to SageMaker and must be able to create a SageMaker domain. For more information, see Create an Administrative User and Group in the *Amazon SageMaker Developer Guide*.
- **Create a SageMaker execution role:** Create a SageMaker execution role with access to SageMaker resources and Amazon Personalize data import operations. The SageMaker execution role must have the `AmazonSageMakerFullAccess` policy attached. If you require more granular Data Wrangler permissions, see Data Wrangler Security and Permissions in the *Amazon SageMaker Developer Guide*. For more information on SageMaker roles, see SageMaker Roles.

To grant access to Amazon Personalize data import operations, attach the following IAM policy to the SageMaker execution role. This policy grants the permissions required to import data into Amazon Personalize and attach a policy to your Amazon S3 bucket. And it grants `PassRole` permissions when the service is Amazon Personalize. Update the Amazon S3 `bucket-name` to the name of the Amazon S3 bucket you want to use as the destination for your formatted data after you prepare it with Data Wrangler.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "personalize:Create*",
                "personalize:List*",
                "personalize:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutBucketPolicy"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "personalize.amazonaws.com"
                }
            }
        }
    ]
```

```
}
```

For information on creating an IAM policy, see Creating IAM policies in the *IAM User Guide*. For information on attaching an IAM policy to role, see Adding and removing IAM identity permissions in the *IAM User Guide*.

Launching Data Wrangler from Amazon Personalize

To launch Data Wrangler from Amazon Personalize, you use the Amazon Personalize console to configure a SageMaker domain and launch Data Wrangler.

To launch Data Wrangler from Amazon Personalize

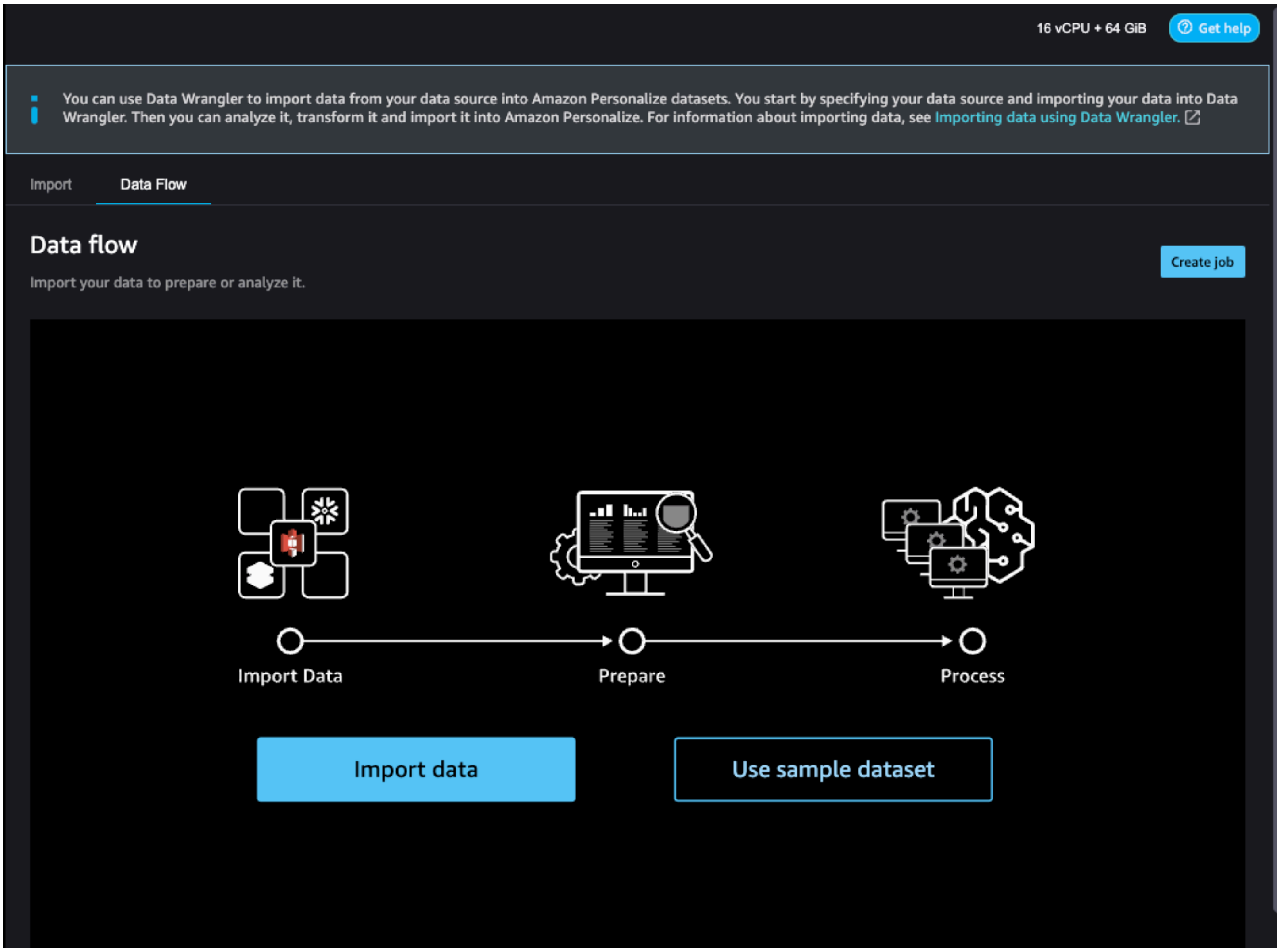
1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/home and sign in to your account.
2. On the **Dataset groups** page, choose your dataset group.
3. In **Set up datasets** choose **Create dataset** and choose the type of dataset to create. You can't use Data Wrangler to prepare an Actions dataset or Action interactions dataset.
4. Choose **Import data using Data Wrangler** and choose **Next**.
5. For **SageMaker domain**, choose to use an existing domain or create a new one. You need a SageMaker Domain to access Data Wrangler in SageMaker Studio Classic. For information about domains and user profiles, see SageMaker Domain in the *Amazon SageMaker Developer Guide*.
6. To use an existing domain, choose a **SageMaker domain** and **User profile** to configure the domain.
7. To create a new domain:
   - Give the new domain a name.
   - Choose a **User profile name**.
   - For **Execution role**, choose the role you created in Setting up permissions. Or, if you have CreateRole permissions, create a new role using the role creation wizard. The role you use must have the `AmazonSageMakerFullAccess` policy attached.
8. Choose **Next**. If you are creating a new domain, SageMaker starts creating your domain. This can take up to ten minutes.
9. Review the details for your SageMaker domain.

10. Choose **Import data with Data Wrangler**. SageMaker Studio Classic starts creating your environment, and when complete, the **Data flow** page of Data Wrangler in SageMaker Studio Classic opens in a new tab. It can take up to five minutes for SageMaker Studio Classic to finish creating your environment. When it finishes, you are ready to start importing data into Data Wrangler. For more information, see Importing data into Data Wrangler.

Importing data into Data Wrangler

After you configure a SageMaker domain and launch Data Wrangler in a new tab, you are ready to import data from your source into Data Wrangler. When you use Data Wrangler to prepare data for Amazon Personalize, you import one dataset at a time. We recommend starting with an Item interactions dataset. You can't use Data Wrangler to prepare an Actions dataset or Action interactions dataset.

You start on the **Data flow** page. The page should look similar to the following.



To start importing data, you choose **Import data** and specify your data source. Data Wrangler supports 40+ sources. These include AWS services, such as Amazon Redshift, Amazon EMR, or Amazon Athena, and third parties, such as Snowflake or DataBricks. Different data sources have different procedures for connecting and importing data.

For a complete list of available sources and step-by-step instructions on importing data, see Import in the *Amazon SageMaker Developer Guide*.

After you import data into Data Wrangler, you are ready to transform it. For information about transforming data, see Transforming data.

Transforming data

To transform data in Data Wrangler, you add a **Transform** step to your data flow. Data Wrangler includes over 300 transforms that you can use to prepare your data, including a **Map columns**

for Amazon Personalize transform. And you can use the general Data Wrangler transforms to fix issues such as outliers, type issues, and missing values.

After you finish transforming your data, you can analyze it with Data Wrangler. Or, if you are finished preparing your data in Data Wrangler, you can process it and import it into Amazon Personalize. For information about analyzing data, see Generating visualizations and data insights. For information about processing and importing data, see Processing data and importing it into Amazon Personalize.

Topics

- Mapping columns for Amazon Personalize
- General Data Wrangler transforms

Mapping columns for Amazon Personalize

To transform your data so it meets Amazon Personalize requirements, you add the **Map columns for Amazon Personalize** transform and map your columns to the required and optional fields for Amazon Personalize.

To use the Map columns for Amazon Personalize transform

1. Choose **+** for your latest transform and choose **Add transform**. If you haven't added a transform, choose the **+** for the **Data types** transform. Data Wrangler adds this transform automatically to your flow.
2. Choose **Add step**.
3. Choose **Transforms for Amazon Personalize**. The **Map columns for Amazon Personalize** transform is selected by default.
4. Use the transform fields to map your data to required Amazon Personalize attributes.
    1. Choose the dataset type that matches your data (Interactions, Items, or Users).
    2. Choose your domain (ECOMMERCE, VIDEO_ON_DEMAND, or custom). The domain you choose must match the domain you specified when you created your dataset group.
    3. Choose the columns that match the required and optional fields for Amazon Personalize. For example, for the item_ID column, choose the column in your data that stores the unique identification information for each of your items.

Each column field is filtered by data type. Only the columns in your data that meet Amazon Personalize data type requirements are available. If your data is not of the required type, you can use the Parse Value as Type Data Wrangler transform to convert it.

General Data Wrangler transforms

The following general Data Wrangler transforms can help you prepare data for Amazon Personalize:

- Data type conversion: If your field is not listed as a possible option in the **Map columns for Amazon Personalize** transform, you might need to convert its data type. The Data Wrangler transform Parse Value as Type can help you convert your data. Or you can use the **Data types** transform that Data Wrangler adds by default when you create a flow. To use this transform, you choose the data type from the **Type** drop-down lists, choose **Preview** and then choose **Update**.

  For information on required data types for fields, see the section for your domain and dataset type in Schemas.
- Handling missing values and outliers: If you generate missing value or outlier insights, you can use the Data Wrangler transforms Handle Outliers and Handle Missing Values to resolve these issues.
- Custom transformations: With Data Wrangler, you can create your own transformations with Python (User-Defined Function), PySpark, pandas, or PySpark (SQL). You might use a custom transform to perform tasks such as dropping duplicate columns or grouping by columns. For more information, see Custom Transforms in the *Amazon SageMaker Developer Guide*.

Generating visualizations and data insights

After you import your data into Data Wrangler, you can use it to generate visualizations and data insights.

- **Visualizations**: Data Wrangler can generate different types of graphs, such as histograms and scatter plots. For example, you can generate a histogram to identify outliers in your data.
- **Data insights**: You can use a *Data Quality and Insights Report for Amazon Personalize* to learn about your data through data insights and column and row statistics. This report can let you know if you have any type issues in your data. And you can learn what actions you can take to

improve your data. These actions can help you meet Amazon Personalize resource requirements, such as model training requirements, or they can lead to improved recommendations.

After you learn about your data through visualizations and insights, you can use this information to help you apply additional transforms to improve your data. Or, if you are finished preparing your data, you can process it and import it into Amazon Personalize. For information about transforming your data, see Transforming data. For information about processing and importing data, see Processing data and importing it into Amazon Personalize.

Generating visualizations

You can use Data Wrangler to create different types of graphs, such as histograms and scatter plots. For example, you can generate a histogram to identify outliers in your data. To generate a data visualization, you add an **Analysis** step to your flow and, from **Analysis type**, choose the visualization you want to create.

For more information about creating visualizations in Data Wrangler, see Analyze and Visualize in the *Amazon SageMaker Developer Guide*.

Generating data insights

You can use Data Wrangler to generate a **Data Quality and Insights Report for Amazon Personalize** report specific to your dataset type. Before generating the report, we recommend that you transform your data to meet Amazon Personalize requirements. This will lead to more relevant insights. For more information, see Transforming data.

Topics

- Report content
- Generating the report

Report content

The **Data Quality and Insights Report for Amazon Personalize** includes the following sections:

- **Summary:** The report summary includes dataset statistics and high priority warnings:
  - **Dataset statistics:** These include Amazon Personalize specific statistics, such as the number of unique users in your interactions data, and general statistics, such as the number of missing values or outliers.

  - **High priority warnings:** These are Amazon Personalize specific insights that have the most impact on training or recommendations. Each warning includes a recommended action that you can take to resolve the issue.
- **Duplicate rows and Incomplete rows:** These sections include information on which rows have missing values and which rows are duplicated in your data.
- **Feature summary:** This section includes the data type for each column, invalid or missing data information, and warning counts.
- **Feature details:** This section includes subsections with detailed information for each of your columns of data. Each subsection includes statistics for the column, such as categorical value count, and missing value information. And each subsection includes Amazon Personalize specific insights and recommended actions for columns of data. For example, an insight might indicate that a column has more than 30 possible categories.

Data type issues

The report identifies columns that are not of the correct data type and specifies the required type. To get insights related to these features, you must convert the data type of the column and generate the report again. To convert the type, you can use the Data Wrangler transform Parse Value as Type.

Amazon Personalize insights

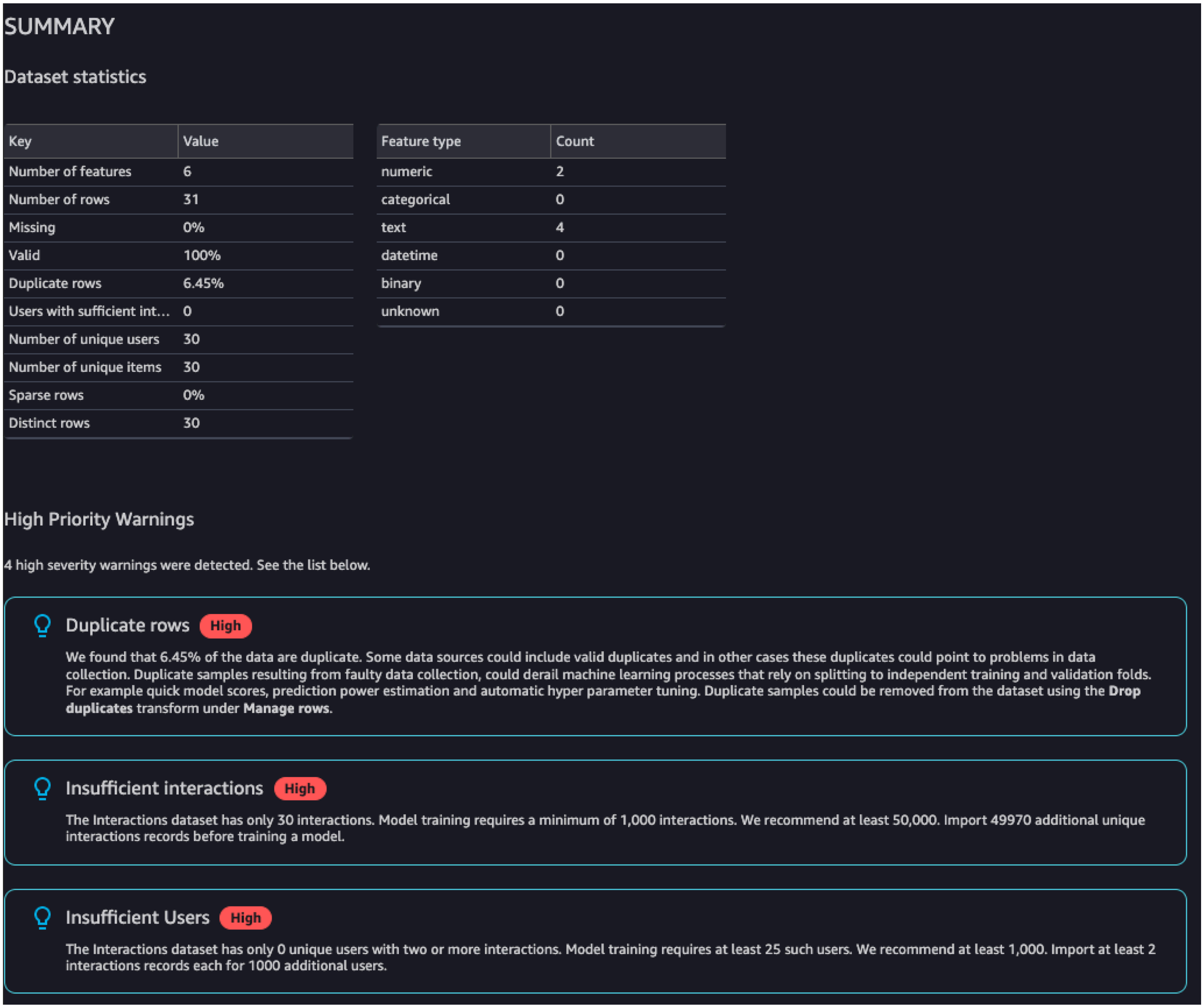
The Amazon Personalize insights include a finding and a suggested action. The action is optional. For example, the report might include an insight and action related to the number of categories for a column of categorical data. If you don't believe the column is a categorical, you can disregard this insight and take no action.

Except for minor wording differences, the Amazon Personalize specific insights are the same as the *single dataset* insights you might generate when you analyze your data with Amazon Personalize. For example, the insights report in Data Wrangler includes insights such as "The Item interactions dataset has only X unique users with two or more interactions." But it doesn't include insights like "X% of items in the *Items dataset* have no interactions in the *Item interactions dataset*."

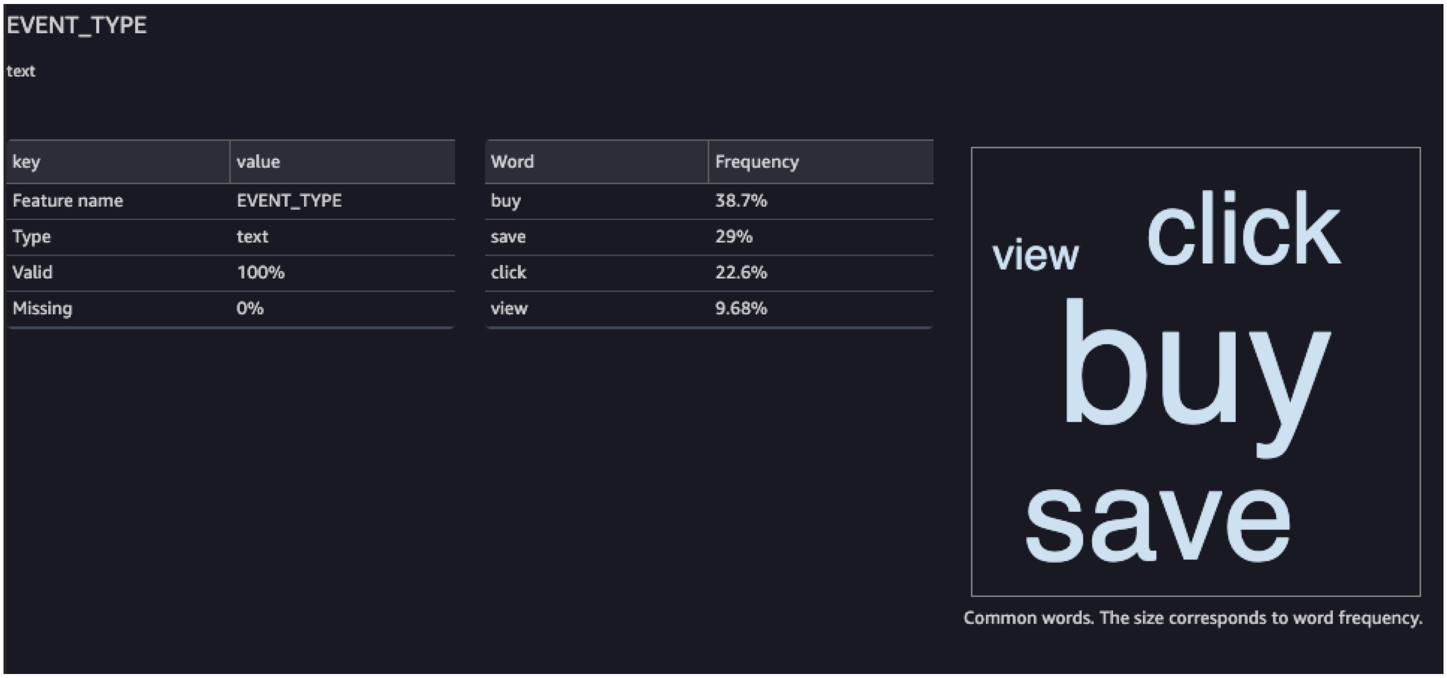
For a list of possible Amazon Personalize specific insights, see the insights that don't reference multiple datasets in Data insights.

Report examples

The look and feel of the Amazon Personalize report is the same as the general insights report in Data Wrangler. For examples of the general insights report, see [Get Insights On Data and Data Quality](#) in the *Amazon SageMaker Developer Guide*. The following example shows how the summary section of a report for an Item interactions dataset. It includes dataset statistics and some possible high priority Item interactions dataset warnings.



The following example shows how the feature details section for an EVENT_TYPE column of an Item interactions dataset might appear in a report.



Generating the report

To generate the **Data Quality and Insights Report for Amazon Personalize**, you choose **Get data insights** for your transform and create an analysis.

To generate Data Quality and Insights Report for Amazon Personalize

1. Choose the **+** option for the transform you are analyzing. If you haven't added a transform, choose the **+** for the **Data types** transform. Data Wrangler adds this transform automatically to your flow.
2. Choose **Get data insights**. The **Create analysis** panel displays.
3. For **Analysis type**, choose **Data Quality and Insights Report for Amazon Personalize**.
4. For **Dataset type**, choose the type of Amazon Personalize dataset you are analyzing.
5. Optionally choose **Run on full data**. By default, Data Wrangler generates insights on only a sample of your data.
6. Choose **Create**. When analysis completes, the report appears.

Processing data and importing it into Amazon Personalize

When you are finished analyzing and transforming your data, you are ready to process it and import it into Amazon Personalize.

- **Processing data** – Processing the data applies your transform to your entire dataset and outputs it to a destination you specify. In this case you specify an Amazon S3 bucket.
- **Importing data into Amazon Personalize** – To import processed data into Amazon Personalize, you run a Jupyter Notebook provided in SageMaker Studio Classic. This notebook creates your Amazon Personalize datasets and imports your data into them.

Processing data

Before you import data into Amazon Personalize, you must apply your transform to your entire dataset and output it to an Amazon S3 bucket. To do this, you create a destination node with the destination set to an Amazon S3 bucket, and then launch a processing job for the transformation.

For step-by-step instructions on specifying a destination and launching a process job, see Launch processing jobs with a few clicks using Amazon SageMaker Data Wrangler. When you add a destination, choose **Amazon S3**. You will use this location when you import the processed data into Amazon Personalize.

When you finish processing your data, you are ready to import it from the Amazon S3 bucket into Amazon Personalize.

Importing data into Amazon Personalize

After you process your data, you are ready to import it into Amazon Personalize. To import processed data into Amazon Personalize, you run a Jupyter Notebook provided in SageMaker Studio Classic. This notebook creates your Amazon Personalize datasets and imports your data into them.

To import processed data into Amazon Personalize

1. For the transformation you want to export, choose **Export to** and choose **Amazon Personalize (via Jupyter Notebook)**.
2. Modify the notebook to specify the Amazon S3 bucket you used as the data destination for the processing job. Optionally specify the domain for your dataset group. By default, the notebook creates a custom dataset group.
3. Review the notebook cells that create the schema. Verify that the schema fields have the expected types and attributes before running the cell.

    - Verify that fields that support null data have `null` listed in the list of types. The following example shows how to add `null` for a field.

```
{
  "name": "GENDER",
  "type": [
    "null",
    "string"
  ],
  "categorical": true
}
```

- Verify that categorical fields have the categorical attribute set to true. The following example shows how to mark a field categorical.

```
{
          "name": "SUBSCRIPTION_MODEL",
          "type": "string",
          "categorical": true
}
```

- Verify that textual fields have the textual attribute set to true. The following example shows how to mark a field as textual.

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

4. Run the notebook to create a schema, and create dataset, and import your data into the Amazon Personalize dataset. You run the notebook just as you would a notebook outside of SageMaker Studio Classic. For information on running Jupyter notebooks, see Running Code. For information on notebooks in SageMaker Studio Classic, see Use Amazon SageMaker Notebooks in the *Amazon SageMaker Developer Guide*.

   After you complete the notebook, if you imported interactions data, you are ready to create recommenders or custom resources. Or you can repeat the process with an items dataset or users dataset. For information about creating recommenders or custom resources, see Step 3: Creating recommenders or custom resources.

Importing data directly into Amazon Personalize datasets

After you have created a dataset, you are ready to import bulk records from a large CSV file into an Amazon Personalize dataset.

To import data directly into Amazon Personalize datasets, you do the following:

1. Create schema JSON file based on your data. For schema requirements and examples, see Schemas.
2. Make sure your data is correctly formatted. The column names must match your schema. Your data must be in a CSV file. For data format guidelines, see Data format guidelines.
3. Upload your CSV files to an Amazon Simple Storage Service (Amazon S3) bucket and give Amazon Personalize access to your Amazon S3 resources.
4. Create an Amazon Personalize schema using the JSON file from step one. And create an Amazon Personalize dataset.
5. Create a dataset import job that populates the dataset with data from your Amazon S3 bucket. To create a dataset import job for interactions datasets, your CSV file must have at minimum 1000 interaction records.

After you import data into an Amazon Personalize dataset, you can analyze it, export it to an Amazon S3 bucket, update it, or delete it by deleting the dataset. For more information, see Managing data.

Topics

Uploading to an Amazon S3 bucket

After you format your historical input data (see Data format guidelines), 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 to set up permissions so Amazon Personalize can access your Amazon Personalize resources on your behalf.

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.

   If you use AWS Key Management Service (AWS KMS) for encryption, you must grant Amazon Personalize and your Amazon Personalize IAM service role permission to use your key. For more information, see Giving Amazon Personalize permission to use your AWS KMS key.

After you upload your data to an Amazon S3 bucket and give Amazon Personalize access to Amazon S3, you are ready to create an Amazon Personalize schema and dataset. See Creating a dataset and a schema.

Creating a dataset and a schema

After you have completed Step 1: Creating a dataset group, 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.

Topics

- Creating a dataset and a schema (console)
- Creating a dataset and a schema (AWS CLI)
- Creating a dataset and a schema (AWS SDKs)

Creating a dataset and a schema (console)

If this is your first dataset in your dataset group, your first dataset type will be an Item interactions dataset. To create your Item 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.

> **Note**
>
> If you just completed Step 1: Creating a dataset group and you are already creating your dataset, 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.
3. In **Set up datasets** choose **Create dataset** and choose the type of dataset to create.
4. Choose **Import data directly into Amazon Personalize datasets** and choose **Next**.
5. In **Dataset details**, for **Dataset name**, specify a name for your dataset.
6. In **Schema details**, for **Schema selection**, either choose an existing schema or choose **Create new schema**.
7. If you are creating a new schema, for **Schema definition**, paste in the schema JSON that matches your data. Use the examples found in Schemas as a guide. After you create a schema, you can't make changes to the schema.
8. For **New schema name**, specify a name for the new schema.
9. For **Tags**, optionally add any tags. For more information about tagging Amazon Personalize resources, see Tagging Amazon Personalize resources.
10. Choose **Next** and follow the instructions in Preparing and importing bulk data 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 operation. Then create a dataset using the CreateDataset operation. For information on Amazon Personalize datasets and schema requirements, see Datasets and schemas.

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 types of datasets recognized by Amazon Personalize. The following is an example of a minimal *Item interactions dataset* schema. For more examples, see Schemas.

```
{
  "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. After you create a schema, you can't make changes to the schema. 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.

   If you are creating a schema for a dataset in a Domain dataset group, add the `domain` parameter and set it to `ECOMMERCE` or `VIDEO_ON_DEMAND`. For more information about the API, see CreateSchema.

```
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) and schema ARN from the previous step. Dataset type values can be `Interactions`, `Users`, `Items`, `Actions`, or `Action_Interactions`. For more information about the API, see CreateDataset.

```
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 create a dataset, you are ready to import your training data. See Preparing and importing bulk data.

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 operation. After you create a schema, you can't make changes to the schema. Then create a dataset using the CreateDataset operation. For information on Amazon Personalize datasets and schema requirements, see Datasets and schemas.

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 types of datasets recognized by Amazon Personalize. The following is an example of a minimal *Item interactions dataset* schema. For more examples, see Schemas.

```
{
  "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 with the following code. Specify the name for your schema and the file path for your schema JSON file.

   If you are creating a schema for a dataset in a Domain dataset group, add the `domain` parameter and set it to `ECOMMERCE` or `VIDEO_ON_DEMAND`. For more information about the API, see CreateSchema.

   SDK for Python (Boto3)

```
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

```
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 "";
}
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { CreateSchemaCommand } from
```

```
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

import fs from 'fs';

let schemaFilePath = "SCHEMA_PATH";
let mySchema = "";

try {
  mySchema = fs.readFileSync(schemaFilePath).toString();
} catch (err) {
  mySchema = 'TEST' // For unit tests.
}
// Set the schema parameters.
export const createSchemaParam = {
  name: 'NAME', /* required */
  schema: mySchema /* required */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateSchemaCommand(createSchemaParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

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 operation. The following code shows how to create a dataset. Specify the Amazon Resource Name (ARN) of your dataset group, the schema ARN from the previous step, and specify the dataset type. Dataset type values can be `Interactions`, `Users`, `Items`, `Actions`, or `Action_Interactions`. For information about the different types of datasets, see Datasets and schemas.

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

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

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

SDK for Java 2.x

```
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 "";
```

```
}
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { CreateDatasetCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the dataset's parameters.
export const createDatasetParam = {
  datasetGroupArn: 'DATASET_GROUP_ARN', /* required */
  datasetType: 'DATASET_TYPE', /* required */
  name: 'NAME', /* required */
  schemaArn: 'SCHEMA_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateDatasetCommand(createDatasetParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

After you create a dataset, you are ready to import your training data. See Preparing and importing bulk data.

Importing bulk records with a dataset import job

After you have formatted your input data (see Data format guidelines), uploaded it to an Amazon Simple Storage Service (Amazon S3) bucket (see Uploading to an Amazon S3 bucket), and completed Creating a dataset and a schema, import the bulk records into the dataset by creating a dataset import job.

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

If you've previously created a dataset import job for a dataset, you can use a new dataset import job to add to or replace the existing bulk data. For more information see Updating existing bulk data.

Topics

- Importing bulk records (console)
- Importing bulk records (AWS CLI)
- Importing bulk records (AWS SDKs)

Importing bulk records (console)

> **Important**
>
> By default, a dataset import job replaces any existing data in the dataset that you imported in bulk. For information about updating existing data, see Updating existing bulk data.

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

If you just created your dataset in Creating a dataset and a schema, skip to step 5. If you've already completed an initial import job and want to refresh your data, see Updating bulk records (console).

To import bulk records (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. The dataset group **Overview** displays.
3. In the navigation pane, choose **Datasets** and choose the dataset you want to import bulk data into.

4. In **Dataset import jobs**, choose **Create dataset import job**.
5. In **Dataset import job details**, for **Data import source** choose **Import data from S3**.
6. For **Dataset import job name**, specify a name for your import job.
7. In **Input source**, for **S3 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>
   ```

   If your CSV files are in a folder in your Amazon S3 bucket and you want to upload multiple CSV files to a dataset with one dataset import job, you can specify the path to the folder. Amazon Personalize only uses the files in the first level of your folder, it doesn't use any data in any sub folders. Use the following syntax with a / after the folder name:

   ```
   s3://<name of your S3 bucket>/<folder path>/
   ```
8. In **IAM role**, choose to either create a new role or use an existing one. If you completed the perquisites, choose **Use an existing service role** and specify the role that you created in Creating an IAM role for Amazon Personalize.
9. If you created a metric attribution and want to publish metrics related to this job to Amazon S3, in **Publish event metrics to S3** choose **Publish metrics for this import job**.

   If you haven't created one and want to publish metrics for this job, choose **Create metric attribution** to create a new one on a different tab. After you create the metric attribution, you can return to this screen and finish creating the import job.

   For more information on metric attributions, see Measuring impact of recommendations.
10. For **Tags**, optionally add any tags. For more information about tagging Amazon Personalize resources, see Tagging Amazon Personalize resources.
11. 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. After you import data into an Amazon Personalize dataset, you can analyze it, export it to an Amazon S3 bucket, update it, or delete it by deleting the dataset. For more information, see Managing data.

    After you import your data you are ready to create a solution. For more information, see Creating a solution and a solution version.

Importing bulk records (AWS CLI)

> **Important**
>
> By default, a dataset import job replaces any existing data in the dataset that you imported in bulk. For information about updating existing data, see Updating existing bulk data.

To import bulk records using the AWS CLI, create a dataset import job using the CreateDatasetImportJob command. If you've previously created a dataset import job for a dataset, you can use the import mode parameter to specify how to add the new data. For a code sample, see Updating bulk records (AWS CLI).

Import bulk records (AWS CLI)

1. Create a dataset import job by running the following command. Provide the Amazon Resource Name (ARN) for your dataset and specify the path to your Amazon S3 bucket where you stored the training data. Use the following syntax for the path:

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

   If your CSV files are in a folder in your Amazon S3 bucket and you want to upload multiple CSV files to a dataset with one dataset import job, you can specify the path to the folder. Amazon Personalize only uses the files in the first level of your folder, it doesn't use any data in any sub folders. Use the following syntax with a / after the folder name:

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

   Provide the AWS Identity and Access Management (IAM) role Amazon Resource Name (ARN) that you created in Creating an IAM role for Amazon Personalize. The default `import-mode` is FULL. For more information see Updating existing bulk data. For more information about the operation, see CreateDatasetImportJob.

   ```
   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 \
   --import-mode FULL
   ```

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.

```
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"
      },
      "importMode": "FULL",
      "roleArn": "role-arn",
      "status": "CREATE PENDING",
      "creationDateTime": 1542392161.837,
      "lastUpdatedDateTime": 1542393013.377
  }
}
```

The dataset import is complete when the status shows as ACTIVE. After you import data into an Amazon Personalize dataset, you can analyze it, export it to an Amazon S3 bucket, update it, or delete it by deleting the dataset. For more information, see Managing data.

After you import your data into the relevant datasets in the dataset group, you can create a solution version (trained model). For more information, see Creating a solution and a solution version.

Importing bulk records (AWS SDKs)

Important

By default, a dataset import job replaces any existing data in the dataset that you imported in bulk. For information about updating existing data, see Updating existing bulk data.

To import data, create a dataset import job with the CreateDatasetImportJob operation. The following code shows how to create a dataset import job.

Give the job name, set the `datasetArn` the Amazon Resource Name (ARN) of your dataset, and set the `dataLocation` to the path to your Amazon S3 bucket where you stored the training data. Use the following syntax for the path:

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

If your CSV files are in a folder in your Amazon S3 bucket and you want to upload multiple CSV files to a dataset with one dataset import job, you can specify the path to the folder. Amazon Personalize only uses the files in the first level of your folder, it doesn't use any data in any sub folders. Use the following syntax with a / after the folder name:

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

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 role for Amazon Personalize. The default `importMode` is `FULL`. For more information see Updating bulk records (AWS SDKs).

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_dataset_import_job(
```

```
    jobName = 'YourImportJob',
    datasetArn = 'dataset_arn',
    dataSource = {'dataLocation':'s3://bucket/file.csv'},
    roleArn = 'role_arn',
    importMode = 'FULL'
)

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

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

  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)
```

```
                .importMode(importMode)
                .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 "";
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
```

```
import { CreateDatasetImportJobCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// Set the dataset import job parameters.
export const datasetImportJobParam = {
  datasetArn: 'DATASET_ARN', /* required */
  dataSource: {
    dataLocation: 's3://<name of your S3 bucket>/<folderName>/<CSVfilename>.csv'  /*
 required */
  },
  jobName: 'NAME',           /* required */
  roleArn: 'ROLE_ARN',       /* required */
  importMode: "FULL"         /* optional, default is FULL */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateDatasetImportJobCommand(datasetImportJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

The response from the DescribeDatasetImportJob 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.

The dataset import is complete when the status shows as ACTIVE. After you import data into an Amazon Personalize dataset, you can analyze it, export it to an Amazon S3 bucket, update it, or delete it by deleting the dataset. For more information, see Managing data.

After you import your data into the relevant datasets in the dataset group, you can create a solution version (trained model). For more information, see Creating a solution and a solution version.

Importing individual records

After you have complete Creating a dataset and a schema, you can import individual records, including item interactions, users, items, actions, or action interactions into an existing dataset. Importing data individually allows you to add small batches of records to your Amazon Personalize datasets as your catalog grows. You can import up to 10 records per individual import operation.

If you use Apache Kafka, you can use the *Kafka connector for Amazon Personalize* to stream data in real time to Amazon Personalize. For information see Kafka Connector for Amazon Personalize in the *personalize-kafka-connector* Github repository.

If you have a large amount of historical records, we recommend that you first import data in bulk and then import data individually as necessary. See Importing data directly into Amazon Personalize datasets.

Filter updates for individual 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 individual import. This update allows your campaigns to use your most recent data when filtering recommendations for your users.

If you already created a recommender or deployed a custom solution version with a campaign, how new individual records influence recommendations depends on the domain use case or recipe that you use. For more information, see How new data influences real-time recommendations.

Topics

- Importing interactions individually
- Importing users individually
- Importing items individually
- Importing actions individually

Importing interactions individually

After you complete Creating a dataset and a schema to create an Item interactions dataset, you can individually import one or more new events into the dataset. To import interaction *events* individually, you create an *event tracker* and then import one or more events into your Item interactions dataset. You can import historical individual 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 with the Amazon Personalize console. We recommend using the Amazon Personalize console to import *only* historical events. For information about using the AWS CLI or the AWS SDKs to record events in real-time, see Recording events.

For information about how Amazon Personalize updates filters for new records and how new records influence recommendations, see Importing individual records.

Topics

Creating an event tracker (console)

> **Note**
>
> If you've created an event tracker, you can skip to Importing events individually (console).

Before you can 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 Item 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 import events with the console (see Importing events individually (console)) or record events in real time using the `PutEvents` operation (see Recording events).

Importing events individually (console)

After you create an event tracker, you can import events individually into an Item interactions dataset. This procedure assumes you have already created an Item interactions dataset. For information about creating datasets, see Creating a dataset and a schema.

To import events individually (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 Item 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 Item 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 individually

After you complete Creating a dataset and a schema to create a Users dataset, you can individually import one or more new users into the dataset. Individually importing users allows you to keep your Users dataset current with small batch imports as your catalog grows. You can import up to 10 users at a time. If you have a large amount of new users, we recommend that you first import data in bulk and then import user data individually as necessary. See Importing data directly into Amazon Personalize datasets.

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 individual records.

Topics

- Importing users individually (console)
- Importing users individually (AWS CLI)
- Importing users individually (AWS SDKs)

Importing users individually (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 Creating a dataset and a schema.

To import users individually (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 individually (AWS CLI)

Add one or more users to your Users dataset with the PutUsers 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 Creating a dataset and a schema.

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 individually (AWS SDKs)

Add one or more users to your Users dataset with the PutUsers operation. If a user with the same `userId` is already in your Users dataset, Amazon Personalize replaces it with the new one. 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 Creating a dataset and a schema.

The following code shows how to add one or more users to your Users dataset. 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".

SDK for Python (Boto3)

```
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

```
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;
}
```

SDK for JavaScript v3

```
import {
  PutUsersCommand,
  PersonalizeEventsClient,
} from "@aws-sdk/client-personalize-events";

const personalizeEventsClient = new PersonalizeEventsClient({
  region: "REGION",
});

// set the put users parameters
var putUsersParam = {
  datasetArn:
    "DATASET ARN",
  users: [
    {
      userId: "userId",
      properties: '{"column1Name": "value", "column2Name": "value"}',
```

```
    },
    {
      userId: "userId",
      properties: '{"column1Name": "value", "column2Name": "value"}',
    },
  ],
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutUsersCommand(putUsersParam)
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Importing items individually

After you complete Creating a dataset and a schema to create an Items dataset, you can individually import one or more new items into the dataset. Individually importing items allows you to keep your Items dataset current with small batch imports as your catalog grows. You can import up to 10 items at a time. If you have a large amount of new items, we recommend that you first import data in bulk and then import item data individually as necessary. See Importing data directly into Amazon Personalize datasets.

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.

For information about how Amazon Personalize updates filters for new records and how new records influence recommendations, see Importing individual records.

Topics

- Importing items individually (console)
- Importing items individually (AWS CLI)
- Importing items individually (AWS SDKs)

Importing items individually (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 Creating a dataset and a schema.

To import items individually (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 individually (AWS CLI)

Add one or more items to your Items dataset using the PutItems 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 Creating a dataset and a schema.

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 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 individually (AWS SDKs)

Add one or more items to your Items dataset using the PutItems operation. You can import up to 10 items with a single PutItems call. If an item with the same itemId is already in your Items dataset, Amazon Personalize replaces it with the new one. This section assumes that you have already created an Items dataset. For information about creating datasets, see Creating a dataset and a schema.

The following code shows how to add one or more items to your Items dataset. 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 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".

SDK for Python (Boto3)

```
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

```
public static int putItems(PersonalizeEventsClient personalizeEventsClient,
                           String datasetArn,
                           String item1Id,
                           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;
    }
```

SDK for JavaScript v3

```
import {
  PutItemsCommand,
  PersonalizeEventsClient,
} from "@aws-sdk/client-personalize-events";

const personalizeEventsClient = new PersonalizeEventsClient({
  region: "REGION",
});

// set the put items parameters
var putItemsParam = {
  datasetArn:
    "DATASET ARN",
  items: [
    {
      itemId: "itemId",
      properties: '{"column1Name": "value", "column2Name": "value"}',
    },
    {
      itemId: "itemId",
      properties: '{"column1Name": "value", "column2Name": "value"}',
    },
  ],
};
export const run = async () => {
  try {
    const response = await personalizeEventsClient.send(
      new PutItemsCommand(putItemsParam)
```

```
    );
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Importing actions individually

After you complete Creating a dataset and a schema to create an Actions dataset, you can individually import one or more new actions into the dataset. When you individually import actions, you keep your Actions dataset current with small batch imports as your catalog grows. You can import up to 10 actions at a time. If you have a large number of new actions, we recommend that you first import data in bulk and then import action data individually as necessary. See Importing data directly into Amazon Personalize datasets.

You can use the Amazon Personalize console, the AWS Command Line Interface (AWS CLI), or AWS SDKs to import actions. If you import an action with the same `actionId` as an action that's already in your Actions dataset, Amazon Personalize replaces it with the new action.

For information about how new records influence recommendations, see How new data influences real-time recommendations.

Topics

- Importing actions individually (console)
- Importing actions individually (AWS CLI)
- Importing actions individually (AWS SDKs)

Importing actions individually (console)

You can import up to 10 actions into an Actions dataset at a time. This section assumes that you have already created an Actions dataset. For information about creating datasets, see Creating a dataset and a schema.

To import actions individually (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 Actions dataset that you want to add to.
3. In the navigation pane, choose **Datasets**.
4. On the **Datasets** page, choose the Actions dataset.
5. At the top right of the dataset details page, choose **Modify dataset**, and then choose **Create record**.
6. In **Create action record(s)** page, for **Record input**, enter the action details in JSON format. The action's field names and values must match the schema you used when you created the Actions 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 actions individually (AWS CLI)

Add one or more actions to your Actions dataset using the `PutActions` API operation. You can import up to 10 actions at once. This section assumes that you have already created an Actions dataset. For information about creating datasets, see Creating a dataset and a schema.

Use the following `put-actions` command to add one or more actions with the AWS CLI. Replace `dataset arn` with the Amazon Resource Name (ARN) of your dataset and `actionId` with the ID of the action. If an action with the same `actionId` is already in your Actions dataset, Amazon Personalize replaces it with the new one.

For `properties`, for each field in your Actions dataset, replace the `propertyName` with the field name from your schema in camel case. For example, ACTION_EXPIRATION_TIMESTAMP would be `actionExpirationTimestamp` and CREATION_TIMESTAMP would be creationTimestamp. Replace `property data` with the data for the property.

```
aws personalize-events put-actions \
  --dataset-arn dataset arn \
  --actions '[{
```

```
      "actionId": "actionId",
      "properties": "{\"propertyName\": "\property data\"}"
    },
    {
      "actionId": "actionId",
      "properties": "{\"propertyName\": "\property data\"}"
    }]'
```

Importing actions individually (AWS SDKs)

Add one or more actions to your Actions dataset using the PutActions operation. You can import up to 10 actions with a single `PutActions` call. If an action with the same `actionId` is already in your Actions dataset, Amazon Personalize replaces it with the new one. This section assumes that you have already created an Actions dataset. For information about creating datasets, see [Creating a dataset and a schema](#).

The following code shows how to add one or more actions to your Actions dataset. For each action, specify the `actionId`. If an action with the same `actionId` is already in your Actions dataset, Amazon Personalize replaces it with the new one. For `properties`, for each additional field in your Actions dataset, replace the `propertyName` with the field name from your schema in camel case. For example, ACTION_EXPIRATION_TIMESTAMP would be `actionExpirationTimestamp` and CREATION_TIMESTAMP would be creationTimestamp. Replace `property data` with the data for the property.

```
import boto3

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

personalize_events.put_actions(
    datasetArn = 'dataset arn',
    actions = [{
      'actionId': 'actionId',
      'properties': "{\"propertyName\": \"property value\"}"
      },
      {
      'actionId': 'actionId',
      'properties': "{\"propertyName\": \"property value\"}"
      }]
)
```

Step 3: Creating recommenders or custom resources

After you import data, you are ready to create recommenders or custom resources. Use these resources to get recommendations. The resources you create depend on your dataset group type:

- For Domain dataset groups, you create recommenders for pre-defined use cases based on your domain. You use the recommenders to get recommendations. For information about available use cases, see Choosing a use case. You can also add custom resources to a Domain dataset group. These include solutions and solution versions trained for custom use cases.
- For Custom dataset groups, you configure a solution with a recipe. Then you create a solution version (train a model). For information about available recipes, see Choosing a recipe.

  For real-time recommendations, you deploy the solution version in a campaign. For batch recommendations and user segments, you don't need a campaign.

Topics

- Creating domain recommenders
- Creating custom resources

Creating domain recommenders

After you import data, you are ready to start creating, evaluating, and managing recommenders in your Domain dataset group. A *recommender* is a Domain dataset group resource that generates recommendations. You use it in your application to get real-time recommendations with the GetRecommendations operation.

Topics

- Creating recommenders
- Evaluating a recommender
- Managing recommenders

Creating recommenders

After you create a Domain dataset group and import data, you can create recommenders for your domain use cases. A *recommender* is a Domain dataset group resource that generates recommendations. You use a recommender in your application to get real-time recommendations with the GetRecommendations operation.

When you create a recommender, you specify a use case and Amazon Personalize trains the models backing the recommender with the best configurations for the use case. Each use case has different API requirements for getting recommendations. For a list of recommender use cases by domain, see Choosing a use case. You can create at most 15 recommenders per region.

Amazon Personalize automatically retrains the models backing your recommenders every 7 days. This is a full retraining that creates entirely new models based on the entirety of the data in your datasets. With *Top picks for you* and *Recommended for you* use cases, Amazon Personalize updates the existing models every two hours to include new items in recommendations with exploration.

When you create a recommender, you can enable item metadata in recommendations. For more information, see Enabling metadata in recommendations.

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

Recommender statuses

A recommender 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 recommender status, navigate to the Recommenders page in the Amazon Personalize console or use the DescribeRecommender operation.

Topics

- Minimum recommendation requests per second and auto-scaling
- Enabling metadata in recommendations
- Creating recommenders (console)
- Creating recommenders (AWS CLI)
- Creating recommenders (AWS SDKs)

Minimum recommendation requests per second and auto-scaling

> **Important**
>
> A high `minRecommendationRequestsPerSecond` will increase your bill. We recommend starting with 1 for `minRecommendationRequestsPerSecond` (the default). Track your usage using Amazon CloudWatch metrics, and increase the `minRecommendationRequestsPerSecond` as necessary.

When you create a recommender, you can configure the recommender's minimum recommendation requests per second. The minimum recommendation requests per second (`minRecommendationRequestsPerSecond`) specifies the baseline recommendation request throughput provisioned by Amazon Personalize. The default minRecommendationRequestsPerSecond is `1`. A recommendation request is a single `GetRecommendations` operation. Request throughput is measured in requests per second and Amazon Personalize uses your requests per second to derive your requests per hour and the price of your recommender usage.

If your requests per second increases beyond `minRecommendationRequestsPerSecond`, Amazon Personalize auto-scales the provisioned capacity up and down, but never below `minRecommendationRequestsPerSecond`. There's a short time delay while the capacity is increased that might cause loss of requests.

Your bill is the greater of either the minimum requests per hour (based on minRecommendationRequestsPerSecond) or the actual number of requests. The actual request throughput used is calculated as the average requests/second within a one-hour window. We recommend starting with the default `minRecommendationRequestsPerSecond`, track your usage using Amazon CloudWatch metrics, and then increase the `minRecommendationRequestsPerSecond` as necessary.

Enabling metadata in recommendations

> **Important**
>
> When you enable metadata in recommendations, you incur additional costs. For more information see [Amazon Personalize pricing](https://aws.amazon.com/personalize/pricing/).

When you create a recommender, you can enable the option to include item metadata from your Items dataset with recommendation results. If enabled, you can specify the columns from your Items dataset in your request for recommendations. Amazon Personalize returns this data for each item in the recommendation response.

You might use metadata to enrich recommendations in your user interface, such as adding the genres for movies to carousels. Or you might use it to visually assess recommendation quality. If you use generative AI in your app, you can plug the metadata into AI prompts to generate more relevant content. For more information about using Amazon Personalize with generative AI, see Amazon Personalize and generative AI.

- To enable metadata with the Amazon Personalize console, when you create the recommender, choose **Return items metadata in recommendation results** in **Advanced configuration**. For more information, see Creating recommenders (console).
- To enable metadata with the AWS SDKs or AWS CLI, use the CreateRecommender API operation and in the `recommenderConfig` set `enableMetadataWithRecommendations` to `true`. For more information, see Creating recommenders (AWS CLI) or Creating recommenders (AWS SDKs).

To add metadata to recommendations, you must have an Items dataset with a column of metadata. You don't have to use the metadata in training. For information about creating a dataset, see Creating a dataset and a schema. For information managing and updating data, see Managing data.

Creating recommenders (console)

> **Important**
>
> A high `minRecommendationRequestsPerSecond` will increase your bill. We recommend starting with 1 for `minRecommendationRequestsPerSecond` (the default). Track your usage using Amazon CloudWatch metrics, and increase the `minRecommendationRequestsPerSecond` as necessary. For more information see Minimum recommendation requests per second and auto-scaling.

Create recommenders for each of your use cases with the Amazon Personalize console as follows. If you just created your Domain dataset group and you are already on the **Overview** page, skip to step 3.

To create recommenders

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 Domain dataset group.
3. In **Step 3**, choose **Use <domain name> recommenders** and choose **Create recommenders**.
4. On the **Choose use cases** page, choose the use cases you want to create recommenders and give each a **Recommender name**. Amazon Personalize creates a recommender for each use case that you choose. The available use cases depend on your domain. For information on choosing a use case see [Choosing a use case](#).
5. Choose **Next**.
6. On the **Advanced configuration** page, configure each recommender depending on your business needs:

   - For each dataset used by the recommender's use case, you can choose the columns Amazon Personalize considers when training the models backing your recommender. By default, Amazon Personalize uses all columns that can be used when training. Columns with the boolean data type and custom string fields that aren't categorical or textual aren't used when training. You can't exclude EVENT_TYPE columns.

     You can change the columns used when training to control what data Amazon Personalize uses when creating your recommender. You might do this to experiment with different combinations of training data. Or you might exclude columns without meaningful data. For example, you might have a column that you want to use only to filter recommendations. You can exclude this column from training and Amazon Personalize considers it only when filtering.
   - You can modify **Minimum recommendation requests per second** to specify a new minimum request capacity for your recommender. A high `minRecommendationRequestsPerSecond` will increase your bill. We recommend starting with 1 (the default). Track your usage using Amazon CloudWatch metrics, and increase the `minRecommendationRequestsPerSecond` as necessary. For more information see [Minimum recommendation requests per second and auto-scaling](#).
   - If you want the ability to include Items dataset metadata with recommendations, choose **Return items metadata in recommendation results**. If enabled, you can specify the columns from your Items dataset in your request for recommendations or personalized

ranking. Amazon Personalize returns this data for each item in the recommendation response.

To enable metadata, you must have an Items dataset with a column of metadata.

- For `Top picks for your` or `Recommended for you` use cases, optionally make changes to exploration configuration. Exploration involves testing different item recommendations to learn how users respond to items with very little interaction data. Use the following fields to configure exploration:
  - Emphasis on exploring less relevant items (exploration weight) – Configure how much to explore. Specify a decimal value between 0 to 1. The default is 0.3. The closer the value is to 1, the more exploration. With more exploration, recommendations include more items with less item interactions data or relevance based on previous behavior. At zero, no exploration occurs and recommendations are based on current data (relevance).
  - Exploration item age cutoff – Specify the maximum item age in days since the latest interaction across all items in the Item interactions dataset. This defines the scope of item exploration based on item age. Amazon Personalize determines item age based on its creation timestamp or, if creation timestamp data is missing, item interactions data. For more information how Amazon Personalize determines item age, see [Creation timestamp data](#).

    To increase the items Amazon Personalize considers during exploration, enter a greater value. The minimum is 1 day and the default is 30 days. Recommendations might include items that are older than the item age cut off you specify. This is because these items are relevant to the user and exploration didn't identify them.
- For **Tags**, optionally add any tags. For more information about tagging Amazon Personalize resources, see [Tagging Amazon Personalize resources](#).

7. To create recommenders for each of your use cases, choose **Create recommenders**.

   You can monitor the status of each recommender on the **Recommenders** page. When your recommender status is Active, you can use it in your application to get recommendations.

Creating recommenders (AWS CLI)

After you create a Domain dataset group and import data, you can create recommenders for your domain use cases. A *recommender* is a Domain dataset group resource that generates recommendations.

For `Top picks for your` or `Recommended for you` use cases, Amazon Personalize uses exploration when recommending items. For more information, see Configuring exploration.

Topics

- Creating a recommender
- Configuring columns used when training
- Configuring exploration
- Enabling metadata in recommendations

Creating a recommender

Use the following AWS CLI code to create a recommender for a domain use case. Run this code for each of your domain use cases. For `recipeArn`, provide the Amazon Resource Name (ARN) for your use case. The available use cases depend on your domain. For a list of use cases and their ARNs see Choosing a use case.

```
aws personalize create-recommender \
--name recommender name \
--dataset-group-arn dataset group ARN \
--recipe-arn recipe ARN
```

Configuring columns used when training

You can modify the columns Amazon Personalize considers when training the models backing your recommender. By default, Amazon Personalize uses all columns that can be used when training. Columns with the boolean data type and custom non-categorical string fields aren't used. You can't exclude EVENT_TYPE columns.

You can change the columns used when training to control what data Amazon Personalize uses when creating your recommender. You might do this to experiment with different combinations of training data. Or you might exclude columns without meaningful data. For example, you might have a column that you want to use only to filter recommendations. You can exclude this column from training and Amazon Personalize considers it only when filtering.

To exclude columns from training, provide the `excludedDatasetColumns` object in the `trainingDataConfig` as part of the recommender configuration. For each key in the object, provide the dataset type. For each value, provide the list of columns to exclude.

```
aws personalize create-recommender \
```

```
--name recommender name \
--dataset-group-arn dataset group ARN \
--recipe-arn recipe ARN \
--recommender-config "{\"trainingDataConfig\": {\"excludedDatasetColumns\":
 { \"datasetType\" : [ \"column1Name\", \"column2Name\"]}}}"
```

Configuring exploration

For `Top picks for your` or `Recommended for you` use cases, Amazon Personalize uses exploration when recommending items. Exploration involves testing different item recommendations to learn how users respond to items with very little interaction data. You can configure exploration with the following:

- Emphasis on exploring less relevant items (exploration weight) – Configure how much to explore. Specify a decimal value between 0 to 1. The default is 0.3. The closer the value is to 1, the more exploration. With more exploration, recommendations include more items with less item interactions data or relevance based on previous behavior. At zero, no exploration occurs and recommendations are based on current data (relevance).
- Exploration item age cutoff – Specify the maximum item age in days since the latest interaction across all items in the Item interactions dataset. This defines the scope of item exploration based on item age. Amazon Personalize determines item age based on its creation timestamp or, if creation timestamp data is missing, item interactions data. For more information how Amazon Personalize determines item age, see Creation timestamp data.

  To increase the items Amazon Personalize considers during exploration, enter a greater value. The minimum is 1 day and the default is 30 days. Recommendations might include items that are older than the item age cut off you specify. This is because these items are relevant to the user and exploration didn't identify them.

The following code shows how to configure exploration when you create a recommender for the `Top picks for you` use case. The example uses the default values.

If you have an Items dataset and want the option to include metadata when you get recommendations, update the `recommender-config` to add a `enableMetadataWithRecommendations` field and set it to `true`.

```
aws personalize create-recommender \
--name recommender name \
--dataset-group-arn dataset group ARN \
```

```
--recipe-arn arn:aws:personalize:::recipe/aws-vod-top-picks \
--recommender-config "{\"itemExplorationConfig\":{\"explorationWeight\":\"0.3\",
\"explorationItemAgeCutOff\":\"30\"}}"
```

Enabling metadata in recommendations

If you have an Items dataset and want the option to include metadata when you get recommendations, set `enableMetadataWithRecommendations` to `true` in the `recommender-config`.

```
aws personalize create-recommender \
--name recommender name \
--dataset-group-arn dataset group \
--recipe-arn recipe ARN \
--recommender-config "{\"enableMetadataWithRecommendations\": "true"}"
```

Creating recommenders (AWS SDKs)

After you create a Domain dataset group and import data, you can create recommenders for your domain use cases. A *recommender* is a Domain dataset group resource that generates recommendations.

For all use cases, you can configure the columns used when training. For more information, see Configuring columns used when training. For `Top picks for your` or `Recommended for you` use cases, Amazon Personalize uses exploration when recommending items. For more information, see Configuring exploration.

Topics

- Creating a recommender
- Configuring exploration
- Configuring columns used when training
- Enabling metadata

Creating a recommender

Create a recommender for a domain use case with the following code. Give your recommender a name and provide your Domain dataset group's Amazon Resource Name (ARN). For `recipeArn`, provide the ARN for your use case. Run this code for each of your domain use cases. The available use cases depend on your domain. For a list of use cases, their ARNs, and their requirements, see Choosing a use case.

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

create_recommender_response = personalize.create_recommender(
  name = 'recommender name',
  recipeArn = 'recipe ARN',
  datasetGroupArn = 'dataset group ARN'
)

recommender_arn = create_recommender_response['recommenderArn']

print('Recommender ARN:' + recommender_arn)
```

SDK for Java 2.x

```
    public static String createRecommender(PersonalizeClient personalizeClient,
            String name,
            String datasetGroupArn,
            String recipeArn) {

        long maxTime = 0;
        long waitInMilliseconds = 30 * 1000; // 30 seconds
        String recommenderStatus = "";

        try {
            CreateRecommenderRequest createRecommenderRequest =
 CreateRecommenderRequest.builder()
                    .datasetGroupArn(datasetGroupArn)
                    .name(name)
                    .recipeArn(recipeArn)
                    .build();

            CreateRecommenderResponse recommenderResponse = personalizeClient
                    .createRecommender(createRecommenderRequest);
            String recommenderArn = recommenderResponse.recommenderArn();
            System.out.println("The recommender ARN is " + recommenderArn);
```

```
            DescribeRecommenderRequest describeRecommenderRequest =
 DescribeRecommenderRequest.builder()
                    .recommenderArn(recommenderArn)
                    .build();

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

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

                recommenderStatus =
 personalizeClient.describeRecommender(describeRecommenderRequest).recommender()
                        .status();
                System.out.println("Recommender status: " + recommenderStatus);

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

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

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { CreateRecommenderCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});
```

```
// set the recommender's parameters
export const createRecommenderParam = {
  name: "RECOMMENDER_NAME",                    /* required */
  recipeArn: "RECIPE_ARN",                     /* required */
  datasetGroupArn: "DATASET_GROUP_ARN"         /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateRecommenderCommand(createRecommenderParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Configuring exploration

For `Top picks for your` or `Recommended for you` use cases, Amazon Personalize uses exploration when recommending items. Exploration involves testing different item recommendations to learn how users respond to items with very little interaction data. You can configure exploration with the following:

- Emphasis on exploring less relevant items (exploration weight) – Configure how much to explore. Specify a decimal value between 0 to 1. The default is 0.3. The closer the value is to 1, the more exploration. With more exploration, recommendations include more items with less item interactions data or relevance based on previous behavior. At zero, no exploration occurs and recommendations are based on current data (relevance).
- Exploration item age cutoff – Specify the maximum item age in days since the latest interaction across all items in the Item interactions dataset. This defines the scope of item exploration based on item age. Amazon Personalize determines item age based on its creation timestamp or, if creation timestamp data is missing, item interactions data. For more information how Amazon Personalize determines item age, see Creation timestamp data.

  To increase the items Amazon Personalize considers during exploration, enter a greater value. The minimum is 1 day and the default is 30 days. Recommendations might include items that are older than the item age cut off you specify. This is because these items are relevant to the user and exploration didn't identify them.

The following code shows how to configure exploration when you create a recommender. The example uses the default values.

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

create_recommender_response = personalize.create_recommender(
  name = 'recommender name',
  recipeArn = 'arn:aws:personalize:::recipe/aws-vod-top-picks',
  datasetGroupArn = 'dataset group ARN',
  recommenderConfig = {"itemExplorationConfig": {"explorationWeight": "0.3",
 "explorationItemAgeCutOff": "30"}}
)

recommender_arn = create_recommender_response['recommenderArn']

print('Recommender ARN:' + recommender_arn)
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { CreateRecommenderCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// set the recommender's parameters
export const createRecommenderParam = {
  name: "RECOMMENDER_NAME",                      /* required */
  recipeArn: "RECIPE_ARN",                       /* required */
  datasetGroupArn: "DATASET_GROUP_ARN",          /* required */
  recommenderConfig: {
    itemExplorationConfig: {
```

```
      explorationWeight: "0.3",
      explorationItemAgeCutOff: "30"
    }
  }
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateRecommenderCommand(createRecommenderParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Configuring columns used when training

You can modify the columns Amazon Personalize considers when training the models backing your recommender. By default, Amazon Personalize uses all columns that can be used when training. Columns with the boolean data type and custom non-categorical string fields aren't used. You can't exclude EVENT_TYPE columns.

You can change the columns used when training to control what data Amazon Personalize uses when creating your recommender. You might do this to experiment with different combinations of training data. Or you might exclude columns without meaningful data. For example, you might have a column that you want to use only to filter recommendations. You can exclude this column from training and Amazon Personalize considers it only when filtering.

To exclude columns from training, provide the `excludedDatasetColumns` object in the `trainingDataConfig` as part of the recommender configuration. For each key, provide the dataset type. For each value, provide the list of columns to exclude. The following code shows how to exclude columns from training when you create a recommender.

SDK for Python (Boto3)

```
import boto3

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

```
create_recommender_response = personalize.create_recommender(
  name = 'recommender name',
  recipeArn = 'recipe name',
  datasetGroupArn = 'dataset group ARN',
  recommenderConfig = {
    "trainingDataConfig": {
      "excludedDatasetColumns": {
        "datasetType": ["COLUMN_A", "COLUMN_B"]
      }
    }
  }
)

recommender_arn = create_recommender_response['recommenderArn']

print('Recommender ARN:' + recommender_arn)
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { CreateRecommenderCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// set the recommender's parameters
export const createRecommenderParam = {
  name: "RECOMMENDER_NAME",              /* required */
  recipeArn: "RECIPE_ARN",               /* required */
  datasetGroupArn: "DATASET_GROUP_ARN", /* required */
  recommenderConfig: {
    trainingDataConfig: {
      excludedDatasetColumns: {
        "DATASET_TYPE": ["COLUMN_A", "COLUMN_B"]
      }
    }
  }
};
```

```
export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateRecommenderCommand(createRecommenderParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Enabling metadata

If you have an Items dataset and want the option to include metadata when you get recommendations, set `enableMetadataWithRecommendations` to `true` in the `recommender-config`.

```
import boto3

personalize = boto3.client('personalize')

create_recommender_response = personalize.create_recommender(
  name = 'recommender name',
  recipeArn = 'recipe name',
  datasetGroupArn = 'dataset group ARN',
  recommenderConfig = {"enableMetadataWithRecommendations": True}
)

recommender_arn = create_recommender_response['recommenderArn']

print('Recommender ARN:' + recommender_arn)
```

Evaluating a recommender

You can evaluate the performance of your recommender 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 that Amazon Personalize generates when you create a recommender. You can use offline metrics to evaluate the performance of the recommender's underlying model. Offline metrics allow you to compare the model with other models trained on the same data. 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 a training set and a testing set. The training set consists of 90% of your users and their interactions data. The testing set consists of the remaining 10% of users and their interactions data.

Amazon Personalize then creates the recommender using the training set. After training completes, Amazon Personalize gives the new recommender the oldest 90% of each user's data from the testing set as input. Amazon Personalize then calculates metrics by comparing the recommendations the recommender generates to the actual interactions in the newest 10% of each user's data from the testing set.

Topics

- Retrieving metrics
- Metric definitions
- Example
- Additional resources

Retrieving metrics

After your recommender is active, you can view the metrics for the recommender in the Amazon Personalize console or retrieve metrics by calling the DescribeRecommender operation.

Topics

- Viewing metrics (console)
- Retrieving metrics (AWS CLI)
- Retrieving metrics (AWS SDKs)

Viewing metrics (console)

To view recommender metrics in the console, you navigate to the details page for your recommender.

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 Domain dataset group.
3. From the navigation pane, choose **Recommenders**.
4. From the list of recommenders, choose the one to view its metrics.

Retrieving metrics (AWS CLI)

The following code shows how to get metrics for a recommender with the AWS CLI.

```
aws personalize describe-recommender \
--recommender-arn recommender arn
```

The following is an example of the metrics output from a recommender created for the *Top picks for you* use case for the VIDEO_ON_DEMAND domain.

```
{
    "recommender": {
        "recommenderArn": "arn:aws:personalize:region:acct-id:recommender/
recommenderName",
        "datasetGroupArn": "arn:aws:personalize:region:acct-id:dataset-group/
dsGroupName",
        "name": "name123",
        "recipeArn": "arn:aws:personalize:::recipe/aws-vod-top-picks",
        "modelMetrics": {
            "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,
        }
        "recommenderConfig": {},
        "creationDateTime": "2022-05-06T10:11:24.589000-07:00",
        "lastUpdatedDateTime": "2022-05-06T10:34:33.270000-07:00",
        "status": "ACTIVE",
    }
}
```

Retrieving metrics (AWS SDKs)

The following code shows how to get metrics for a recommender with the SDK for Python (Boto3).

```
import boto3

personalize = boto3.client('personalize')

response = personalize.describe_recommender(
    recommenderArn = 'recommender_arn'
)
print(response['recommender']['modelMetrics'])
```

The following is an example the metrics output from a recommender created for the *Top picks for you* use case for the VIDEO_ON_DEMAND domain.

```
{
    "recommender": {
        "recommenderArn": "arn:aws:personalize:region:acct-id:recommender/
recommenderName",
        "datasetGroupArn": "arn:aws:personalize:region:acct-id:dataset-group/
dsGroupName",
        "name": "name123",
        "recipeArn": "arn:aws:personalize:::recipe/aws-vod-top-picks",
        "modelMetrics": {
            "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,
        }
        "recommenderConfig": {},
        "creationDateTime": "2022-05-06T10:11:24.589000-07:00",
        "lastUpdatedDateTime": "2022-05-06T10:34:33.270000-07:00",
        "status": "ACTIVE",
    }
}
```

Metric definitions

The metrics Amazon Personalize generates for recommenders are described below using the following terms:

- *Relevant recommendation* is a recommendation for an item that the user actually interacted with. These items are from the newest 10% of each user's interactions data from the testing set.
- *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.

For each metric, higher numbers (closer to 1) are better. To dive deeper, see the resources listed in [Additional resources](#).

coverage

The value for *coverage* tells you the proportion of unique items that Amazon Personalize might recommend out of the total number of unique items in Interactions and Items datasets. A higher coverage score means Amazon Personalize recommends more of your items, rather than the same few items repeatedly for different users. Use cases that feature item exploration, such as the *Top picks for you* (VIDEO_ON_DEMAND) and *Recommended for you* (ECOMMERCE), have higher coverage than those that don't.

mean reciprocal rank at 25

This metric tells you about a model's ability to generate a relevant recommendation at the top ranked position. You might choose a model with a high *mean reciprocal rank at 25* if you are generating relevant search results for a user, and don't expect the user to choose an item lower on the list. For example, users frequently choose the first cooking recipe in search results.

Amazon Personalize calculates this metric using the average reciprocal rank score for requests for recommendations. Each reciprocal rank score is calculated as follows: `1 / the rank of the highest item interacted with by the user`, where the total possible rankings is 25. Other lower ranked items the user interacts with are ignored. If the user chose the first item, the score is 1. If they don't choose any items, the score is 0.

For example, you might show three different users 25 recommendations each:

- If User 1 clicks the item at rank *4* and the item at rank *10*, their reciprocal rank score is *1/4*.
- If User 2 clicks an item at rank *2*, an item at rank *4*, and an item at rank *12*, their reciprocal rank score is 1/2.

- If User 3 clicks on a single item at rank 6, their reciprocal rank score is 1/6.

The mean reciprocal rank over all requests for recommendations (in this case 3) is calculated as `(1/4 + 1/2 + 1/6) / 3 = .3056`.

normalized discounted cumulative gain (NDCG) at K (5, 10, or 25)

This metric tells you about how well your model ranks recommendations, where K is a sample size of 5, 10, or 25 recommendations. This metric is useful if you are most interested in the ranking of recommendations beyond just the highest ranked item (for this, see `mean reciprocal rank at 25`). For example, the score for `NDCG at 10` would be useful if you have an application that shows up to 10 movies in a carousel at a time.

Amazon Personalize calculates the NDCG by assigning weight to recommendations based on their ranking position for each user in the testing set. Each recommendation is discounted (given a lower weight) by a factor dependent on its position. The final metric is the average of all users in the testing set. 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 + position)`, where the top of the list is position `1`.

precision at K

This metric 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 for each user in the testing set, divided by K, where K is 5, 10, or 25. The final metric is the average across all users in the testing set.

For example, if you recommend 10 items to a user, and the user interacts with 3 of them, the precision at K is 3 correctly predicted items divided by the total 10 recommended items: `3 / 10 = .30`.

This metric rewards precise recommendation of relevant items. The closer the score is to one, the more precise the model.

Example

The following is a simple example for a recommender that 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

Additional resources

To dive deeper in different types of metrics for recommender systems, see the following external resources:

- MRR vs MAP vs NDCG: Rank-Aware Evaluation Metrics And When To Use Them
- Discounted Cumulative Gain: the ranking metrics you should know about
- Recall and Precision at k for Recommender Systems
- Ranking Evaluation Metrics for Recommender Systems

Managing recommenders

You don't have to manage the models backing your recommenders. Amazon Personalize automatically retrains them every 7 days. This is a full retraining that creates entirely new models based on the entirety of the data in your datasets. For *Top picks for you* and *Recommended for you*, Amazon Personalize updates the existing models every two hours to consider new items for recommendations with exploration. For more information, see Automatic updates.

Managing recommenders involves the following:

- **Stopping and starting recommenders –** If you want to pause billing for an active recommender, you can stop the recommender and restart it later. For more information, see Stopping and starting a recommender.
- **Updating recommender configuration –** You can update the columns the recommender uses in training and update the recommender's request capacity. For more information, see Updating a recommender.
- **Deleting a recommender –** You can delete recommenders with the DeleteRecommender operation. Or you can delete a recommender from the recommender details page in the Amazon Personalize console.

Topics

Updating a recommender

After you create a recommender, you can update the recommender's configuration:

- You can update the columns the recommender uses in training. If you modify the columns used when training, Amazon Personalize automatically starts a full retraining of the models backing your recommender. While the update completes, you can still get recommendations from the recommender. The recommender uses the previous configuration until the update completes. To track the status of this update, use the `latestRecommenderUpdate` returned in the DescribeRecommender operation. If you provide the same columns you provided when you created the recommender, no update occurs.
- You can update the recommender's minimum recommendation requests per second. This specifies the baseline recommendation request throughput that's provisioned by Amazon Personalize. A high value will increase your bill. We recommend starting with 1. Track your usage using Amazon CloudWatch metrics, and increase it as necessary. For more information, see Minimum recommendation requests per second and auto-scaling.

- For *Top picks for you* and *Recommended for you* use cases, you can update exploration configuration by adjusting the emphasis on exploring relevant items and the exploration item age cutoff. For information about exploration, see the section for your use case in Choosing a use case.

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

Topics

- Updating a recommender (Amazon Personalize console)
- Updating a recommender (AWS CLI)
- Updating a recommender (AWS SDKs)

Updating a recommender (Amazon Personalize console)

After you create a recommender, you can update it. You can update the columns the recommender uses in training and the recommender's minimum recommendation requests per second. For *Top picks for you* and *Recommended for you* use cases, you can update exploration configuration. To update a recommender with the console, do the following.

To update a recommender's configuration (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 Domain dataset group.
3. From the navigation pane, choose **Recommenders**.
4. On the **Recommenders** page, choose the recommender that you want to update.
5. In **Recommender configuration** choose **Edit**.
6. Change the recommender's configuration and choose **Update**. For information on the different configuration options, see Creating recommenders (console).

Updating a recommender (AWS CLI)

To update recommender with the AWS CLI, use the `update-recommender` command. Provide the Amazon Resource Name (ARN) for the recommender and updated configuration. The following code shows how to update the columns a recommender uses for training.

```
aws personalize update-recommender \
--dataset-group-arn dataset group ARN \
--recommender-config "{\"trainingDataConfig\": {\"excludedDatasetColumns\":
 { \"datasetType\" : [ \"column1Name\", \"column2Name\"]}}}"
```

If you modify the columns used in training, Amazon Personalize automatically starts a full retraining of the models backing your recommender. While the update completes, you can still get recommendations from the recommender. The recommender uses the previous configuration until the update completes. To track the status of this update, use the `latestRecommenderUpdate` returned in the DescribeRecommender operation.

For more information about the different configurations you can change, see RecommenderConfig.

Updating a recommender (AWS SDKs)

To update recommender with the AWS, use the UpdateRecommender operation. Provide the Amazon Resource Name (ARN) for the recommender and specify the new configuration. The following code shows how to update the columns a recommender uses for training.

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

update_recommender_response = personalize.update_recommender(
  recommenderArn = 'dataset group ARN',
  recommenderConfig = {
    "trainingDataConfig": {
      "excludedDatasetColumns": {
        "datasetType": ["COLUMN_A", "COLUMN_B"]
      }
    }
  }
)
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { UpdateRecommenderCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";
```

```
// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// set the request's parameters
export const updateRecommenderParam = {
  recommenderArn: "RECOMMENDER_ARN", /* required */
  recommenderConfig: {
    trainingDataConfig: {
      excludedDatasetColumns: {
        "DATASET_TYPE": ["COLUMN_A", "COLUMN_B"]
      }
    }
  }
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 UpdateRecommenderCommand(updateRecommenderParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

If you modify the columns used in training in the `excludedDatasetColumns` of the `recommenderConfig`, Amazon Personalize automatically starts a full retraining of the models backing your recommender. While the update completes, you can still get recommendations from the recommender. The recommender uses the previous configuration until the update completes. To track the status of this update, use the `latestRecommenderUpdate` returned in the [DescribeRecommender](https://docs.aws.amazon.com/personalize/latest/dg/API_DescribeRecommender.html) operation.

For more information about the different configurations you can change, see [RecommenderConfig](https://docs.aws.amazon.com/personalize/latest/dg/API_RecommenderConfig.html).

Stopping and starting a recommender

After your recommender is active, you can stop a recommender and start it later. This way, you can pause recommender billing and only pay for it when you use it. For example, you might need to get recommendations only on certain days of the week. You can stop the recommender on the days you don't need it, and then start the recommender on the days you do.

After you stop a recommender, you can't use it to get recommendations. Stopping a recommender halts recommender billing and retraining. However, stopping a recommender doesn't delete the recommender. You can restart it at any time and resume getting recommendations. Starting a recommender doesn't create a new recommender with your data. Rather, it resumes recommender billing and retraining every 7 days.

You can stop and start a recommender with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), AWS SDKs, or the StartRecommender and StopRecommender API operations.

Recommender states

When you stop a recommender, the recommender state changes from ACTIVE to INACTIVE in the following sequence:

ACTIVE > STOP PENDING > STOP IN PROGRESS > INACTIVE

When you start a recommender, the recommender state changes from INACTIVE to ACTIVE in the following sequence:

INACTIVE > START PENDING > START IN PROGRESS > ACTIVE

Topics

- Stopping and starting a recommender (console)
- Stopping and restarting a recommender (AWS CLI)
- Stopping and restarting a recommender (AWS SDKs)

Stopping and starting a recommender (console)

You can use the Amazon Personalize to stop and restart a recommender.

Topics

- Stopping a recommender (console)
- Starting a recommender (console)

Stopping a recommender (console)

You can use the Amazon Personalize console to stop an active recommender as follows.

To stop a recommender

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 Domain dataset group.
3. From the navigation pane, choose **Recommenders**.
4. On the **Recommenders** page, choose the recommender that you want to stop.
5. Choose **Stop recommender** at the top right and confirm on the window that displays.

   When the recommender status is inactive, your recommender has stopped. This halts any recommender billing and retraining. You can't use the recommender until you start it.

Starting a recommender (console)

You can use the Amazon Personalize console to start an inactive recommender as follows.

To start a recommender

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 Domain dataset group.
3. From the navigation pane, choose **Recommenders**.
4. On the **Recommenders** page, choose the recommender that you want to start.
5. Choose **Start recommender** at the top right and confirm that you want to start it on the window that displays.

   When the recommender status is active, you can resume getting recommendations from it. Recommender billing and automatic retraining also resumes.

Stopping and restarting a recommender (AWS CLI)

To stop an active recommender with the AWS CLI, use the `stop-recommender` command and provide the Amazon Resource Name (ARN) for the recommender as follows:

```
aws personalize stop-recommender --recommender-arn "recommender arn"
```

To start an inactive recommender with the AWS CLI, use the `start-recommender` command and provide the ARN for the stopped recommender as follows:

```
aws personalize start-recommender --recommender-arn "recommender arn"
```

For more information about the API operations, see StartRecommender and StopRecommender.

Stopping and restarting a recommender (AWS SDKs)

You can use the AWS SDKs to start an active recommender or stop an inactive recommender. For more information about the API operations, see StartRecommender and StopRecommender.

Topics

Stopping a recommender (AWS SDKs)

The following code shows how to stop an active recommender with the AWS SDKs. Stopping halts any recommender billing and automatic retraining. You can't use the recommender until you restart it.

SDK for Python (Boto3)

To stop an active recommender with the SDK for Python (Boto3), use the `stop_recommender` method and provide the Amazon Resource Name (ARN) for the recommender as follows.

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

stop_recommender_response = personalize stop_recommender(
    recommenderArn = "recommenderARN"
)
print(stop_recommender_response)
```

SDK for Java 2.x

To stop an active recommender with the SDK for Java 2.x, use the `stopRecommender` method and provide the ARN for the recommender as follows.

```
public static void stopRecommender(PersonalizeClient personalizeClient,
                                                     String datasetGroupArn) {

    try {

        StopRecommenderRequest stopRecommenderRequest =
 StopRecommenderRequest.builder()
                .recommenderArn(recommenderArn)
                .build();
        personalizeClient.stopRecommender(stopRecommenderRequest);
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { StopRecommenderCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// set the request params
export const stopRecommenderParam = {
  recommenderArn: "RECOMMENDER_ARN" /* required */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new StopRecommenderCommand(stopRecommenderParam)
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
```

```
run();
```

Starting a recommender (AWS SDKs)

The following code shows how to start an inactive recommender with the AWS SDKs. When the recommender status is active, you can resume getting recommendations from it. At the same time, recommender billing and automatic retraining also resumes.

SDK for Python (Boto3)

To start an inactive recommender with the SDK for Python (Boto3), use the `start_recommender` method and provide the Amazon Resource Name (ARN) for the recommender as follows.

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

start_recommender_response = personalize start_recommender(
    recommenderArn = "recommenderARN"
)
print(start_recommender_response)
```

SDK for Java 2.x

To start an inactive recommender with the SDK for Java 2.x, use the `startRecommender` method and provide the ARN for the recommender as follows.

```
public static void startRecommender(PersonalizeClient personalizeClient,
                                                      String datasetGroupArn) {


    try {

        StartRecommenderRequest startRecommenderRequest =
 StartRecommenderRequest.builder()
                .recommenderArn(recommenderArn)
                .build();
        personalizeClient.startRecommender(startRecommenderRequest);
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
```

```
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { StartRecommenderCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// set the request params
export const startRecommenderParam = {
  recommenderArn: "RECOMMENDER_ARN" /* required */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new StartRecommenderCommand(startRecommenderParam)
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Creating custom resources

After you import your data, you are ready to create the custom resources that you use to get recommendations. To create the custom resources that generate recommendations, you do the following:

1. **Create and configure a solution:** Customize solution parameters and recipe-specific hyperparameters so the model meets your specific business needs. By default, new solution versions use automatic training to create solution versions at a configurable frequency. The default frequency is every 7 days.

2. **Create a solution version (train a model):** For solutions that use automatic training, solution version creation starts automatically after your solution is active. For solutions that use manual training, you manually create a solution version. The solution version generates Amazon Personalize recommendations or user segments.
3. **Deploy the solution version with a campaign (only for real-time recommendations):** Create a campaign to deploy your solution version. You use the campaign when you request real-time recommendations. If you are getting batch recommendations, you don't need to create a campaign.

Topics

- Creating a solution and a solution version
- Creating a campaign

Creating a solution and a solution version

After you finish importing data, 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).

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

1. **Create a solution** – Customize solution parameters and recipe-specific hyperparameters so the model meets your specific business needs. See Creating and configuring a solution. For a list of available recipes, see Choosing a recipe.

   By default, new solution versions use automatic training to create solution versions every 7 days. You can configure the training frequency.
2. **Create a solution version (train a model):** For solutions that use automatic training, solution version creation starts automatically after your solution is active. You can still create solution versions manually. For solutions with automatic training turned off, you manually create a solution version. See Creating a solution version.
3. **Evaluate the solution version** – Use the metrics Amazon Personalize generates from the new solution version to evaluate the performance of the model. See Evaluating a solution version with metrics.

Topics

Creating and configuring a solution

After you finish importing data, you are ready to create a solution. A *solution* refers to the combination of an Amazon Personalize recipe, customized training parameters, and one or more solution versions. A *solution version* refers to a trained machine learning model. By default, all new solutions use automatic training to create a new solution version every 7 days. For more information, see Configuring automatic training.

If you have an existing solution, you can use the Amazon Personalize console to clone the solution. When you clone a solution, you can use the configuration of the existing solution as a starting point, such as the recipe and hyperparameters, and make any changes. For more information, see Cloning a solution (console).

You can create and configure a solution by using the console, AWS Command Line Interface (AWS CLI), or AWS SDKs. After you create a solution, you can view its configuration details on the solution's details page of the Amazon Personalize console, or with the DescribeSolution operation.

After you create a solution, you can’t change its configuration. By default, all new solutions use automatic training. With automatic training, you incur training costs while your solution is active. To avoid unnecessary costs, make sure to delete the solution when you are finished. For information about training costs, see Amazon Personalize pricing.

Topics

Creating a solution (console)

> **Important**
>
> After you create a solution, you can't change its configuration. By default, all new solutions use automatic training. With automatic training, you incur training costs while your solution is active. To avoid unnecessary costs, make sure to delete the solution when you are finished. For information about training costs, see Amazon Personalize pricing.

To create a solution in the console, choose your dataset group and then specify a solution name, recipe, and optional training configuration.

To 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. On the **Dataset groups** page, choose your dataset group.
3. On the **Overview** page, for **Step 3**, do one of the following:
   - If you created a Domain dataset group, choose **Use custom resources**, and choose **Create solutions**.
   - If you created a Custom dataset group, choose **Create solutions**.
4. For **Solution name**, specify a name for your solution.
5. For **Solution type**, choose the type of solution that you want to create. The type you choose determines what recipes are available.
   - Choose **Item recommendation** to get item recommendations for your users. For example, personalized movie recommendations.
   - Choose **Action recommendation** to get action recommendations for your users. For example, generate the next best action for a user, such as download your app.
   - Choose **User segmentation** to get user segments (groups of users) based on your item data.
6. For **Recipe**, choose a recipe (see Choosing a recipe).
7. For **Tags**, optionally add any tags. For more information about tagging Amazon Personalize resources, see Tagging Amazon Personalize resources.
8. Choose **Next**.

9. On the **Training configuration** page, customize the solution to meet your business requirements.

    - In **Automatic training**, choose whether the solution uses automatic training. If you use automatic training, you can change the `Automatic training frequency`. The default training frequency is every 7 days.

      We recommend using automatic training. It makes it easier for you to maintain recommendation relevance. Your training frequency depends on your business requirements, the recipe that you use, and how frequently you import data. For more information, see Configuring automatic training. For information about maintaining relevance, see Maintaining recommendation relevance.

    - In **Hyperparameter configuration**, configure any hyperparameter options based on your recipe and business needs. Different recipes use different hyperparameters. For the hyperparameters available to you, see the individual recipes in Choosing a recipe.

    - In **Columns for training**, if your recipe generates item recommendations or user segments, optionally choose the columns that Amazon Personalize considers when creating solution versions. For more information, see Configuring columns used when training.

    - In **Additional configuration**, if your Item 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 item interactions data that Amazon Personalize uses when training the model. For more information see Choosing the item interaction data used for training.

    - If you use either the User-Personalization recipe or Personalized-Ranking recipe recipe, optionally specify an **Objective** and choose an **Objective sensitivity** to optimize your solution for an objective in addition to relevance. The objective sensitivity configures how Amazon Personalize balances recommending items based on your objective compared with relevance through interactions data. For more information, see Optimizing a solution for an additional objective.

10. Choose **Next** and review the solution details. You can't change your solution's configuration after you create it.

11. Choose **Create solution**. After you create a solution, Amazon Personalize starts creating your first solution version within an hour. To monitor training status, choose your solution and view the status in the **Solution versions** section. Automatically created solution versions have a **Training type** of AUTOMATIC.

When the solution version is ACTIVE, you are ready to use it to get recommendations. How you use an active solution version depends on how you get recommendations:

- For real-time recommendations, you deploy an ACTIVE solution version with an Amazon Personalize campaign. You use the campaign to get recommendations for your users. See Creating a campaign.
- For batch recommendations, you specify an ACTIVE solution version when you create a batch inference job or batch segment job. See Batch recommendations and user segments (custom resources).

Creating a solution (AWS CLI)

> **Important**
>
> After you create a solution, you can't change its configuration. By default, all new solutions use automatic training. With automatic training, you incur training costs while your solution is active. To avoid unnecessary costs, make sure to delete the solution when you are finished. For information about training costs, see Amazon Personalize pricing.

To create a solution with the AWS CLI, use the `create-solution` command. This command uses the CreateSolution API operation. The following code shows you how to create a solution that uses automatic training. It automatically creates a new solution version every five days.

To use the code, update it to give the solution a name, specify the Amazon Resource Name (ARN) of your dataset group, optionally change the training frequency, and specify the ARN of the recipe to use. For information about recipes, see Choosing a recipe.

```
aws personalize create-solution \
--name solution name \
--dataset-group-arn dataset group ARN \
--recipe-arn recipe ARN \
--perform-auto-training \
--solution-config "{\"autoTrainingConfig\": {\"schedulingExpression\": \"rate(5
 days)\"}}"
```

- We recommend that you use automatic training. It makes it easier for you to maintain and improve recommendation relevance. By default, all new solutions use automatic training.

The default training frequency is every 7 days. Your training frequency depends on your business requirements, the recipe that you use, and how frequently you import data. For more information, see Configuring automatic training.

- Depending on your recipe, you can modify the code to configure recipe specific properties and hyperparameters (see Hyperparameters and HPO), configure the columns used for training (see Configuring columns used when training (AWS CLI)), or filter the item interactions data used for training (see Choosing the item interaction data used for training).
- If you use either the User-Personalization recipe or Personalized-Ranking recipe recipe, you can optimize your solution for an objective, in addition to relevance. For more information, see Optimizing a solution for an additional objective.

After you create the solution, record the solution ARN for future use. With automatic training, solution version creation starts within one after the solution is ACTIVE. If you manually create a solution version within the hour, the solution skips the first automatic training. After training starts, you can get the solution version's Amazon Resource Name (ARN) with the ListSolutionVersions API operation. To get its status, use the DescribeSolutionVersion API opearation.

When the solution version is ACTIVE, you are ready to use it to get recommendations. How you use an active solution version depends on how you get recommendations:

- For real-time recommendations, you deploy an ACTIVE solution version with an Amazon Personalize campaign. You use the campaign to get recommendations for your users. See Creating a campaign.
- For batch recommendations, you specify an ACTIVE solution version when you create a batch inference job or batch segment job. See Batch recommendations and user segments (custom resources).

Creating a solution (AWS SDKs)

> **Important**
>
> After you create a solution, you can’t change its configuration. By default, all new solutions use automatic training. With automatic training, you incur training costs while your solution is active. To avoid unnecessary costs, make sure to delete the solution when you are finished. For information about training costs, see Amazon Personalize pricing.

To create a solution with AWS SDKs, use the CreateSolution API operation. The following code shows you how to create a solution that uses automatic training. It automatically creates a new solution version every five days.

To use the code, update it to give the solution a name, specify the Amazon Resource Name (ARN) of your dataset group, optionally change the training frequency, and specify the ARN of the recipe that you want to use. For information about recipes, see Choosing a recipe.

```
import boto3

personalize = boto3.client('personalize')

create_solution_response = personalize.create_solution(
  name = 'solution name',
  recipeArn = 'recipe ARN',
  datasetGroupArn = 'dataset group ARN',
  performAutoTraining = True,
  solutionConfig = {
    "autoTrainingConfig": {
      "schedulingExpression": "rate(5 days)"
    }
  }
)
solution_arn = create_solution_response['solutionArn']
print('solution_arn: ', solution_arn)
```

- We recommend that you use automatic training. It makes it easier for you to maintain and improve recommendation relevance. By default, all new solutions use automatic training. The default training frequency is every 7 days. Your training frequency depends on your business requirements, the recipe that you use, and how frequently you import data. For more information, see Configuring automatic training.
- Depending on your recipe, you can modify the code to configure recipe specific properties and hyperparameters (see Hyperparameters and HPO), configure the columns used for training (see Configuring columns used when training (AWS SDKs)), or filter the item interactions data used for training (see Choosing the item interaction data used for training).
- If you use either the User-Personalization recipe or Personalized-Ranking recipe recipe, you can optimize your solution for an objective, in addition to relevance. For more information, see Optimizing a solution for an additional objective.

After you create the solution, record the solution ARN for future use. With automatic training, solution version creation starts within one after the solution is ACTIVE. If you manually create a solution version within the hour, the solution skips the first automatic training. After training starts, you can get the solution version's Amazon Resource Name (ARN) with the ListSolutionVersions API operation. To get its status, use the DescribeSolutionVersion API opearation.

When the solution version is ACTIVE, you are ready to use it to get recommendations. How you use an active solution version depends on how you get recommendations:

- For real-time recommendations, you deploy an ACTIVE solution version with an Amazon Personalize campaign. You use the campaign to get recommendations for your users. See Creating a campaign.
- For batch recommendations, you specify an ACTIVE solution version when you create a batch inference job or batch segment job. See Batch recommendations and user segments (custom resources).

Training configuration

When you create a solution, you can configure training to meet your specific business needs:

- You can configure whether the solution uses automatic training. And you can configure the training frequency. By default, all solutions use automatic training. For more information, see Configuring automatic training.
- If your recipe generates item recommendations or user segments, you can modify the columns Amazon Personalize considers when training a model (creating a solution version). For more information, see Configuring columns used when training.
- You can configure hyperparameters to optimize the model based on your recipe and business needs. Different recipes use different hyperparameters. For information about configuring hyperparameters, see Hyperparameters and HPO. For the available hyperparameters for your recipe, see the page for your recipe in Choosing a recipe.
- If you use either the User-Personalization recipe or Personalized-Ranking recipe recipe, you can to optimize your solution for an objective in addition to relevance. For more information see Optimizing a solution for an additional objective.
- If you have event type and event value data, you can use it to choose the item interactions records Amazon Personalize considers during training. For more information see Choosing the item interaction data used for training.

Topics

- Configuring automatic training
- Configuring columns used when training
- Optimizing a solution for an additional objective
- Hyperparameters and HPO
- Choosing the item interaction data used for training

Configuring automatic training

> **Important**
>
> After you create a solution, you can't change its configuration. By default, all new solutions use automatic training. With automatic training, you incur training costs while your solution is active. To avoid unnecessary costs, make sure to delete the solution when you are finished. For information about training costs, see Amazon Personalize pricing.

When you create a solution, you can configure whether the solution uses automatic training. You can also configure the training frequency. For example, you can configure the solution to create a new solution version every five days.

By default, all new solutions use automatic training to create a new solution version every 7 days. Training continues until you delete the solution.

We recommend that you use automatic training. It makes maintaining your solution easier. It removes the manual training required for the solution to learn from your most recent data. Without automatic training, you must manually create new solution versions for the solution to learn from your most recent data. This can result in stale recommendations and a lower conversion rate. For more information about maintaining Amazon Personalize recommendations, see Maintaining recommendation relevance.

You can configure automatic training with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs. For steps on configuring automatic training with the console, see Creating a solution (console).

After you create the solution, record the solution ARN for future use. With automatic training, solution version creation starts within one after the solution is ACTIVE. If you manually create a solution version within the hour, the solution skips the first automatic training. After training starts, you can get the solution version's Amazon Resource Name (ARN) with the ListSolutionVersions API operation. To get its status, use the DescribeSolutionVersion API opearation.

When the solution version is ACTIVE, you are ready to use it to get recommendations. How you use an active solution version depends on how you get recommendations:

- For real-time recommendations, you deploy an ACTIVE solution version with an Amazon Personalize campaign. You use the campaign to get recommendations for your users. See Creating a campaign.
- For batch recommendations, you specify an ACTIVE solution version when you create a batch inference job or batch segment job. See Batch recommendations and user segments (custom resources).

Topics

- Guidelines and requirements
- Configuring automatic training (AWS CLI)
- Configuring automatic training (SDKs)

Guidelines and requirements

The following are guidelines and requirements for automatic training:

- Each training considers all of the data in your dataset group that you include in training. For information about configuring the columns used in training, see Configuring columns used when training.
- You can still manually create solution versions.
- Automatic training starts within one hour after your solution is active. If you manually create a solution version within the hour, the solution skips the first automatic training.
- Training scheduling is based on training start date. For example, if your first solution version starts training at 7:00 pm, and you use weekly training, the next solution version will start training a week later at 7:00 pm.
- For all recipes, we recommend at least a weekly training frequency. You can specify a training frequency between 1 and 30 days. The default is every 7 days.
  - If you use User-Personalization or Next-Best-Action, the solution automatically updates to consider new items or actions for recommendations. Automatic updates aren't the same as

automatic training. An automatic update doesn't create a completely new solution version, and the model doesn't learn from your most recent data. To maintain your solution, your training frequency should still be at least weekly. For more formation about automatic updates, including additional guidelines and requirements, see Automatic updates.

- If you use Trending-Now, Amazon Personalize automatically identifies the top trending items in your interactions data over a configurable interval of time. Trending-Now can recommend items added since the last training through bulk or streaming interactions data. Your training frequency should still be at least weekly. For more information, see Trending-Now recipe.
- If you don't use a recipe with automatic updates or the Trending-Now recipe, Amazon Personalize considers new items for recommendations only after the next training. For example, if you use the Similar-Items recipe and add new items daily, you would have to use a daily automatic training frequency for these items to appear in recommendations that same day.

Configuring automatic training (AWS CLI)

The following code shows you how to create a solution that automatically creates a solution version every five days. To turn off automatic training, set `perform-auto-training` to `false`.

To change the training frequency, you can modify the `schedulingExpression` in the `autoTrainingConfig`. The expression must be in `rate(value unit)` format. For the value, specify a number between 1 and 30. For the unit, specify `day` or `days`.

For a full explanation of the `create-solution` command, see Creating a solution (AWS CLI).

```
aws personalize create-solution \
--name solution name \
--dataset-group-arn dataset group ARN \
--recipe-arn recipe ARN \
--perform-auto-training \
--solution-config "{\"autoTrainingConfig\": {\"schedulingExpression\": \"rate(5
 days)\"}}"
```

Configuring automatic training (SDKs)

The following code shows you how to create a solution with automatic training with the SDK for Python (Boto3). The solution automatically creates a solution version every five days. To turn off automatic training, set `performAutoTraining` to `false`.

To change the training frequency, you can modify the `schedulingExpression` in the `autoTrainingConfig`. The expression must be in `rate(value unit)` format. For the value, specify a number between 1 and 30. For the unit, specify `day` or `days`.

For a full explanation of the CreateSolution API operation, see [Creating a solution (AWS SDKs)](#).

```
import boto3

personalize = boto3.client('personalize')

create_solution_response = personalize.create_solution(
  name = 'solution name',
  recipeArn = 'recipe ARN',
  datasetGroupArn = 'dataset group ARN',
  performAutoTraining = True,
  solutionConfig = {
    "autoTrainingConfig": {
      "schedulingExpression": "rate(5 days)"
    }
  }
)
solution_arn = create_solution_response['solutionArn']
print('solution_arn: ', solution_arn)
```

Configuring columns used when training

> **Important**
>
> After you create a solution, you can't change its configuration. By default, all new solutions use automatic training. With automatic training, you incur training costs while your solution is active. To avoid unnecessary costs, make sure to delete the solution when you are finished. For information about training costs, see [Amazon Personalize pricing](#).

If your recipe generates item recommendations or user segments, you can modify the columns Amazon Personalize considers when creating a solution version (training a model). By default, Amazon Personalize uses all columns that can be used when training. Columns with the boolean data type and custom string fields that aren't categorical or textual aren't used when training. You can't exclude EVENT_TYPE columns.

You can change the columns used when training to control what data Amazon Personalize uses when training a model (creating a solution version). You might do this to experiment with different combinations of training data. Or you might exclude columns without meaningful data. For example, might have a column that you want to use only to filter recommendations. You can exclude this column from training and Amazon Personalize considers it only when filtering.

If you have already created a solution and you want to modify the columns it uses when training, you can clone the solution. When you clone a solution, you can use the configuration of the existing solution as a starting point, such as the recipe and hyperparameters, and make any changes as necessary. For more information, see Cloning a solution (console).

You can configure the columns Amazon Personalize uses when training with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDK. For information about choosing columns with the Amazon Personalize console, see the advanced configuration steps in Creating a solution (console). After you create a solution, you can view the columns the solution uses on the solution's details page of the Amazon Personalize console, or with the DescribeSolution operation.

Topics

Configuring columns used when training (AWS CLI)

To exclude columns from training, provide the `excludedDatasetColumns` object in the `trainingDataConfig` as part of the solution configuration. For each key, provide the dataset type. For each value, provide the list of columns to exclude. The following code shows how to exclude columns from training when you create a solution with the AWS CLI.

```
aws personalize create-solution \
--name solution name \
--dataset-group-arn dataset group ARN \
--recipe-arn recipe ARN \
--solution-config "{\"trainingDataConfig\": {\"excludedDatasetColumns\":
 { \"datasetType\" : [ \"column1Name\", \"column2Name\"]}}}"
```

Configuring columns used when training (AWS SDKs)

To exclude columns from training, provide the `excludedDatasetColumns` object in the `trainingDataConfig` as part of the solution configuration. For each key, provide the dataset type. For each value, provide the list of columns to exclude. The following code shows how to exclude columns from training when you create a solution with the SDK for Python (Boto3).

```
import boto3

personalize = boto3.client('personalize')

create_solution_response = personalize.create_solution(
  name = 'solution name',
  recipeArn = 'recipe ARN',
  datasetGroupArn = 'dataset group ARN',
  solutionConfig = {
    "trainingDataConfig": {
      "excludedDatasetColumns": {
          "datasetType": ["COLUMN_A", "COLUMN_B"]
      }
    }
  }
)
solution_arn = create_solution_response['solutionArn']
print('solution_arn: ', solution_arn)
```

Optimizing a solution for an additional objective

> **Important**
>
> After you create a solution, you can't change its configuration. By default, all new solutions use automatic training. With automatic training, you incur training costs while your solution is active. To avoid unnecessary costs, make sure to delete the solution when you are finished. For information about training costs, see [Amazon Personalize pricing](https://aws.amazon.com/personalize/pricing/).

If you use the User-Personalization recipe or Personalized-Ranking recipe, you can optimize an Amazon Personalize solution for an objective in addition to maximum relevance, such as maximizing revenue, before training.

With item recommendation recipes, the primary objective of Amazon Personalize is to predict the most relevant items for your users based on historical and real-time item 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.

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.

You can use the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs. For information about using the Amazon Personalize console, see Creating a solution (console).

Topics

Guidelines and requirements

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 Schemas.

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 item interactions data to predict the most relevant items for your user.
- Low: Amazon Personalize places less emphasis on your objective. Relevance through item interactions data is more important.
- Medium: Amazon Personalize places equal emphasis on your objective and relevance through item interactions data.
- High: Amazon Personalize places more emphasis on your objective. Relevance through item 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 [Evaluating a solution version with metrics].

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 item interaction datas data. The `objectiveSensitivity` can be `OFF`, LOW, `MEDIUM` or `HIGH`. For more information, see [Balancing objective emphasis and relevance].

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 [Creating a solution (AWS CLI)]). Once you create a solution version, you can view the optimization performance with the solution version metrics. See [Measuring optimization performance].

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 item interaction datas data. The `objectiveSensitivity` can be `OFF`, `LOW`, `MEDIUM` or `HIGH`. For more information, see Balancing objective emphasis and relevance.

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)). Once you create a solution version, you can view the optimization performance with the solution version metrics. See Measuring optimization performance.

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 "";
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { CreateSolutionCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create the personalizeClient
const personalizeClient = new PersonalizeClient({ region: "REGION"});

// set the solution parameters.
export const createSolutionParam = {
  datasetGroupArn: 'DATASET_GROUP_ARN',              /* required */
  recipeArn: 'RECIPE_ARN',                           /* required */
  name: 'NAME',                                      /* required */
  solutionConfig: {
    optimizationObjective: {
      itemAttribute: "COLUMN_NAME",           /* specify the numerical column from
 the Items dataset related to your objective */
      objectiveSensitivity: "MEDIUM"          /* specify OFF, LOW, MEDIUM, or HIGH
 */
    }
  }
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateSolutionCommand(createSolutionParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
```

```
      console.log("Error", err);
    }
  };
  run();
```

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

> **Important**
>
> After you create a solution, you can't change its configuration. By default, all new solutions use automatic training. With automatic training, you incur training costs while your solution is active. To avoid unnecessary costs, make sure to delete the solution when you are finished. For information about training costs, see Amazon Personalize pricing.

You specify hyperparameters before training to optimize the trained model for your particular use case. 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 object that is passed to the CreateSolution 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.

```
{
  "name": "string",
  "recipeArn": "string",
  "eventType": "string",
  "solutionConfig": {
      "optimizationObjective": {
          "itemAttribute": "string",
```

```
            "objectiveSensitivity": "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 Choosing a recipe.

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. You must provide each type in your request, but if a recipe doesn't have a parameter of a type, you can leave it empty. For example, User-Personalization does not have a tunable hyperparameter of continuous type. So for the `continousHyperParameterRange`, you would pass an empty array.

The following code shows how to create a solution with HPO enabled using the SDK for Python (Boto3). The solution in the example uses the User-Personalization recipe recipe and has HPO set to `true`. The code provides a value for `hidden_dimension` and the `categoricalHyperParameterRanges` and `integerHyperParameterRanges`. The `continousHyperParameterRange` is empty and the `hpoResourceConfig` sets the `maxNumberOfTrainingJobs` and `maxParallelTrainingJobs`.

```
create_solution_response = personalize.create_solution(
    name = solutionName,
    datasetGroupArn = 'arn:aws:personalize:region:accountId:dataset-group/
datasetGroupName',
    recipeArn = 'arn:aws:personalize:::recipe/aws-user-personalization',
    performHPO = True,
    solutionConfig = {
        "algorithmHyperParameters": {
          "hidden_dimension": "55"
        },
        "hpoConfig": {
          "algorithmHyperParameterRanges": {
              "categoricalHyperParameterRanges": [
                  {
                      "name": "recency_mask",
                      "values": [ "true", "false"]
                  }
              ],
              "integerHyperParameterRanges": [
                  {
                      "name": "bptt",
                      "minValue": 2,
                      "maxValue": 22
                  }
              ],
              "continuousHyperParameterRanges": [

              ]
          },
          "hpoResourceConfig": {
              "maxNumberOfTrainingJobs": "4",
              "maxParallelTrainingJobs": "2"
          }
        }
    }
)
```

For more information about HPO, see Automatic model tuning.

Viewing hyperparameters

You can view the hyperparameters of the solution by calling the DescribeSolution operation. The following sample shows a `DescribeSolution` output. After creating a solution version (training a model), you can also view hyperparameters with the DescribeSolutionVersion operation.

```
{
  "solution": {
    "name": "hpo_coonfig_solution",
    "solutionArn": "arn:aws:personalize:region:accountId:solution/solutionName",
    "performHPO": true,
    "performAutoML": false,
    "recipeArn": "arn:aws:personalize:::recipe/aws-user-personalization",
    "datasetGroupArn": "arn:aws:personalize:region:accountId:dataset-group/
datasetGroupName",
    "eventType": "click",
    "solutionConfig": {
      "hpoConfig": {
        "hpoResourceConfig": {
          "maxNumberOfTrainingJobs": "4",
          "maxParallelTrainingJobs": "2"
        },
        "algorithmHyperParameterRanges": {
          "integerHyperParameterRanges": [
            {
              "name": "training.bptt",
              "minValue": 2,
              "maxValue": 22
            }
          ],
          "continuousHyperParameterRanges": [],
          "categoricalHyperParameterRanges": [
            {
              "name": "data.recency_mask",
              "values": [
                "true",
                "false"
              ]
            }
          ]
        }
      },
      "algorithmHyperParameters": {
```

```
            "hidden_dimension": "55"
          }
        },
        "status": "ACTIVE",
        "creationDateTime": "2022-07-08T12:12:48.565000-07:00",
        "lastUpdatedDateTime": "2022-07-08T12:12:48.565000-07:00"
    }
}
```

Choosing the item interaction data used for training

Important

After you create a solution, you can't change its configuration. By default, all new solutions use automatic training. With automatic training, you incur training costs while your solution is active. To avoid unnecessary costs, make sure to delete the solution when you are finished. For information about training costs, see Amazon Personalize pricing.

You can choose the events in an Item interactions dataset that Amazon Personalize uses when creating a solution version (training a model). Choosing item interaction 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 Item interactions datasets, see Schemas and Item interactions dataset.

You can choose item interaction data as follows:

- **Choose records based on type** – When you configure a solution, if your Item 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 Item 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 Item 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 item interaction data for training with equal weight regardless of type.
- **Choose records based on type and value** – When you configure a solution, if your Item 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](#).

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

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 Data format guidelines.
3. Upload your data to an Amazon Simple Storage Service (Amazon S3) bucket. For a code sample, see Uploading to an Amazon S3 bucket.
4. Import your data into Amazon Personalize with the CreateDatasetImportJob 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).
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.

```
# 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 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.

```
# 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 request.

```
# 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 and a solution version.

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 Evaluating a solution version with metrics.

Cloning a solution (console)

When you create a new solution, you can use the Amazon Personalize console to clone a solution. When you clone a solution, you can use the configuration of the existing solution as a starting point, such as the recipe and hyperparameters, and make any changes as necessary. This is useful if you want to make one change to a solution, but leave all other properties unchanged. For example, adding a new column of training data to your dataset. In this case, you would clone a solution, give the solution a name, change the columns used when training, and leave all other properties unchanged.

Cloning a solution

To clone a solution, you choose the existing solution, and choose the **Clone solution** option. Then give the new solution a name, and modify the relevant fields.

To clone a solution

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 **Custom resources** and choose **Solutions**.
4. Choose the solution that you want to clone.
5. Choose **Actions**, and choose **Clone solution**.
6. Give the new solution a name.
7. Make any changes to the solution details and advanced configuration. Amazon Personalize pre-populates these fields with values from the existing solution. For information about each field, see Creating and configuring a solution.

Creating a solution version

After you complete Creating and configuring a solution, you are ready to start training:

- If your solution uses automatic training, the solution creates solution versions for you at the training frequency you specify. By default, all new solutions use automatic training to create a new solution version every 7 days. You can still manually create solution versions. For more information, see Configuring automatic training.
- If you turn off auto training for your solution or you want to manually train, you can manually create a solution version. A *solution version* refers to a trained machine learning model. You can create a solution version using the console, AWS Command Line Interface (AWS CLI), or AWS SDKs. If your solution version has a status of CREATE_PENDING or CREATE_IN_PROGRESS, you can use the the section called "StopSolutionVersionCreation" operation to stop the solution version creation process. See Stopping the creation of a solution version.

Topics

Creating a solution version (console)

When you initially create your solution with the Amazon Personalize console, you also 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 . See Creating a campaign or Batch recommendations and user segments (custom resources).

If you 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, under **Custom resources**, 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 Evaluating a solution version with metrics.

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.

How you use an active solution version depends on how you get recommendations:

- For real-time recommendations, you deploy an ACTIVE solution version with an Amazon Personalize campaign. You use the campaign to get recommendations for your users. See Creating a campaign.
- For batch recommendations, you specify an ACTIVE solution version when you create a batch inference job or batch segment job. See Batch recommendations and user segments (custom resources).

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 Creating and configuring a solution.

```
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.

```
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 Evaluating a solution version with metrics. 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 operation to stop the solution version creation process. See Stopping the creation of a solution version.

How you use an active solution version depends on how you get recommendations:

- For real-time recommendations, you deploy an ACTIVE solution version with an Amazon Personalize campaign. You use the campaign to get recommendations for your users. See Creating a campaign.
- For batch recommendations, you specify an ACTIVE solution version when you create a batch inference job or batch segment job. See Batch recommendations and user segments (custom resources).

Creating a solution version (AWS SDKs)

When your solution is ACTIVE, use the following code to create a solution version. Specify the Amazon Resource Name (ARN) from Creating and configuring a solution. Use the DescribeSolutionVersion operation to retrieve the solution version's status.

SDK for Python (Boto3)

```
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

```
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());
                }
            }

            // 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()
```

```
                    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 "";
    }
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { CreateSolutionVersionCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the solution version parameters.
export const solutionVersionParam = {
  solutionArn: 'SOLUTION_ARN' /* required */
}

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateSolutionVersionCommand(solutionVersionParam));
    console.log("Success", response);
```

```
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

To check the current solution version status, call the DescribeSolutionVersion 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 Evaluating a solution version with metrics. 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 operation to stop the solution version creation process. See Stopping the creation of a solution version.

How you use an active solution version depends on how you get recommendations:

- For real-time recommendations, you deploy an ACTIVE solution version with an Amazon Personalize campaign. You use the campaign to get recommendations for your users. See Creating a campaign.
- For batch recommendations, you specify an ACTIVE solution version when you create a batch inference job or batch segment job. See Batch recommendations and user segments (custom resources).

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 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 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)

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 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 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.

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 models trained with the same data. 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 a training set, a testing set, and for PERSONALIZED_ACTIONS, a validation set. The split depends on the type of recipe you choose:

- For USER_SEGMENTATION recipes, the training set consists of 80% of each user's interactions data and the testing set consists of 20% of each user's interactions data.
- For all other recipe types, the training set consists of 90% of your users and their interactions data. The testing set consists of the remaining 10% of users and their interactions data.

Amazon Personalize then creates the solution version using the training set. After training completes, Amazon Personalize gives the new 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 recipe, which recommends the top K most popular items.

Topics

- Retrieving solution version metrics
- Metric definitions
- Example

- Additional resources

Retrieving solution version metrics

After you create a solution version, you can use metrics to evaluate its performance. You can retrieve metrics for a solution version with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), and AWS SDKs.

Topics

Retrieving solution version metrics (console)

To view recommender metrics in the console, you navigate to the details page for your solution version.

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 Custom dataset group.
3. From the navigation pane, choose **Custom resources** and then choose **Solutions and recipes**.
4. Choose your solution.
5. In **Solution versions**, choose your solution version to view its details page. The metrics are listed on the **Solution version metrics** tab in the bottom pane. For definitions of metrics, see Metric definitions.

   Now that you have evaluated your solution version, you can create a campaign by deploying the solution version with the best metrics for your use case. For more information about deploying a solution, see Creating a campaign.

Retrieving solution version metrics (AWS CLI)

You retrieve the metrics for a specific solution version by calling the GetSolutionMetrics operation. The following code shows how to retrieve metrics with the AWS CLI.

```
personalize get-solution-metrics --solution-version-arn solution version ARN
```

The following is an example the output from a solution version created using the User-Personalization recipe with an additional optimization objective.

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

For explanations of each metric, see Metric definitions. Now that you have evaluated your solution version, you can create a campaign by deploying the solution version with the best metrics for your use case. For more information about deploying a solution, see Creating a campaign.

Retrieving solution version metrics (AWS SDKs)

You retrieve the metrics for a specific solution version by calling the GetSolutionMetrics operation. Use the following code to retrieve metrics.

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

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

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

SDK for Java 2.x

```
public static void getSolutionVersionMetrics(PersonalizeClient personalizeClient,
 String solutionVersionArn) {

    try {
        GetSolutionMetricsRequest request = GetSolutionMetricsRequest.builder()
                .solutionVersionArn(solutionVersionArn)
                .build();
        Map<String, Double> metrics =
 personalizeClient.getSolutionMetrics(request).metrics();
        metrics.forEach((key, value) -> System.out.println(key + " " + value));
    } catch (PersonalizeException e ) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

The following is an example the output from a solution version created using the User-Personalization 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
    }
}
```


For explanations of each metric, see Metric definitions. Now that you have evaluated your solution version, you can create a campaign by deploying the solution version with the best metrics for your use case. For more information about deploying a solution, see Creating a campaign.

Metric definitions

The metrics Amazon Personalize generates for solution versions are described below using the following terms:

- *Relevant recommendation* is a recommendation for an item that the user actually interacted with. These items are from the newest 10% of each user's interactions data from the testing set.
- *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.

For each metric, higher numbers (closer to 1) are better. To dive deeper, see the resources listed in Additional resources.

coverage

The value for *coverage* tells you the proportion of unique items (for item recommendations), actions (for action recommendations), or users (for user segment recommendations), that Amazon Personalize might recommend out of the total number of unique records in your datasets.

A higher coverage score means Amazon Personalize recommends more of your catalog, rather than the records same repeatedly. Recipes that feature item exploration, such as User-Personalization, have higher coverage than those that don't, such as Similar-Items.

mean reciprocal rank at 25

This metric tells you about a model's ability to generate a relevant item recommendations at the top ranked position.

You might choose a model with a high *mean reciprocal rank at 25* if you are generating item search results for a user, and don't expect the user to choose an item lower on the list. For example, users frequently choose the first cooking recipe in search results. Amazon Personalize doesn't generate this metric for PERSONALIZED_ACTIONS or USER_SEGMENTATION recipes.

Amazon Personalize calculates this metric using the average reciprocal rank score for requests for recommendations. Each reciprocal rank score is calculated as follows: `1 / the rank of the highest item interacted with by the user`, where the total possible rankings is 25. Other lower ranked items the user interacts with are ignored. If the user chose the first item, the score is 1. If they don't choose any items, the score is 0.

For example, you might show three different users 25 recommendations each:

- If User 1 clicks the item at rank *4* and the item at rank *10*, their reciprocal rank score is *1/4*.
- If User 2 clicks an item at rank *2*, an item at rank *4*, and an item at rank *12*, their reciprocal rank score is 1/2.
- If User 3 clicks on a single item at rank *6*, their reciprocal rank score is 1/6.

The mean reciprocal rank over all requests for recommendations (in this case 3) is calculated as `(1/4 + 1/2 + 1/6) / 3 = .3056`.

normalized discounted cumulative gain (NDCG) at K (5/10/25)

This metric tells you about how well your model ranks item or action recommendations, where K is a sample size of 5, 10, or 25 recommendations. This metric is useful if you are most interested in the ranking of recommendations beyond just the highest ranked item or action(for this, see `mean reciprocal rank at 25`). For example, the score for `NDCG at 10` would be useful if you have an application that shows up to 10 movies in a carousel at a time.

Amazon Personalize calculates the NDCG by assigning weight to recommendations based on their ranking position for each user in the testing set. Each recommendation is discounted (given a lower weight) by a factor dependent on its position. The final metric is the average of all users in the testing set. 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 + position)`, where the top of the list is position `1`.

precision at K

This metric 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 for each user in the testing set, divided by K, where K is 5, 10, or 25. The final metric is the average across all users in the testing set.

For example, if you recommend 10 items to a user, and the user interacts with 3 of them, the precision at K is 3 correctly predicted items divided by the total 10 recommended items: `3 / 10 = .30`.

This metric rewards precise recommendation of relevant items. The closer the score is to one, the more precise the model.

precision

If you train a solution version with the Next-Best-Action recipe, Amazon Personalize generates a `precision` metric instead of `precision at K`. This metric tells you how good your model is at predicting actions users will actually take.

To calculate `precision`, for each action in your dataset, Amazon Personalize divides the number of users that were correctly predicted to take the action by the total number of times the action was recommended. Amazon Personalize then calculates the average for all actions in your dataset.

For example, if an action was recommended to 100 users, and 60 users took the action and 40 users who didn't, the `precision` for the action is: `60 / 100 = .60`. Amazon Personalize then applies this calculation for all actions and returns the average.

This metric rewards precise recommendation of relevant actions. The closer the score is to one, the more precise the model.

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.

trend prediction accuracy

If you trained the solution version with the Trending-Now recipe, the rate of increase in popularity of items recommended by the model. The higher the trend prediction accuracy (the closer to 1), the better the model is at correctly identifying trending items.

To calculate popularity acceleration, Amazon Personalize divides the rate of increase in popularity across all recommended items by the total popularity increase of the top 25 trending items. These items come from the actual interactions in the testing set.

Depending on your data distribution and what you choose for Trend discovery frequency, the value for trend prediction accuracy can be 0.0.

hit (hit at K)

If you trained the solution version with a USER_SEGMENTATION recipe, the average number of users in the predicted top relevant K results that match the actual users. Actual users are the users who actually interacted with the items in the test set. K is the top 1% of the most relevant users. The higher the value the more accurate the predictions.

recall (recall at K)

If you trained the solution version with a USER_SEGMENTATION recipe, the average percentage of predicted users in the predicted top relevant K results that match the actual users. Actual users are the users who actually interacted with the items in the test set. K is the top 1% of the most relevant users. The higher the value, the more accurate the predictions.

recall

If you train a solution version with the Next-Best-Action recipe, this metric tells you how good your solution version is at discovering actions that users will interact with.

To calculate `recall`, for each action in your dataset, Amazon Personalize divides the number of users that were correctly predicted to take the action by the total number of users that actually take the action in the testing set. Amazon Personalize then calculates the average for all actions in your dataset.

For example, if 100 users take an action in the testing set, and Amazon Personalize predicted 50 of these users would take the action, the `recall` for the action is: `50 / 100 = .50`. Amazon Personalize then applies this calculation for all actions and returns the average.

Area under the curve (AUC)

If you trained the solution version with a PERSONALIZED_ACTIONS recipe, the area under the Receiver Operating Characteristic curve for your solution version. This metric tells you how well the solution version performs at correctly identifying actions that users will take.

The Receiver Operating Characteristic curve plots the performance of the solution version. It plots the true positive (actions correctly predicted as relevant) and false positive (actions incorrectly predicted as relevant) rates at different threshold values. The Area under the curve (AUC) is a score that summarizes the performance of the solution version based on its curve.

The AUC of a solution version can between 0 and 1. The closer to 1, the better the model is at predicting relevant actions for your users.

Example

The following is a simple example for a solution version that 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

Additional resources

For information on evaluating a solution version with A/B testing, see Using A/B testing to measure the efficacy of recommendations generated by Amazon Personalize. To dive deeper in different types of metrics for recommender systems, see the following external resources:

- MRR vs MAP vs NDCG: Rank-Aware Evaluation Metrics And When To Use Them
- Discounted Cumulative Gain: the ranking metrics you should know about
- Recall and Precision at k for Recommender Systems
- Ranking Evaluation Metrics for Recommender Systems
- Receiver operating characteristic

Creating a campaign

For real-time recommendations with custom resources, after you complete Creating a solution version, you are ready to deploy your solution version with a campaign.

A *campaign* deploys a solution version (trained model) with a provisioned transaction capacity for generating real-time recommendations. After you create a campaign, you use the GetRecommendations or GetPersonalizedRanking API operations to get recommendations. If you are getting batch recommendations, you don't need to create a campaign. For more information see Batch recommendations and user segments (custom resources).

When you create a campaign, you can configure the following:

- You can configure the campaign to automatically update to use your solution's latest solution version. For more information see Enabling automatic campaign updates.
- You can enable item metadata in recommendations. For more information, see Metadata with recommendations.
- You can specify the minimum provisioned transactions per second for the campaign. This is the baseline transaction throughput for the campaign provisioned by Amazon Personalize. It sets the minimum billing charge for the campaign while it is active. For more information, see Minimum provisioned transactions per second and auto-scaling.

You can create a campaign with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs. If you want to change an existing campaign's settings, such as enabling metadata in recommendations, you must update your campaign. For more information see Updating a campaign.

You incur campaign costs while the campaign is active. To avoid unnecessary costs, make sure to delete the campaign when you are finished. For information about campaign costs, see Amazon Personalize pricing.

Topics

Enabling automatic campaign updates

When you create a campaign, you can enable automatic campaign updates. With automatic updates, the campaign automatically updates to deploy the latest automatically or manually trained solution version of your solution. This makes it easier for you to keep your campaign current.

For example, if your solution uses automatic training to create a new solution version every seven days, your campaign would automatically update to use the latest solution version for every weekly training. If you don't use automatic campaign updates, you must manually update the campaign to deploy the latest trained model.

- To enable automatic campaign updates when you create a campaign with the Amazon Personalize console, choose **Automatically update to use your solution's latest solution version** in the **Campaign details**. You can find the timestamp for the latest update on the campaign details page.

  For more information, see Creating a campaign (console).
- To enable automatic campaign updates when you use the CreateCampaign API operation, for the `SolutionVersionArn` parameter, specify the Amazon Resource Name (ARN)

of your solution in `SolutionArn/$LATEST` format. In the `campaignConfig`, set `enableMetadataInInferenceResponse` to `true`.

To get the timestamp of the latest campaign update, you can use the DescribeCampaign API operation and check `latestCampaignUpdate` details in the response.

For code samples that show you how to enable automatic updates, see Creating a campaign (AWS CLI) or Creating a campaign (AWS SDKs).

Metadata with recommendations

Important

When you enable metadata in recommendations, you incur additional costs. For more information see Amazon Personalize pricing.

When you create a campaign, you can enable metadata in recommendations. If enabled, when you get recommendations you can choose the columns from your Items dataset to include in results. Amazon Personalize returns this data for each item in the recommendation response.

You might use metadata to enrich recommendations in your user interface, such as adding the genres for movies to carousels. Or you might use it to visually assess recommendation quality. If you use generative AI in your app, you can plug the metadata into AI prompts to generate more relevant content. For more information about using Amazon Personalize with generative AI, see Amazon Personalize and generative AI.

- To enable metadata when you create a campaign with Amazon Personalize console, choose **Return items metadata in recommendation results** in the **Campaign details**.

  For more information, see Creating a campaign (console).

- To enable metadata with the CreateCampaign API operation, in the `campaignConfig` set `enableMetadataInInferenceResponse` to `true`.

  For more information, see Creating a campaign (AWS CLI) or Creating a campaign (AWS SDKs).

To add metadata to recommendations, you must have an Items dataset with a column of metadata. You don't have to use the metadata in training. For information about creating a dataset, see

Creating a dataset and a schema. For information managing and updating data, see Managing data.

Minimum provisioned transactions per second and auto-scaling

> **Important**
>
> A high `minProvisionedTPS` will increase your cost. We recommend starting with 1 for `minProvisionedTPS` (the default). Track your usage using Amazon CloudWatch metrics, and increase the `minProvisionedTPS` as necessary.

When you create an Amazon Personalize campaign, you can specify the minimum provisioned transactions per second (`minProvisionedTPS`) for the campaign. This is the baseline transaction throughput for the campaign provisioned by Amazon Personalize. It sets the minimum billing charge for the campaign while it is active. A transaction is a single `GetRecommendations` or `GetPersonalizedRanking` request. The default `minProvisionedTPS` is 1.

If your TPS increases beyond the `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. When your traffic reduces, capacity returns to the `minProvisionedTPS`.

You are charged for the minimum provisioned TPS or, if your requests exceed the `minProvisionedTPS`, the actual TPS. The actual TPS is the total number of recommendation requests you make. We recommend starting with a low `minProvisionedTPS`, track your usage using Amazon CloudWatch metrics, and then increase the `minProvisionedTPS` as necessary.

For more information about campaign costs, see Amazon Personalize pricing.

Creating a campaign (console)

> **Important**
>
> You incur campaign costs while the campaign is active. To avoid unnecessary costs, make sure to delete the campaign when you are finished. For information about campaign costs, see Amazon Personalize pricing.

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 that you want to deploy.
3. In the navigation pane, under **Custom resources**, 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.
   - **Automatically update to use your solution's latest solution version –** Choose this option to have the campaign automatically use the latest active solution version. If you don't choose this, you must manually update the campaign each time you want to deploy a new solution version. For more information, see Enabling automatic campaign updates.
   - **Solution version ID –** If you don't use automatic campaign updates to use the latest solution version, choose the ID of the solution version that you want to deploy.
   - **Minimum provisioned transactions per second (called minProvisionedTPS in APIs) –** Set the minimum provisioned transactions per second that Amazon Personalize supports. A high value will increase your charges. We recommend that you start with 1 (the default). Track your usage by using Amazon CloudWatch metrics, and increase the `minProvisionedTPS` as necessary. For more information, see Minimum provisioned transactions per second and auto-scaling.
   - **Return items metadata in recommendation results –** Choose this option if you want the option to include metadata with recommendation results. If enabled, you can specify the columns from your Items dataset when you get recommendations. For more information, see Metadata with recommendations.
6. If you used the User-Personalization recipe, in **Campaign configuration**, you can optionally enter values for the **Exploration weight** and **Exploration item age cut off**. For more information, see User-Personalization.

7. For **Tags**, optionally add any tags. For more information about tagging Amazon Personalize resources, see Tagging Amazon Personalize resources.
8. Choose **Create campaign**.
9. 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: Getting recommendations.

   The campaign is ready when its status is ACTIVE. If you retrain your solution version, or if you want to change your campaign settings, you must update your campaign. For more information, see Updating a campaign.

Creating a campaign (AWS CLI)

> **Important**
>
> You incur campaign costs while the campaign is active. To avoid unnecessary costs, make sure to delete the campaign when you are finished. For information about campaign costs, see Amazon Personalize pricing.

After your solution version is Active, you are ready to deploy it with an Amazon Personalize campaign. To create a campaign with the AWS CLI, you use the `create-campaign` command.

The following code sample shows you how to create a campaign. It deploys the latest solution version of a solution that uses the User-Personalization recipe. The campaign it creates automatically updates to use future solution versions. The code uses the following configuration:

- It configures the campaign to automatically update to use the latest solution version for your solution: The `solution-version-arn` is in *`solution ARN`*`/$LATEST` format, and `syncWithLatestSolutionVersion` is `True`. To use the code, replace `solution ARN` with the Amazon Resource Name (ARN) of your solution.

  To disable automatic `syncWithLatestSolutionVersion`, specify only the solution version ARN (without `/$LATEST`), and set `syncWithLatestSolutionVersion` to `False`.
- It sets the `enableMetadataWithRecommendations` option to `True`. This enables a recommendation request option to include item metadata from an Items dataset with recommendation results. To disable this option, set it to `False`. For more information, see Metadata with recommendations.

- It sets `min-provisioned-tps` to 1 (the default). We recommend starting with 1 for `minProvisionedTPS` (the default). Track your usage by using Amazon CloudWatch metrics, and increase the `minProvisionedTPS` as necessary. For more information, see Minimum provisioned transactions per second and auto-scaling.

For a complete list of all parameters, see CreateCampaign.

```
aws personalize create-campaign \
--name campaign name \
--solution-version-arn solution ARN\/$LATEST \
--min-provisioned-tps 1 \
--campaign-config "{"\"syncWithLatestSolutionVersion"\": "true",
 "\"enableMetadataWithRecommendations"\": "true"}"
```

The campaign is ready when its status is ACTIVE. To get the current status, call DescribeCampaign and check that the `status` field is `ACTIVE`.

If you retrain your solution version and your campaign doesn't automatically update to use the latest solution version, or if your want to change your campaign settings, you must update your campaign. For more information, see Updating a campaign.

Amazon Personalize provides you with operations for managing campaigns such as ListCampaigns to list the campaigns that you have created. You can delete a campaign by calling DeleteCampaign. If you delete a campaign, the solution versions that are part of the campaign are not deleted.

After you have created your campaign, you can use it to make recommendations. For more information, see Step 4: Getting recommendations.

Creating a campaign (AWS SDKs)

> **Important**
>
> You incur campaign costs while the campaign is active. To avoid unnecessary costs, make sure to delete the campaign when you are finished. For information about campaign costs, see Amazon Personalize pricing.

After your solution version is Active, you are ready to deploy it with an Amazon Personalize campaign. To create a campaign with AWS SDKs, you use the CreateCampaign API operation.

The following code sample shows you how to create a campaign. The code deploys the latest solution version of a solution that uses the User-Personalization recipe. The campaign it creates automatically updates to use future solution versions. The code uses the following configuration:

- It configures the campaign to automatically update to use the latest solution version for your solution: The `solutionVersionArn` is in *`solution ARN`*`/$LATEST` format, and `syncWithLatestSolutionVersion` is `True`. To use the code, replace `solution ARN` with the Amazon Resource Name (ARN) of your solution version.

  To disable automatic `syncWithLatestSolutionVersion`, specify only the solution version ARN (without `/$LATEST`), and set `syncWithLatestSolutionVersion` to `False`.
- It sets the `enableMetadataWithRecommendations` option to `True`. This enables a recommendation request option to include item metadata from an Items dataset with recommendation results. To disable this option, set it to `False`. For more information, see Metadata with recommendations.
- It sets `minProvisionedTPS` to 1 (the default). We recommend that you start with 1 for `minProvisionedTPS` (the default). Track your usage by using Amazon CloudWatch metrics, and increase the `minProvisionedTPS` as necessary. For more information, see Minimum provisioned transactions per second and auto-scaling.

For a complete list of all parameters, see CreateCampaign.

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_campaign(
    name = 'campaign name',
    solutionVersionArn = 'solution ARN/$LATEST',
    minProvisionedTPS = 1,
    campaignConfig = {"syncWithLatestSolutionVersion": True,
 "enableMetadataWithRecommendations": True}
)

arn = response['campaignArn']

description = personalize.describe_campaign(campaignArn = arn)['campaign']
print('Name: ' + description['name'])
print('ARN: ' + description['campaignArn'])
```

```
print('Status: ' + description['status'])
```

The campaign is ready when its status is ACTIVE. To get the current status, call DescribeCampaign, and check that the `status` field is `ACTIVE`.

If you manually retrain your solution version, or if you want to change your campaign settings, you must update your campaign. For more information, see Updating a campaign.

Amazon Personalize provides you with operations for managing campaigns such as ListCampaigns to list the campaigns that you have created. You can delete a campaign by calling DeleteCampaign. 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 Step 4: Getting recommendations.

Updating a campaign

To deploy a retrained solution version with an existing campaign or to change your campaign's Minimum provisioned TPS or campaign configuration, such as enabling metadata in recommendations, you must manually update the campaign.

With User-Personalization or Next-Best-Action, Amazon Personalize automatically updates your latest solution version (trained with `trainingMode` set to `FULL`) every two hours to include new items or actions 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, see Automatic updates.

You manually update a campaign with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs.

Topics

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](), 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]() 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.

SDK for Python (Boto3)

Use the following `update_campaign` method to deploy a new solution version. Replace `campaign arn` 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.

```
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);
    }
}
```

Step 4: Getting recommendations

Depending on your resources, you can get recommendations in real time or with a batch workflow.

- With custom resources, you can get real-time recommendations or batch recommendations. For real-time recommendations, you must create a custom campaign before you get recommendations. For batch recommendations, you don't need to create a campaign.
- With recommenders in a Domain dataset group, you can get only real-time recommendations.

The following topics explain how and when to use each recommendation type.

Topics

Recommendation scores

With custom solutions created with the User-Personalization, Personalized-Ranking, and PERSONALIZED_ACTIONS recipes, Amazon Personalize includes a score for each item in recommendations. These scores represent the relative certainty that Amazon Personalize has about which item or action the user will select next. Higher scores represent greater certainty.

- For information about scores for User-Personalization, see How User-Personalization recommendation scoring works (custom resources).
- For information about scores for PERSONALIZED_ACTIONS recipes, see How action recommendation scoring works.
- For information on scores for Personalized-Ranking recommendations, see How personalized ranking scoring works.

For batch inference jobs, item scores are calculated just as described in How User-Personalization recommendation scoring works (custom resources) and How personalized ranking scoring works. You can view scores in the batch inference job's output JSON file.

Getting real-time recommendations

Real-time recommendations are recommendations you request and show your user as they use your application. You can get real-time recommendations from Amazon Personalize with a recommender (for Domain dataset groups) or a custom campaign.

- For domain recommenders, you can get real-time recommendations with the the section called "GetRecommendations" operation. Or you can test your recommender with the Amazon Personalize console.
- For custom resources, 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", GetActionRecommendations, or the section called "GetPersonalizedRanking" API operations. Or you can test your campaign with the Amazon Personalize console.

If you use domain use cases or recipes that provide real-time personalization, such as the *Top picks for you* use case or the *User-Personalization* recipe, Amazon Personalize updates recommendations based on your user's most recent activity as you record their interactions with your catalog. For more information on recording real-time events and personalization, see Recording events.

If you configured your campaign to return metadata for recommended items, you can specify the columns to include in your GetRecommendations or GetPersonalizedRanking API operations. Or you can specify the columns when you test the campaign with the Amazon Personalize console.

For some use cases and recipes, you can specify a promotion in your request. A *promotion* defines additional business rules that apply to a configurable subset of recommended items. For more information see Promoting items in recommendations.

Topics

Getting item recommendations

You can get item recommendations from a Amazon Personalize recommender or custom campaign with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs.

> **Note**
>
> If you used a PERSONALIZED_RANKING custom recipe, see Getting a personalized ranking (custom resources).

Topics

How User-Personalization recommendation scoring works (custom resources)

With the User-Personalization recipe, Amazon Personalize generates scores for items based on on a user's interaction data and metadata. These scores represent the relative certainty that Amazon

Personalize has in whether the user will interact with the item next. Higher scores represent greater certainty.

Amazon Personalize scores all of the items in your catalog 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 domain recommenders or the Similar-Items, SIMS or Popularity-Count recipes. For information on scores for Personalized-Ranking recommendations, see How personalized ranking scoring works.

Getting item recommendations (console)

To get recommendations with the Amazon Personalize console, you provide the request information on the details page of either a recommender (Domain dataset group) or a custom campaign.

To get recommendations

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 or recommender you are using.

3. In the navigation pane, choose **Campaigns** or **Recommenders**.
4. Choose the target campaign or recommender.
5. For a campaigns, under **Test campaign results**, enter your recommendation request details based on the recipe you used. For a recommenders choose **Test recommender** and enter your recommendation request details based on your use case.

   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 as if it is their `userId`. For more information about recording events for anonymous users, see Recording events for anonymous users.
6. Optionally choose a filter. For more information, see Filtering recommendations and user segments.
7. If you use contextual metadata, provide data for each context. For each context, for the **Key** enter the metadata field. For the **Value** enter the context data. For more information, see Increasing recommendation relevance with contextual metadata.
8. If you enabled metadata in recommendations for your campaign or recommender, for **Items dataset columns**, choose the metadata columns that you want to include in recommendation results. For information about enabling metadata for a campaign, see Metadata with recommendations. For information about enabling metadata for a recommender, see Enabling metadata in recommendations.
9. If you want to promote a subset of items, optionally complete the **Promotion** fields. For more information see Promoting items in recommendations.
10. Choose **Get recommendations**. A table containing the user’s top 25 recommended items displays.

Getting item recommendations (AWS CLI)

The following code samples show different variations of how to get item recommendations with the AWS CLI.

Topics

- Getting item recommendations
- Including item metadata with recommendations

Getting item recommendations

Use the following code to get recommendations from a campaign. To get recommendations from a recommender, replace the `campaign-arn` parameter with the `recommender-arn`.

Specify the ID of the user you want to get recommendations for, and the Amazon Resource Name (ARN) of your campaign or recommender. 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 as if it is their `userId`. For more information about recording events for anonymous users, see Recording events for anonymous users.

```
aws personalize-runtime get-recommendations \
--campaign-arn campaign arn \
--user-id User ID \
--num-results 10
```

Including item metadata with recommendations

If you enabled metadata in recommendations for your campaign or recommender, you can specify the Items dataset metadata columns to include in the response. For information about enabling metadata for a campaign, see Metadata with recommendations. For information about enabling metadata for a recommender, see Enabling metadata in recommendations.

The following code sample shows how to specify the metadata columns as part of your request for recommendations.

```
aws personalize-runtime get-recommendations \
--campaign-arn campaign arn \
--user-id User ID \
--num-results 10 \
--metadata-columns "{\"ITEMS\": ["\"columnNameA"\",\"columnNameB"\"]}"
```

Getting item recommendations (AWS SDKs)

The following code samples show different variations of how to get item recommendations with the AWS SDKs.

Topics

Getting item recommendations

The following code shows how to get Amazon Personalize recommendations for a user from a campaign. To get recommendations from a recommender, replace the `campaignArn` parameter with the `recommenderArn`.

Specify the ID of the user you want to get recommendations for, and the Amazon Resource Name (ARN) of your campaign or recommender. 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 enabled metadata in recommendations for your campaign or recommender, you can specify the Items dataset metadata columns to include in the response. For a code sample, see Including item metadata with recommendations. For information about enabling metadata, see Metadata with recommendations.

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 as if it is their `userId`. For more information about recording events for anonymous users, see Recording events for anonymous users.

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);
        }
    }
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from
  "@aws-sdk/client-personalize-runtime";

import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
```

```
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the recommendation request parameters.
export const getRecommendationsParam = {
  campaignArn: 'CAMPAIGN_ARN', /* required */
  userId: 'USER_ID',      /* required */
  numResults: 15    /* optional */
}

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
 GetRecommendationsCommand(getRecommendationsParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Including item metadata with recommendations

If you enabled metadata in recommendations for your campaign or recommender, you can specify the Items dataset metadata columns to include in the response. For information about enabling metadata for a campaign, see Metadata with recommendations. For information about enabling metadata for a recommender, see Enabling metadata in recommendations.

The following code sample shows how to specify the metadata columns as part of your request for recommendations.

```
import boto3

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

response = personalizeRt.get_recommendations(
  campaignArn = 'Campaign ARN',
  userId = 'User ID',
  numResults = 10
```

```
  metadataColumns = {
    "ITEMS": ['columnNameA','columnNameB']
  }
)

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

Promoting items in recommendations

With all domain use cases and some custom recipes, you can specify a promotion when you get recommendations. A *promotion* defines additional business rules that apply to a configurable subset of recommended items. For example, you might have a streaming app and want to promote your own shows and movies but also recommend relevant titles. You could use a promotion to specify that a certain percentage of recommended items must come from the category *in-house*. The remaining recommended items would continue to be relevant recommendations based on your recipe and any request filters.

To apply a promotion, you specify the following in your recommendation request:

- The percentage of recommended items to apply the promotion filter to.
- A filter that specifies the promotion criteria. For more information, see Promotion filters.

In the recommendation response, promoted items are positioned randomly relative to other recommended items, but in sorted order relative to other promoted items. Depending on your recipe, recommended items that aren't part of a promotion are sorted by relevance to the user, popularity, or similarity. If there aren't enough items that meet the promotion criteria, the result will contain as many promoted items as possible.

You can apply a promotion to recommendations with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs.

Topics

- Use cases and recipes that support promotions
- Promotion filters
- Promoting items (console)
- Promoting items (AWS CLI)

- Promoting items (AWS SDKs)

Use cases and recipes that support promotions

All use cases support promotions. The following custom recipes support promotions:

- USER_PERSONALIZATION recipes
- RELATED_ITEMS recipes
- POPULAR_ITEMS recipes

Promotion filters

When you apply a promotion to a recommendation request, you choose a filter that specifies the promotion criteria. You can use an existing filter or create a new one. You create and manage filters for promotions as you would other filters in Amazon Personalize. For information about creating and managing filters, see Filtering results.

The only difference between a promotion filter and a filter that you choose outside the promotion (the *request filter*) is how Amazon Personalize applies them. A promotion filter applies to only promoted items, while a request filter applies to only the remaining recommended items. If you specify a request filter and promotion filter, and want to apply both filters to promoted items, your promotion filter's expression must include both expressions. The way you combine two expressions depends on the datasets you use. For more information on filter expressions, their rules, and how to create them, see Filter expressions.

Filter expression examples

The following expression includes only items from the category "in-house". You might use this expression if you want to promote your own content in your recommendations.

```
INCLUDE ItemID WHERE Items.OWNER IN ("in-house")
```

The following expression includes only items created earlier than a timestamp you specify. You might use this expression to promote items created recently.

```
INCLUDE ItemID WHERE Items.CREATION_TIMESTAMP < $DATE
```

The following expression shows how you might apply a request filter to promoted items. It includes only available clothing items as promoted items. In this scenario, the `Items.AVAILABLE IN`

`("True")` would also be used in the request filter expression, so that all recommendations are for items that are available.

```
INCLUDE ItemID WHERE Items.CATEGORY IN ("clothing") AND Items.AVAILABLE IN ("True")
```

For a more complete list of filter examples, see Filter expression examples.

Promoting items (console)

To promote certain items in recommendations with the Amazon Personalize console, create a filter, and then provide the promotion details in the recommendation request. For information on other fields, see Getting item recommendations (console).

To promote items in recommendations

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 or recommender you are using.
3. If you haven't already, create a filter that specifies the promotion criteria. You create filters for promotions the same way that you create request filters. For information on creating and managing filters, see Filtering results.
4. In the navigation pane, choose **Recommenders** or **Campaigns**.
5. Choose the target campaign or recommender.
6. For campaigns, under **Test campaign results**, enter your recommendation request details based on the recipe you used. For recommenders, choose **Test recommender** and enter your recommendation request details.
7. Optionally choose a filter for the request. This filter applies to only non-promoted items. For information on creating and managing filters, see Filtering results.
8. If you use contextual metadata, provide data for each context. For each context, for the **Key** enter the metadata field. For the **Value**, enter the context data. For more information, see Increasing recommendation relevance with contextual metadata.
9. For **Promotion** specify the following:

    - **Percent promoted items**: Enter the percentage of recommended items to apply the promotion to.
    - **Filter**: Choose a filter that specifies the promotion criteria. This filter applies to the promoted items instead of any request filter that you may have specified in step 7.

   - **Filter parameter**: If your promotion uses 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.

10. Choose **Get recommendations**. A table containing the user's top 25 recommended items displays. The **Promoted item** column indicates whether the item was included because of your promotion. Promoted items are positioned randomly relative to other recommended items, but in sorted order relative to other promoted items. Depending on your use case or recipe, recommended items that aren't part of a promotion are sorted by relevance to the user, popularity, or similarity. If there aren't enough items that meet the promotion criteria, the result will contain as many promoted items as possible.

Promoting items (AWS CLI)

The following code shows how to promote items in recommendations with the AWS CLI and a custom campaign. To promote items with a recommender, replace the `campaign-arn` parameter with a `recommender-arn` and specify the Amazon Resource Name (ARN) for the recommender. For the promotion fields, specify the following:

- name: Give the promotion a name. The recommendation response uses the name to identify promoted items.
- percent-promoted-items: The percentage of recommended items to apply the promotion to. In this example, 50% of items will be promoted items.
- filterArn: Specify the Amazon Resource Name (ARN) of the filter that defines the promotion criteria. For more information, see Promotion filters.
- parameter names and values: If your filter expression has any parameters, provide the parameter names (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. When you use the AWS CLI, for each value you must use the / character to escape both quotes and the / character. The following code example shows how to format the values.

The code shows how to use both a request filter and a promotion filter. A promotion filter applies to only promoted items, while a request filter applies to only the remaining recommended items. For more information, see Promotion filters.

For information about additional fields, see Getting item recommendations (AWS SDKs) and Getting a personalized ranking using contextual metadata.

```
aws personalize-runtime get-recommendations \
--campaign-arn CampaignArn \
--user-id 1 \
--num-results 10  \
--filter-arn RequestFilterArn \
--filter-values '{
    "RequestFilterParameterName": "\"value\"",
    "RequestFilterParameterName": "\"value1\",\"value2\",\"value3\""
  }' \
--promotions "[{
  \"name\": \"promotionName\",
  \"percentPromotedItems\": 50,
  \"filterArn\": \"PromotionFilterARN\",
  \"filterValues\": {\"PromotionParameterName\":\"\\\"value1, value2\\\"\"}
}]"
```

A list of recommended items displays. Promoted items are positioned randomly relative to other recommended items, but in sorted order relative to other promoted items. Depending on your recipe, recommended items that aren't part of a promotion are sorted by relevance to the user, popularity, or similarity. If there aren't enough items that meet the promotion criteria, the result will contain as many promoted items as possible.

```
{
  "itemList": [
      {
          "itemId1": "123",
          "score": .0117211,
          "promotionName": "promotionName"
      },
      {
         "itemId2": "456",
         "score": .0077976
      },
      {
         "itemId3": "789",
         "score": .0067171
      },
      .....
 ]
```

Promoting items (AWS SDKs)

The following code shows how to promote items in recommendations with the SDK for Python (Boto3) and the SDK for Java 2.x and a custom campaign. To promote items with a recommender, replace the `campaignArn` parameter with `recommenderArn` and specify the Amazon Resource Name (ARN) for the recommender. For the promotion fields, specify the following:

- name: Specify the name of the promotion. The recommendation response includes the name to identify promoted items.
- percentPromotedItems: The percentage of recommended items to apply the promotion to.
- promotionFilterARN: The Amazon Resource Name (ARN) of the filter that defines the promotion criteria. For more information, see Promotion filters.
- Any parameter names and values: If your filter expression has any parameters, 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"\"`.

The following code shows how to use both a request filter and a promotion filter. A promotion filter applies to only promoted items, while a request filter applies to only the remaining recommended items. For more information, see Promotion filters.

For information about additional fields, see Getting item recommendations (AWS SDKs) and Getting a personalized ranking using contextual metadata.

SDK for Python (Boto3)

```
import boto3

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

response = personalizeRt.get_recommendations(
  campaignArn = "CampaignARN",
  userId = '1',
  numResults = 10,
  filterArn = 'RequestFilterARN',
  filterValues = {
      "RequestFilterParameterName": "\"value1\"",
      "RequestFilterParameterName": "\"value1\",\"value2\",\"value3\""
```

```
        ....
    },
    promotions = [{
      "name" : "promotionName",
      "percentPromotedItems" : 50,
      "filterArn": "promotionFilterARN",
      "filterValues": {
        "PromotionParameterName": "\"Value1\",\"Value2\""
        ...
      }
    }]
)

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

SDK for Java 2.x

```
public static void getRecommendationsWithPromotedItems(PersonalizeRuntimeClient
 personalizeRuntimeClient,
                                               String campaignArn,
                                               String userId,
                                               String requestFilterArn,
                                               String requestParameterName,
                                               String requestParameterValue1,
                                               String requestParameterValue2,
                                               String promotionName,
                                               int percentPromotedItems,
                                               String promotionFilterArn,
                                               String promotionParameterName,
                                               String promotionParameterValue1,
                                               String promotionParameterValue2) {

  try {

      Map<String, String> promotionFilterValues = new HashMap<>();

      promotionFilterValues.put(promotionParameterName, String.format("\"%1$s\",
\"%2$s\"",
              promotionParameterValue1, promotionParameterValue2));
```

```
        Promotion newPromotion = Promotion.builder()
                .name(promotionName)
                .percentPromotedItems(percentPromotedItems)
                .filterArn(promotionFilterArn)
                .filterValues(promotionFilterValues)
                .build();

        List<Promotion> promotionList = new List<>();

        promotionsList.add(newPromotion);

        Map<String, String> requestfilterValues = new HashMap<>();

        requestfilterValues.put(requestParameterName, String.format("\"%1$s\",\"%2$s
\"",
                requestParameterValue1, requestParameterValue2));

        GetRecommendationsRequest recommendationsRequest =
 GetRecommendationsRequest.builder()
                .campaignArn(campaignArn)
                .numResults(20)
                .userId(userId)
                .filterArn(requestFilterArn)
                .fitlerValues(requestFilterValues)
                .promotions(promotionList)
                .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());
            System.out.println("Promotion name is : "+item.promotionName());
        }
    } catch (PersonalizeRuntimeException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { GetRecommendationsCommand, PersonalizeRuntimeClient } from
  "@aws-sdk/client-personalize-runtime";

// create personalizeRuntimeClient.
const personalizeRuntimeClient = new PersonalizeRuntimeClient({
  region: "REGION",
});

// set recommendation request param
export const getRecommendationsParam = {
  campaignArn: "CAMPAIGN_ARN",  /* required */
  userId: "USER_ID", /* required */
  numResults: 25, /* optional */
  filterArn: "FILTER_ARN", /* provide if you are applying a custom filter */
  filterValues: {
    "PARAM_NAME": "\"PARAM_VALUE\"" /* provide if your filter has a placeholder
 parameter */
  },
  promotions: [
    {
      name: "PROMOTION_NAME", /* specify the name of the promotion. The
 recommendation response includes the name to identify promoted items. */
      percentPromotedItems: 50, /* the percentage of recommended items to apply the
 promotion to. */
      filterArn:
        "PROMOTION_FILTER_ARN", /* the Amazon Resource Name (ARN) of the filter that
 defines the promotion criteria. */
      filterValues: {
        "PARAM_NAME": "\"PARAM_VALUE\"" /* provide if your promotion filter has a
 placeholder parameter */
      },
    },
  ],
};

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
 GetRecommendationsCommand(getRecommendationsParam));
    console.log("Success!", "\nItems are: ");
    response.itemList.forEach(element => console.log(element.itemId))
```

```
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

A list of recommended items displays. Promoted items are positioned randomly relative to other recommended items, but in sorted order relative to other promoted items. Depending on your recipe, recommended items that aren't part of a promotion are sorted by relevance to the user, popularity, or similarity. If there aren't enough items that meet the promotion criteria, the result will contain as many promoted items as possible.

```
{
  "itemList": [
      {
          "itemId1": "123",
          "score": .0117211,
          "promotionName": "promotionName"
      },
      {
         "itemId2": "456",
         "score": .0077976
      },
      {
         "itemId3": "789",
         "score": .0067171
      },
      .....
 ]
```

Getting action recommendations

If you use a PERSONALIZED_ACTIONS recipe, you can get action recommendations from your campaign in real time. You can get action recommendations with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs.

Topics

- How action recommendation scoring works
- Getting action recommendations (console)
- Getting action recommendations (AWS CLI)
- Getting action recommendations (AWS SDKs)

How action recommendation scoring works

With the Next-Best-Action recipe, Amazon Personalize generates scores for actions based on the likelihood that the user will interact with the action. Scores can be between 0 – 1.0. The closer to 1.0, the more likely it is that the user will interact with the action.

If you haven't imported any action interaction data, all recommended actions will have a score of 0.0. If Amazon Personalize recommends an action as part of *exploration*, the item will have a score of 0.0. Amazon Personalize uses exploration to recommend actions without action interaction data. For more information about exploration, see Exploration.

Getting action recommendations (console)

To get action recommendations with the Amazon Personalize console, you provide the request information on the details page of your custom campaign.

To get action recommendations

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're using.
3. In the navigation pane, under **Custom resources**, choose **Campaigns**.
4. Choose the target campaign.
5. Under **Test campaign results**, enter your recommendation request details.

   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 as if it is their `userId`. For more information about recording events for anonymous users, see Recording events for anonymous users.
6. Optionally choose a filter. For more information, see Filtering recommendations and user segments.

7. Choose **Get recommendations**. A table containing the user's top 5 recommended actions appears.

Getting action recommendations (AWS CLI)

Use the following code to get action recommendations from a campaign. Specify the ID of the user that you want to get recommendations for and the Amazon Resource Name (ARN) of your campaign.

To change the number of recommended actions, change the value for `numResults`. The default is 5 actions. The maximum is 100 actions.

To filter actions recommendations by custom criteria, you can create a filter and apply it to the `get-action-recommendations` operation. For more information, see Filtering recommendations and user segments.

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 as if it is their `userId`. For more information about recording events for anonymous users, see Recording events for anonymous users.

```
aws personalize-runtime get-action-recommendations \
--campaign-arn campaign arn \
--user-id User ID \
--num-results 10
```

Getting action recommendations (AWS SDKs)

The following code shows how to get Amazon Personalize recommendations for a user from a campaign. Specify the ID of the user you want to get recommendations for, and the Amazon Resource Name (ARN) of your campaign.

To change the number of recommended actions, change the value for `numResults`. The default is 5 actions. The maximum is 100 actions.

To filter actions recommendations by custom criteria, you can create a filter and apply it to the GetActionRecommendations API request. For more information, see Filtering recommendations and user segments.

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 as if it is their userId. For more information about recording events for anonymous users, see Recording events for anonymous users.

```
import boto3

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

response = personalizeRt.get_action_recommendations(
    campaignArn = 'Campaign ARN',
    userId = 'User ID',
    numResults = 10
)

print("Recommended actions")
for item in response['actionList']:
    print (item['actionId'])
```

Getting a personalized ranking (custom resources)

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

Topics

- How personalized ranking scoring works
- Getting a personalized ranking (console)
- Getting a personalized ranking (AWS CLI)
- Getting a personalized ranking (AWS SDKs)
- Personalized-Ranking sample notebook

How personalized ranking scoring works

Like the scores returned by the `GetRecommendations` operation for solutions created with the User-Personalization recipe, `GetPersonalizedRanking` scores sum to 1, but because the list of considered items is much smaller than your full catalog, 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:

$$\mathrm{score}(u, i) = \frac{\exp(\bar{w}_u^\top w_i)}{\sum_{j \in \mathrm{input}} \exp(\bar{w}_u^\top w_j)}$$

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 and user segments.
8. If you enabled metadata in recommendations for your campaign, for **Items dataset columns**, choose the metadata columns that you want to include in recommendation results. For information about enabling metadata, see Metadata with recommendations.

9. If your campaign uses contextual metadata (for requirements see [Increasing recommendation relevance with contextual metadata](#)) optionally provide context data.

   For each context, for the **Key**, enter the metadata field, and for the **Value**, enter the context data.

10. 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)

The following code samples show how different variations of how to get a personalized ranking with the AWS CLI.

Topics

- Getting a personalized ranking
- Including item metadata in a personalized ranking

Getting a personalized ranking

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
```

Including item metadata in a personalized ranking

If you enabled metadata in recommendations for your campaign, you can specify the Items dataset metadata columns to include in the response. For information about enabling metadata, see Metadata with recommendations.

The following code sample shows how to specify the metadata columns as part of your request for a personalized ranking.

```
aws personalize-runtime get-personalized-ranking \
--campaign-arn Campaign ARN \
--user-id 12 \
--input-list 3 4 10 8 12 7
--metadata-columns "{\"ITEMS\": [\"\"columnNameA"\",\"\"columnNameB"\"]}"
```

Getting a personalized ranking (AWS SDKs)

The following code samples show how different variations of how to get a personalized ranking with the AWS SDKs.

Topics

Getting a personalized ranking

The following code shows how to get a personalized ranking for a user. Specify the user's ID and a 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 returned. Amazon Personalize considers the first item in the list of most interest to the user.

SDK for Python (Boto3)

```
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

```
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;
}
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
```

```
import { GetPersonalizedRankingCommand } from
  "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set the ranking request parameters.
export const getPersonalizedRankingParam = {
  campaignArn: "CAMPAIGN_ARN", /* required */
  userId: 'USER_ID',      /* required */
  inputList: ["ITEM_ID_1", "ITEM_ID_2", "ITEM_ID_3", "ITEM_ID_4"]
}

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
 GetPersonalizedRankingCommand(getPersonalizedRankingParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Including item metadata in a personalized ranking

If you enabled metadata in recommendations for your campaign, you can specify the Items dataset metadata columns to include in the response. For information about enabling metadata, see Metadata with recommendations.

The following code sample shows how to specify the metadata columns as part of your request for a personalized ranking.

```
import boto3

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

response = personalizeRt.get_personalized_ranking(
    campaignArn = "Campaign arn",
```

```
    userId = "UserID",
    inputList = ['ItemID1','ItemID2'],
    metadataColumns = {
      "ITEMS": ['columnNameA','columnNameB']
    }
)

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

Getting a personalized ranking using contextual metadata

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](#).

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 item recommendations or get a personalized ranking.

To use contextual metadata, the schema of the Item interactions dataset must have a metadata fields for the contextual data. For example, a DEVICE field (see Schemas).

For Domain dataset groups, the following recommender use cases can use contextual metadata:

- Recommended for you (ECOMMERCE domain)
- Top picks for you (VIDEO_ON_DEMAND domain)

For custom resources, recipes that use contextual metadata include the following:

- User-Personalization
- Personalized-Ranking

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.

You can get recommendations with contextual metadata with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs.

Getting recommendations using contextual metadata (AWS Python SDK)

To increase recommendation relevance, include contextual metadata for a user, such as their device type or the time of day, when you get item recommendations or get a personalized ranking.

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. If you created a recommender, replace `campaignArn` with `recommenderArn`. 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'])
```

Batch recommendations and user segments (custom resources)

With custom resources, you can get batch recommendations or user segments with an asynchronous batch flow. For example, you might get product recommendations for all users on an email list or item-to-item similarities across an inventory. Or with the USER_SEGMENTATION recipes, you can get user segments for data-driven advertising based on items in your inventory and your user's interactions.

- To get *batch recommendations*, 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 *item recommendations*, and exports the recommendations to an Amazon S3 bucket.
- To get *user segments*, you use a batch segment job. A *batch segment job* is a tool that imports your batch input data from an Amazon S3 bucket, uses your solution version trained with a USER_SEGMENTATION recipe to generate *user segments*, and exports the segments to an Amazon S3 bucket.

Topics

Getting batch recommendations

With custom resources, you can get item recommendations with an asynchronous batch flow. For example, you might get product recommendations for all users on an email list or item-to-item similarities across an inventory.

To get batch recommendations for items, 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 custom solution version to generate *item recommendations*, and then exports the item recommendations to an Amazon S3 bucket. Depending on the recipe, your input data is a list of users, or items, or a list of users each with a collection of items.

If your solution uses the Similar Items recipe and you have an Items dataset with textual data and item title data, you can generate batch recommendations with themes for each group of items. For more information, see [Batch recommendations with themes from Content Generator](#).

When generating batch recommendations, Amazon Personalize considers all bulk data present at the time of latest solution version creation. This data can be imported with an import mode of FULL or INCREMENTAL. For newer bulk records to influence batch recommendations, you must create a new solution version and then create the batch inference job.

Amazon Personalize uses data from individual imports when generating batch recommendations as follows:

- New interactions with existing items and users: If you use the User-Personalization or Personalized-Ranking recipes, Amazon Personalize considers new interactions data with existing items and users within about 15 minutes from data import. To make sure events are considered, we recommend you wait at minimum 15 minutes after import before you start a batch inference job. For all other recipes, you must create a new solution version for streamed events to influence batch recommendations.
- New users: For users without interactions data, recommendations are initially for only popular items. If you use use User-Personalization or Personalized-Ranking and you record events for the user, their recommendations might become more relevant within about 15 minutes after import without retraining. To make sure events are considered, we recommend you wait at minimum 15 minutes after import before you start a batch inference job. For all other recipes, you must create a new solution version for streamed events to influence batch recommendations for users without interactions data.
- New items: With User-Personalization, when you create a batch inference job and specify the latest fully trained solution version for your solution, Amazon Personalize automatically updates the solution version to include new items in recommendations with exploration. If you don't specify the latest solution version, no update occurs. For any other recipe, you must create a new solution version for new items to be featured in batch recommendations. For more information about exploration, see [Exploration](#).

Topics

- Batch workflow
- Guidelines and requirements
- Batch workflow scoring
- Batch recommendations with themes from Content Generator
- Preparing input data for batch recommendations
- Creating a batch inference job
- Batch inference job output examples

Batch workflow

The batch workflow is as follows:

1. Prepare and upload your input data in JSON format to an Amazon S3 bucket. The format of your input data depends on the recipe you use. See Preparing input data for batch recommendations.
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.
4. When the batch inference is complete, retrieve the item recommendations from your output location in Amazon S3.

Guidelines and requirements

The following are guidelines and requirements for getting batch recommendations:

- Your Amazon Personalize IAM service role must have permission to read and add files to your Amazon S3 buckets. For information on granting permissions, see Service role policy for batch workflows. For more information on bucket permissions, see User policy examples in the *Amazon Simple Storage Service Developer Guide*. If you use AWS Key Management Service (AWS KMS) for encryption, you must grant Amazon Personalize and your Amazon Personalize IAM service role permission to use your key. For more information, see Giving Amazon Personalize permission to use your AWS KMS key.
- You must create a custom solution and solution version before you create a batch inference job. However, you don't need to create an Amazon Personalize campaign. If you created a Domain dataset group, you can still create custom resources.

- To generate themes with recommendations, you must use the Similar-Items recipe. And you must have an Items dataset with textual data and item title data. For more information about themed recommendations, see Batch recommendations with themes from Content Generator.
- Your input data must be formatted as described in Preparing input data for user segments.
- You can't get batch recommendations with the Trending-Now or Next-Best-Action recipes.
- If you use a filter with placeholder parameters, you must include the values for the parameters in your input data in a `filterValues` object. For more information, see Providing filter values in your input JSON.
- We recommend that you use a different location for your output data (either a folder or a different Amazon S3 bucket) than your input data.
- Batch recommendations might not be exactly the same as real-time recommendations. This is because batch inference jobs take longer to complete and only consider data available 15 minutes before the start of the job.

Batch workflow scoring

Batch recommendations include scores as follows:

- With User-Personalization and Personalized-Ranking recipes, Amazon Personalize calculates batch inference job recommendation scores as described in How User-Personalization recommendation scoring works (custom resources) and How personalized ranking scoring works. You can view scores in the batch inference job's output JSON file.
- With the Similar-Items recipe, if you get themed batch recommendations, Amazon Personalize ranks each set of related items based on how relevant the theme is for each item. Each item includes a score from 0 to 1. The higher the score, the more closely related the item is to the theme. For more information about recommendations with themes, see Batch recommendations with themes from Content Generator.

Batch recommendations with themes from Content Generator

Important

When you get batch recommendations with themes, you incur additional costs. For more information, see Amazon Personalize pricing.

If you use the Similar-Items recipe, Amazon Personalize Content Generator can add descriptive themes to batch recommendations. *Content Generator* is a generative artificial intelligence (generative AI) capability managed by Amazon Personalize.

When you get batch recommendations with themes, Amazon Personalize Content Generator adds a descriptive theme for each set of similar items. The theme is based on the item description and item name data in your Items dataset. Amazon Personalize includes the themes in the output of the batch inference job. You can use the themes to make the text in your application or marketing messages more compelling.

For example, if you get related items recommendations for a breakfast food item, Amazon Personalize might generate a theme like *Rise and shine* or *Morning essentials*. You might use the theme to replace a generic carousel title, like *Frequently bought together*. Or you might incorporate the theme in a promotional email or marketing campaign for new menu options.

AWS doesn't monitor themes from Content Generator. To confirm the theme quality, you can use the scores produced for each recommended item. For more information, see Ranking and scoring for batch recommendations with themes.

Topics

Supported regions

Amazon Personalize Content Generator is only available in the following AWS Regions:

- US East (N. Virginia)
- US West (Oregon)
- Asia Pacific (Tokyo)

Guidelines and requirements

The following are guidelines and requirements for generating recommendations with themes:

- Your input file can have up to 100 items. For information about input data for batch recommendations, see [Preparing input data for batch recommendations](#).
- Your solution must use the [Similar-Items recipe](#).
- You must have an Items dataset with the following data. This data can help generate more relevant themes.
  - It must have a textual field, such as a DESCRIPTION field. For information about textual data, see [Unstructured text metadata](#).
  - It must have a string column with item name data, such as a TITLE field.

  If your Items dataset doesn't have this data, you can add it. For information about updating existing data, see [Updating data](#).

Ranking and scoring for batch recommendations with themes

When you get batch recommendations with themes, Amazon Personalize ranks each set of items based on how relevant the theme is for each item. Each item includes a score in a rough range of -0.1 and 0.6. The higher the score, the more closely related the item is to the theme. You might use the scores to set a threshold to show only items that are strongly related to the theme.

For example, Amazon Personalize might return a theme of `For your sweet tooth`, and the related items and their scores might be: hard candy (score 0.19884521), chocolate (score .17664525), apple (score .08994528), popsicle (score .14294521), sweet potato (score .07794527), and carrot (score .04994523). In your application, you might add a rule to include only items with a score of `.10` or greater, eliminating the fruits and vegetables.

The following example shows the format of the output of a batch inference job that generates movie recommendations with themes.

```
{"input":{"itemId":"40"},"output":{"recommendedItems":
["36","50","44","22","21","29","3","1","2","39"],"theme":"Movies
 with a strong female lead","itemsThemeRelevanceScores":
[0.19994527,0.183059963,0.17478035,0.1618133,0.1574806,0.15468733,0.1499242,0.14353688,0.135314
{"input":{"itemId":"43"},"output":{"recommendedItems":
["50","21","36","3","17","2","39","1","10","5"],"theme":"The best movies of
 1995","itemsThemeRelevanceScores":
[0.184988,0.1795761,0.11143453,0.0989443,0.08258403,0.07952615,0.07115086,0.0621634,-0.138913,-
...
```

Generating batch recommendations with themes

To generate batch recommendations with themes, you complete the batch workflow as described in Batch workflow. You prepare your input data in the same way you would for a `RELATED_ITEMS` recipe. For an example, see RELATED_ITEMS recipes.

When you create the batch inference job, you enable theme generation and specify the item title column of your Items dataset.

- For information about using the Amazon Personalize console to create a batch inference job that generates themes, see Creating a batch inference job.
- For a code sample that shows how to use the SDK for Python (Boto3) to create a batch inference job that generates themes, see Creating a batch inference job that generates themes.

Preparing input data for batch recommendations

A batch inference job imports your batch input JSON data from an Amazon S3 bucket, uses your custom solution version to generate recommendations, and then exports the item recommendations to an Amazon S3 bucket. 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. You can then run multiple batch inference jobs using the same input data location.

If you use a filter with placeholder parameters, such as `$GENRE`, you must provide the values for the parameters in a `filterValues` object in your input JSON. For more information see Providing filter values in your input JSON.

To prepare and import data

1. Format your batch input data depending on your recipe. You can't get batch recommendations with the Trending-Now recipe.

   - For USER_PERSONALIZATION recipes and the Popularity-Count recipe, your input data is a JSON file with a list of userIds
   - For RELATED_ITEMS recipes, your input data is a list of itemIds
   - For PERSONALIZED_RANKING recipes, your input data is a list of userIds, each paired with a collection of itemIds

Separate each row with a new line. For input data examples, see [Batch inference job input and output JSON examples](#).

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. By creating a separate location for the output JSON, you can run multiple batch inference jobs with the same input data location.
4. Create a batch inference job. Amazon Personalize outputs the recommendations from your solution version to your output data location.

Batch inference job input and output JSON examples

How you format your input data the recipe you use. If you use a filter with placeholder parameters, such as `$GENRE`, you must provide the values for the parameters in a `filterValues` object in your input JSON. For more information see [Providing filter values in your input JSON](#).

The following sections list correctly formatted JSON input and output examples for batch inference jobs. You can't get batch recommendations with the Trending-Now recipe.

Topics

- [USER_PERSONALIZATION recipes](#)
- [POPULAR_ITEMS recipes (Popularity-Count only)](#)
- [PERSONALIZED_RANKING recipes](#)
- [RELATED_ITEMS recipes](#)

USER_PERSONALIZATION recipes

The following shows correctly formatted JSON input and output examples for USER_PERSONALIZATION 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","6237
[0.0152238,0.0069081,0.0068222,0.006394,0.0059746,0.0055851,0.0049357,0.0044644,0.0042968,0.
{"input":{"userId":"663"},"output":{"recommendedItems":
["368","377","25","780","1610","648","1270","6","165","1196","1097","300","1183","608","104"
[0.0406197,0.0372557,0.0254077,0.0151975,0.014991,0.0127175,0.0124547,0.0116712,0.0091098,0.
{"input":{"userId":"3384"},"output":{"recommendedItems":
["597","21","223","2144","208","2424","594","595","920","104","520","367","2081","39","1035"
[0.0241061,0.0119394,0.0118012,0.010662,0.0086972,0.0079428,0.0073218,0.0071438,0.0069602,0.
...
```

POPULAR_ITEMS recipes (Popularity-Count only)

The following shows correctly formatted JSON input and output examples for the Popularity-Count recipe. You can't get batch recommendations with the Trending-Now recipe.

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

The following shows correctly formatted JSON input and output examples for 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

The following shows correctly formatted JSON input and output examples for 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"]}}
...
```

The following shows correctly formatted JSON input and output examples for the Similar-Items recipe with themes.

Input

Separate each `itemId` with a new line as follows.

```
{"itemId": "40"}
{"itemId": "43"}
...
```

Output

```
{"input":{"itemId":"40"},"output":{"recommendedItems":
["36","50","44","22","21","29","3","1","2","39"],"theme":"Movies
 with a strong female lead","itemsThemeRelevanceScores":
[0.19994527,0.183059963,0.17478035,0.1618133,0.1574806,0.15468733,0.1499242,0.14353688,0.135
{"input":{"itemId":"43"},"output":{"recommendedItems":
["50","21","36","3","17","2","39","1","10","5"],"theme":"The best movies of
 1995","itemsThemeRelevanceScores":
[0.184988,0.1795761,0.11143453,0.0989443,0.08258403,0.07952615,0.07115086,0.0621634,-0.13891
...
```

Creating a batch inference job

Create a batch inference job to get batch item recommendations for users based on input data from Amazon S3. The input data can be a list of users or items (or both) in JSON format. You can create a batch inference job with the Amazon Personalize console, the AWS Command Line Interface (AWS CLI), or AWS SDKs.

When you create a batch inference job, you specify the Amazon S3 paths to your input and output locations. Amazon S3 is prefix based. If you provide a prefix for the input data location, Amazon Personalize uses all files matching that prefix as input data. For example, if you provide `s3://<name of your S3 bucket>/folderName` and your bucket also has a folder with a path of `s3://<name of your S3 bucket>/folderName_test`, Amazon Personalize uses all files in both folders as input data. To use only the files within a specific folder as input data, end the Amazon S3 path with a prefix delimiter, such as /: `s3://<name of your S3 bucket>/folderName/` For more information about how Amazon S3 organizes objects, see Organizing, listing, and working with your objects.

For more information about the batch workflow in Amazon Personalize, including permissions requirements, recommendation scoring, and preparing and importing input data, see Getting batch recommendations.

Topics

Creating a batch inference job (console)

After you have completed Preparing input data for batch recommendations, 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 create a batch inference job (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. From the navigation pane, under **Custom resources**, choose **Batch inference jobs**.
4. Choose **Create batch inference job**.
5. Choose the batch inference job type.
   - To generate item recommendations without themes, choose **Item recommendations**.
   - If you use the Similar-Items recipe and want to add descriptive themes to groups of similar items, choose **Themed recommendations with Content Generator**. To generate themes, you must have an Items dataset with item name data and textual data. For more information, see Batch recommendations with themes from Content Generator.
6. In **Batch inference job details**, in **Batch inference job name**, specify a name for your batch inference job.

7. For **Solution**, choose the solution and then choose the **Solution version ID** that you want to use to generate the recommendations.
8. For **Number of results**, optionally specify the number of recommendations for each line of input data. The default is 25.
9. If your batch job generates recommendations with themes, in **Themed recommendations details**, choose the column containing names or titles for the items in your Items dataset. This data can help generate more relevant themes. For more information, see Batch recommendations with themes from Content Generator.
10. In **Input source**, 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>.json`**

    Your input data must be in the correct format for the recipe your solution uses. For input data examples see Batch inference job input and output JSON examples.
11. For **Decryption key**, if you use your own AWS KMS key for bucket encryption, specify the Amazon Resource Name (ARN) of your key. Amazon Personalize must have permission to use your key. For information about granting permissions, see Giving Amazon Personalize permission to use your AWS KMS key.
12. In **Output destination**, 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>/`**
13. For **Encryption key**, if you use your own AWS KMS key for encryption, specify the ARN of your key. Amazon Personalize must have permission to use your key. For information about granting permissions, see Giving Amazon Personalize permission to use your AWS KMS key.
14. 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.
15. In **Filters** optionally choose a filter to apply a filter to the batch recommendations. If your filter uses placeholder parameters, make sure the values for the parameters are included in your input JSON. For more information see Providing filter values in your input JSON.
16. For **Tags**, optionally add any tags. For more information about tagging Amazon Personalize resources, see Tagging Amazon Personalize resources.

17. 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.

    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 input data for batch recommendations, you are ready to create a batch inference job with the CreateBatchInferenceJob operation.

Topics

Creating a batch inference job

You can use the `create-batch-inference-job` command to create a batch inference job. 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. Optionally provide a filter ARN to filter recommendations. If your filter uses placeholder parameters, make sure the values for the parameters are included in your input JSON. For more information see Filtering batch recommendations and user segments (custom resources).

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). Use the following syntax for input and output locations: **`s3://<name of your S3 bucket>/<folder name>/<input JSON file name>.json`** 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.

```
aws personalize create-batch-inference-job \
--job-name Batch job name \
--solution-version-arn Solution version ARN \
--filter-arn Filter 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\"}}"
```

Creating a batch inference job that generates themes

To generate themes for similar items, you must use the Similar-Items recipe and your Items dataset must have a textual field and a column of item name data. For more information about recommendations with themes, see Batch recommendations with themes from Content Generator.

The following code creates a batch inference job that generates recommendations with themes. Leave the `batch-inference-job-mode` set to `THEME_GENERATION`. Replace `COLUMN_NAME` with the name of the column that stores your item name data.

```
aws personalize create-batch-inference-job \
--job-name Themed batch job name \
--solution-version-arn Solution version ARN \
--filter-arn Filter 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-mode THEME_GENERATION \
--theme-generation-config "{\"fieldsForThemeGeneration\": {\"itemName\":
\"COLUMN_NAME\"}}"
```

Creating a batch inference job (AWS SDKs)

After you have completed Preparing input data for batch recommendations, you are ready to create a batch inference job with the CreateBatchInferenceJob operation.

Topics

- Creating a batch inference job
- Creating a batch inference job that generates themes

Creating a batch inference job

You can use the following code to create a batch inference job. Specify a job name, the Amazon Resource Name (ARN) of your solution version, and 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.

We recommend using a different location for your output data (either a folder or a different Amazon S3 bucket). Use the following syntax for input and output locations: **`s3://<name of your S3 bucket>/<folder name>/<input JSON file name>.json`** and **`s3://<name of your S3 bucket>/<output folder name>/`**.

For `numResults`, specify the number of items you want Amazon Personalize to predict for each line of input data. Optionally provide a filter ARN to filter recommendations. If your filter uses placeholder parameters, make sure the values for the parameters are included in your input JSON. For more information see Filtering batch recommendations and user segments (custom resources).

SDK for Python (Boto3)

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.

```
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",
    filterArn = "Filter 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>.json"}},
    jobOutput =
        {"s3DataDestination": {"path": "s3://<name of your S3 bucket>/<output folder
 name>/"}}
)
```

SDK for Java 2.x

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.

```
public static String createPersonalizeBatchInferenceJob(PersonalizeClient
 personalizeClient,
                                                        String solutionVersionArn,
                                                        String jobName,
                                                        String filterArn,
                                                        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)
                .filterArn(filterArn)
                .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 "";
}
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchInferenceJobCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch inference job's parameters.

export const createBatchInferenceJobParam = {
  jobName: 'JOB_NAME',
  jobInput: {         /* required */
    s3DataSource: {   /* required */
      path: 'INPUT_PATH', /* required */
      // kmsKeyArn: 'INPUT_KMS_KEY_ARN' /* optional */'
    }
  },
  jobOutput: {         /* required */
    s3DataDestination: {   /* required */
```

```
      path: 'OUTPUT_PATH', /* required */
      // kmsKeyArn: 'OUTPUT_KMS_KEY_ARN' /* optional */'
    }
  },
  roleArn: 'ROLE_ARN', /* required */
  solutionVersionArn: 'SOLUTION_VERSION_ARN', /* required */
  numResults: 20 /* optional integer*/
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateBatchInferenceJobCommand(createBatchInferenceJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Processing the batch job might take a while to complete. You can check a job's status by calling DescribeBatchInferenceJob 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.

Creating a batch inference job that generates themes

To generate themes for similar items, you must use the Similar-Items recipe and your Items dataset must have a textual field and a column of item name data. For more information about recommendations with themes, see Batch recommendations with themes from Content Generator.

The following code creates a batch inference job that generates recommendations with themes. Leave the `batchInferenceJobMode` set to `"THEME_GENERATION"`. Replace `COLUMNN_NAME` with the name of the column that stores your item name data.

```
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",
    filterArn = "Filter ARN",
    batchInferenceJobMode = "THEME_GENERATION",
    themeGenerationConfig = {
      "fieldsForThemeGeneration": {
          "itemName": "COLUMN_NAME"
      }
    },
    jobInput =
       {"s3DataSource": {"path": "s3://<name of your S3 bucket>/<folder name>/<input
 JSON file name>.json"}},
    jobOutput =
       {"s3DataDestination": {"path": "s3://<name of your S3 bucket>/<output folder
 name>/"}}
)
```

Batch inference job output examples

When you create a batch inference job, the job imports your batch input data from an Amazon S3 bucket, uses your solution version to generate *item recommendations*, and exports the recommendations to an Amazon S3 bucket in JSON format.

The following sections list output file examples for batch inference jobs by recipe type. You can't get batch recommendations with the Trending-Now or Next-Best-Action recipes.

Topics

- USER_PERSONALIZATION recipes
- POPULAR_ITEMS recipes
- PERSONALIZED_RANKING recipes
- RELATED_ITEMS recipes

USER_PERSONALIZATION recipes

The following is an example of the output JSON file for a USER_PERSONALIZATION recipe.

```
{"input":{"userId":"4638"},"output":{"recommendedItems":
["63992","115149","110102","148626","148888","31685","102445","69526","92535","143355","62374",
[0.0152238,0.0069081,0.0068222,0.006394,0.0059746,0.0055851,0.0049357,0.0044644,0.0042968,0.004
```

```
{"input":{"userId":"663"},"output":{"recommendedItems":
["368","377","25","780","1610","648","1270","6","165","1196","1097","300","1183","608","104","4
[0.0406197,0.0372557,0.0254077,0.0151975,0.014991,0.0127175,0.0124547,0.0116712,0.0091098,0.008
{"input":{"userId":"3384"},"output":{"recommendedItems":
["597","21","223","2144","208","2424","594","595","920","104","520","367","2081","39","1035","2
[0.0241061,0.0119394,0.0118012,0.010662,0.0086972,0.0079428,0.0073218,0.0071438,0.0069602,0.005
...
```

POPULAR_ITEMS recipes

The following example shows the format of the output JSON file for the Popularity-Count recipe. You can't get batch recommendations with the Trending-Now recipe.

```
{"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

The following example shows the format of the output JSON file for a PERSONALIZED_RANKING recipe.

```
{"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

The following example shows the format of the output JSON file for a RELATED_ITEMS recipe.

```
{"input": {"itemId": "105"}, "output": {"recommendedItems": ["106", "107", "49"]}}
{"input": {"itemId": "106"}, "output": {"recommendedItems": ["105", "107", "49"]}}
{"input": {"itemId": "441"}, "output": {"recommendedItems": ["2", "442", "435"]}}
...
```

The following example shows the format of the output JSON file for the Similar-Items recipe with themes. For more information about recommendations with themes, see Batch recommendations with themes from Content Generator.

```
{"input":{"itemId":"40"},"output":{"recommendedItems":
["36","50","44","22","21","29","3","1","2","39"],"theme":"Movies
 with a strong female lead","itemsThemeRelevanceScores":
[0.19994527,0.183059963,0.17478035,0.1618133,0.1574806,0.15468733,0.1499242,0.14353688,0.135314
{"input":{"itemId":"43"},"output":{"recommendedItems":
["50","21","36","3","17","2","39","1","10","5"],"theme":"The best movies of
 1995","itemsThemeRelevanceScores":
[0.184988,0.1795761,0.11143453,0.0989443,0.08258403,0.07952615,0.07115086,0.0621634,-0.138913,-
...
```

Getting user segments

To get *user segments*, you use a batch segment job. A *batch segment job* is a tool that imports your batch input data from an Amazon S3 bucket and uses your solution version trained with a USER_SEGMENTATION recipe to generate *user segments* for each row of input data.

Depending on the recipe, the input data is a list of items or item metadata attributes in JSON format. For item attributes, your input data can include expressions to create user segments based on multiple metadata attributes. A batch segment job exports user segments to an output Amazon S3 bucket. Each user segment is sorted in descending order based on the probability that each user will interact with the item in your input data.

When generating user segments, Amazon Personalize considers data in datasets from bulk and individual imports:

- For bulk data, Amazon Personalize generates segments using only the bulk data present at the last full solution version training. And it uses only bulk data that you imported with an import mode of FULL (replacing existing data).
- For data from individual data import operations, Amazon Personalize generates user segments using the data present at the last full solution version training. To have newer records impact user segments, create a new solution version and then create a batch segment job.

Generating user segments works as follows:

1. Prepare and upload your input data in JSON format to an Amazon S3 bucket. The format of your input data depends on the recipe you use and the job you are creating. See Preparing input data for user segments.
2. Create a separate location for your output data, either a different folder or a different Amazon S3 bucket.
3. Create a batch segment job. See Creating a batch segment job.
4. When the batch segment job is complete, retrieve the user segments from your output location in Amazon S3.

Topics

Guidelines and requirements

The following are guidelines and requirements for batch getting batch segments:

- You must use a USER_SEGMENTATION recipe.
- Your Amazon Personalize IAM service role needs permission to read and add files to your Amazon S3 buckets. For information on granting permissions, see Service role policy for batch workflows. For more information on bucket permissions, see User policy examples in the *Amazon Simple Storage Service Developer Guide*.

  If you use AWS Key Management Service (AWS KMS) for encryption, you must grant Amazon Personalize and your Amazon Personalize IAM service role permission to use your key. For more information, see Giving Amazon Personalize permission to use your AWS KMS key.
- You must create a custom solution and solution version before you create a batch inference job. However, you don't need to create an Amazon Personalize campaign. If you created a Domain dataset group, you can still create custom resources.
- Your input data must be formatted as described in Preparing input data for user segments.
- If you use the Item-Attribute-Affinity recipe, the attributes in your input data can't include unstructured textual item metadata, such as a product description.

- If you use a filter with placeholder parameters, you must include the values for the parameters in your input data in a `filterValues` object. For more information, see Providing filter values in your input JSON.
- We recommend that you use a different location for your output data (either a folder or a different Amazon S3 bucket) than your input data.

Preparing input data for user segments

Batch segment jobs use a solution version to make user segments based on data that you provide in an input JSON file. Before you can get user segments, 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. You can then run multiple batch inference jobs using the same input data location.

If you use a filter with placeholder parameters, such as `$GENRE`, you must provide the values for the parameters in a `filterValues` object in your input JSON. For more information see Providing filter values in your input JSON.

To prepare and import data

1. Format your batch input data depending on the recipe your solution uses. Separate input data element with a new line. Your input data is either a list of itemIds (Item-Affinity) or item attributes (Item-Attribute-Affinity).

   For item attributes, input data can include logical expressions with the `AND` operator to get users for multiple items or attributes per query. For more information, see Specifying item attributes for the Item-Attribute-Affinity recipe.

   For input data examples for both recipes, see Batch segment job input and output JSON examples.
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. By creating a separate location for the output JSON, you can run multiple batch segment jobs with the same input data location.

After you have prepared your input data and uploaded it to an Amazon S3 bucket, you are ready to generate user segments with a batch segment job. For more information, see [Creating a batch segment job](#).

Topics

- [Specifying item attributes for the Item-Attribute-Affinity recipe](#)
- [Batch segment job input and output JSON examples](#)

Specifying item attributes for the Item-Attribute-Affinity recipe

If you use the Item-Attribute-Affinity recipe, your input data is a list of item attributes. You can mix different columns of metadata. For example one row might be a numerical column and the next might be a categorical column. You can't use unstructured textual item metadata as an item attribute.

Your input item metadata can include logical expressions with the `AND` operator to get a user segment for multiple attributes. For example, a line of your input data might be `{"itemAttributes": "ITEMS.genres = "\Comedy\" AND ITEMS.genres = "\Action\""}` or `{"itemAttributes": "ITEMS.genres = "\Comedy\" AND ITEMS.audience = "\teen\""}`.

When you combine two attributes with the `AND` operator, you create a user segment with users who are more likely to interact with items that have both attributes based on the users interactions history. Unlike filter expressions (which use the `IN` operator for string equality), batch segment input expressions support only the = symbol for equality for string matching.

Batch segment job input and output JSON examples

For a batch segment job, your input data must be either a list of itemIds (Item-Affinity recipe) or item attributes (Item-Attribute-Affinity). Each line of input data is a separate inference query. Each user segment is sorted in descending order based on the probability that each user will interact with items in your inventory.

If you use a filter with placeholder parameters, such as `$GENRE`, you must provide the values for the parameters in a `filterValues` object in your input JSON. For more information see [Providing filter values in your input JSON](#).

The following are correctly formatted JSON input and output examples for batch segment jobs organized by recipe.

Item-Affinity

Input

Your input data can have a maximum of 500 items. Separate each `itemId` with a new line as follows.

```
{"itemId": "105"}
{"itemId": "106"}
{"itemId": "441"}
...
```

Output

```
{"input": {"itemId": "105"}, "output": {"recommendedUsers": ["106", "107", "49"]}}
{"input": {"itemId": "106"}, "output": {"recommendedUsers": ["105", "107", "49"]}}
{"input": {"itemId": "441"}, "output": {"recommendedUsers": ["2", "442", "435"]}}
...
```

Item-Attribute-Affinity

Input

Your input data can have a maximum of 10 queries, where each query is one or more non-textual item attributes. Separate each attribute or attribute expression with a new line as follows.

```
{"itemAttributes": "ITEMS.genres = \"Comedy\" AND ITEMS.genres = \"Action\""}
{"itemAttributes": "ITEMS.genres = \"Comedy\""}
{"itemAttributes": "ITEMS.genres = \"Horror\" AND ITEMS.genres = \"Action\""}
...
```

Output

```
{"itemAttributes": "ITEMS.genres = \"Comedy\" AND ITEMS.genres = \"Action\"",
 "output": {"recommendedUsers": ["25", "78", "108"]}}
{"itemAttributes": "ITEMS.genres = \"Adventure\"", "output": {"recommendedUsers":
 ["87", "31", "129"]}}
{"itemAttributes": "ITEMS.genres = \"Horror\" AND ITEMS.genres = \"Action\"",
 "output": {"recommendedUsers": ["8", "442", "435"]}}
```

```
...
```

Creating a batch segment job

If you used a USER_SEGMENTATION recipe, you can create batch segment jobs to get user segments with your solution version. Each user segment is sorted in descending order based on the probability that each user will interact with items in your inventory. Depending on the recipe, your input data must be a list of items (Item-Affinity recipe) or item attributes (Item-Attribute-Affinity recipe) in JSON format. You can create a batch segment job with the Amazon Personalize console, the AWS Command Line Interface (AWS CLI), or AWS SDKs.

When you create a batch segment job, you specify the Amazon S3 paths to your input and output locations. Amazon S3 is prefix based. If you provide a prefix for the input data location, Amazon Personalize uses all files matching that prefix as input data. For example, if you provide `s3://<name of your S3 bucket>/folderName` and your bucket also has a folder with a path of `s3://<name of your S3 bucket>/folderName_test`, Amazon Personalize uses all files in both folders as input data. To use only the files within a specific folder as input data, end the Amazon S3 path with a prefix delimiter, such as /: `s3://<name of your S3 bucket>/folderName/` For more information about how Amazon S3 organizes objects, see Organizing, listing, and working with your objects.

Topics

Creating a batch segment job (console)

After you have completed Preparing input data for batch recommendations, you are ready to create a batch segment job. This procedure assumes that you have already created a solution and a solution version (trained model) with a USER_SEGEMENTATION recipe.

To get create a batch segment job (console)

1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/home and sign in to your account.
2. On the **Datasets group** page, choose your dataset group.

3. Choose **batch segment jobs** in the navigation pane, then choose **Create batch segment job**.
4. In **batch segment job details**, for **Batch segment job name**, specify a name for your batch segment job.
5. For **Solution**, choose the solution and then choose the **Solution version ID** that you want to use to generate the recommendations. You can create batch segment jobs only if you used a USER_SEGEMENTATION recipe.
6. For **Number of users**, optionally specify the number of users Amazon Personalize generates for each user segment. The default is 25. The maximum is 5 million.
7. For **Input source**, specify the Amazon S3 path to your input file or use the **Browse S3** to choose your Amazon S3 bucket.

   Use the following syntax: **`s3://<name of your S3 bucket>/<folder name>/<input JSON file name>.json`**

   Your input data must be in the correct format for the recipe your solution uses. For input data examples see Batch segment job input and output JSON examples.
8. For **Output destination**, specify the path to your output location or use the **Browse S3** to choose your Amazon S3 bucket. 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 **IAM role**, choose one of the following:
   - Choose **Create and use new service role** and enter the **Service role name** to create a new role, or
   - If you've already created a role with the correct permissions, choose **Use an existing service role** and choose the IAM role.

   The role you use must have read and write access to your input and output Amazon S3 buckets respectively.
10. For **Filter configuration** optionally choose a filter to apply a filter to the user segments. If your filter uses placeholder parameters, make sure the values for the parameters are included in your input JSON. For more information see Providing filter values in your input JSON.
11. For **Tags**, optionally add any tags. For more information about tagging Amazon Personalize resources, see Tagging Amazon Personalize resources.

12. Choose **Create batch segment job**. Batch segment job creation starts and the **Batch segment jobs** page appears with the **Batch segment job detail** section displayed.
13. When the batch segment 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 segment job (AWS CLI)

After you have completed Preparing input data for batch recommendations, you are ready to create a batch segment job using the following `create-batch-segment-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. For `num-results` specify the number of users you want Amazon Personalize to predict for each line of input data. The default is 25. The maximum is 5 million. Optionally provide a `filter-arn` to filter user segments. If your filter uses placeholder parameters, make sure the values for the parameters are included in your input JSON. For more information see Filtering batch recommendations and user segments (custom resources).

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). Use the following syntax for input and output locations: **`s3://<name of your S3 bucket>/<folder name>/<input JSON file name>.json`** and **`s3://<name of your S3 bucket>/<output folder name>/`**.

```
aws personalize create-batch-segment-job \
                --job-name Job name \
                --solution-version-arn Solution version ARN \
                --num-results The number of predicted users \
                --filter-arn Filter ARN \
                --job-input s3DataSource={path=s3://S3 input path} \
                --job-output s3DataDestination={path=s3://S3 output path} \
                --role-arn IAM service role ARN
{
   "batchSegmentJobArn": "arn:aws:personalize:us-west-2:acct-id:batch-segment-job/
batchSegmentJobName"
}
```

Creating a batch segment job (AWS SDKs)

After you have completed Preparing input data for batch recommendations, you are ready to create a batch segment job with the `CreateBatchSegmentJob` operation. The following code shows how to create a batch segment job. Give the job a name, specify the Amazon Resource Name (ARN) of the solution version to use, specify the ARN for your Amazon Personalize IAM role, and specify the Amazon S3 path to your input file and output locations. Your IAM service role must have read and write access to your input and output Amazon S3 buckets respectively.

We recommend using a different location for your output data (either a folder or a different Amazon S3 bucket). Use the following syntax for input and output locations: **`s3://<name of your S3 bucket>/<folder name>/<input JSON file name>.json`** and **`s3://<name of your S3 bucket>/<output folder name>/`**.

For `numResults`, specify the number of users you want Amazon Personalize to predict for each line of input data. The default is 25. The maximum is 5 million. Optionally provide a `filterArn` to filter user segments. If your filter uses placeholder parameters, make sure the values for the parameters are included in your input JSON. For more information see Filtering batch recommendations and user segments (custom resources).

SDK for Python (Boto3)

```
import boto3

personalize_rec = boto3.client(service_name='personalize')

personalize_rec.create_batch_segment_job (
    solutionVersionArn = "Solution version ARN",
    jobName = "Job name",
    numResults = Number of predicted users,
    filterArn = "Filter ARN",
    roleArn = "IAM service role ARN",
    jobInput =
       {"s3DataSource": {"path": "s3://<name of your S3 bucket>/<folder name>/<input
 JSON file name>.json"}},
    jobOutput =
       {"s3DataDestination": {"path": "s3://<name of your S3 bucket>/<output folder
 name>/"}}
)
```

SDK for Java 2.x

```
public static String createBatchSegmentJob(PersonalizeClient personalizeClient,
                                                          String solutionVersionArn,
                                                          String jobName,
                                                          String filterArn,
                                                          int numResults,
                                                          String
 s3InputDataSourcePath,
                                                          String
 s3DataDestinationPath,
                                                          String roleArn,
                                                          String explorationWeight,
                                                          String
 explorationItemAgeCutOff) {

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

  try {
      // Set up data input and output parameters.
      S3DataConfig inputSource = S3DataConfig.builder()
              .path(s3InputDataSourcePath)
              .build();
      S3DataConfig outputDestination = S3DataConfig.builder()
              .path(s3DataDestinationPath)
              .build();

      BatchSegmentJobInput jobInput = BatchSegmentJobInput.builder()
              .s3DataSource(inputSource)
              .build();
      BatchSegmentJobOutput jobOutputLocation = BatchSegmentJobOutput.builder()
              .s3DataDestination(outputDestination)
              .build();


      CreateBatchSegmentJobRequest createBatchSegmentJobRequest =
 CreateBatchSegmentJobRequest.builder()
              .solutionVersionArn(solutionVersionArn)
              .filterArn(filterArn)
              .jobInput(jobInput)
              .jobOutput(jobOutputLocation)
              .jobName(jobName)
```

```
                .numResults(numResults)
                .roleArn(roleArn)
                .build();

        batchSegmentJobArn =
 personalizeClient.createBatchSegmentJob(createBatchSegmentJobRequest)
                .batchSegmentJobArn();
        DescribeBatchSegmentJobRequest describeBatchSegmentJobRequest =
 DescribeBatchSegmentJobRequest.builder()
                .batchSegmentJobArn(batchSegmentJobArn)
                .build();

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

        // wait until the batch segment job is complete.
        while (Instant.now().getEpochSecond() < maxTime) {

            BatchSegmentJob batchSegmentJob = personalizeClient
                    .describeBatchSegmentJob(describeBatchSegmentJobRequest)
                    .batchSegmentJob();

            status = batchSegmentJob.status();
            System.out.println("batch segment 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 batchSegmentJobArn;

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

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { CreateBatchSegmentJobCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";

// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the batch segment job's parameters.

export const createBatchSegmentJobParam = {
  jobName: 'NAME',
  jobInput: {         /* required */
    s3DataSource: {   /* required */
      path: 'INPUT_PATH', /* required */
      // kmsKeyArn: 'INPUT_KMS_KEY_ARN' /* optional */'
    }
  },
  jobOutput: {         /* required */
    s3DataDestination: {   /* required */
      path: 'OUTPUT_PATH', /* required */
      // kmsKeyArn: 'OUTPUT_KMS_KEY_ARN' /* optional */'
    }
  },
  roleArn: 'ROLE_ARN', /* required */
  solutionVersionArn: 'SOLUTION_VERSION_ARN', /* required */
  numResults: 20 /* optional */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateBatchSegmentJobCommand(createBatchSegmentJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Processing the batch job might take a while to complete. You can check a job's status by calling DescribeBatchSegmentJob and passing a `batchSegmentJobArn` as the input parameter. You can also list all Amazon Personalize batch segment jobs in your AWS environment by calling ListBatchSegmentJobs.

Batch segment job output examples

A batch segment job imports your batch input data from an Amazon S3 bucket, uses your solution version trained with a USER_SEGMENTATION recipe to generate *user segments*, and exports the segments to an Amazon S3 bucket.

The following sections list correctly formatted JSON output examples for batch segment jobs by recipe.

Topics

- Item-Affinity
- Item-Attribute-Affinity

Item-Affinity

The following example shows the format of the output JSON file for the Item-Affinity recipe.

```
{"input": {"itemId": "105"}, "output": {"recommendedUsers": ["106", "107", "49"]}}
{"input": {"itemId": "106"}, "output": {"recommendedUsers": ["105", "107", "49"]}}
{"input": {"itemId": "441"}, "output": {"recommendedUsers": ["2", "442", "435"]}}
...
```

Item-Attribute-Affinity

The following example shows the format of the output JSON file for the Item-Attribute-Affinity recipe.

```
{"itemAttributes": "ITEMS.genres = \"Comedy\" AND ITEMS.genres = \"Action\"", "output":
 {"recommendedUsers": ["25", "78", "108"]}}
{"itemAttributes": "ITEMS.genres = \"Adventure\"", "output": {"recommendedUsers":
 ["87", "31", "129"]}}
{"itemAttributes": "ITEMS.genres = \"Horror\" AND ITEMS.genres = \"Action\"", "output":
 {"recommendedUsers": ["8", "442", "435"]}}
...
```

Maintaining recommendation relevance

Relevant recommendations can 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 custom resources up to date. This allows Amazon Personalize to learn from your user's most recent behavior and include your newest items in recommendations.

Topics

Keeping datasets current

As your catalog grows, update your historical data with bulk or individual data import operations. For more information about importing historical data, see Step 2: Preparing and importing data. For information on how data you import after training a model influences recommendations, see How new data influences real-time recommendations.

For use cases and recipes that provide personalized real-time recommendations, keep your Item interactions dataset up to date with your users' behavior. Do this by recording item interactions with an event tracker and the PutEvents API operation. Amazon Personalize updates recommendations based on your user's most recent activity as they interact with your catalog. For information about real-time personalization, see Real-time personalization. For more information on recording real-time events, see Recording events.

Maintaining domain recommenders

Amazon Personalize automatically retrains the models backing your recommenders every 7 days. This is a full retraining that creates entirely new models based on the entirety of the data in your datasets. If you modify the columns used in training, Amazon Personalize automatically starts a full retraining of the models backing your recommender.

- For *Top picks for you* and *Recommended for you* use cases, Amazon Personalize updates your recommender to consider new items for recommendations. Automatic updates are not a full retraining where the model learns from your users' behavior. Instead, automatic updates allow Amazon Personalize to feature your new items in recommendations before the recommender's next full retraining. For information about automatic updates, see Automatic updates.

- If you use the *Trending now* use case, Amazon Personalize automatically evaluates your interactions data every two hours and identifies trending items. You don't have to wait for your recommender to retrain.

While recommender retraining is in progress, you can still get recommendations from the recommender. Until the retraining completes, the recommender uses the previous configuration and models. To track updates, you can view the timestamp for the latest recommender update on the **Recommender details** page in the Amazon Personalize console. Or you can view the `latestRecommenderUpdate` details from the DescribeRecommender operation.

Maintaining custom solutions

By default, all new solutions use automatic training to create a new solution version every 7 days. Training continues until you delete the solution.

When you create a solution, we recommend that you use automatic training to manage solution version creation. This makes maintaining your solution easier. It removes the manual training required for the solution to learn from your more recent data. Without automatic training, you must manually create new solution versions for the solution to learn from your most recent data. For more information about configuring automatic training, see Configuring automatic training.

Your training frequency depends on your business requirements, the recipe that you use, and how frequently you import data. For all recipes, we recommend training at least weekly. With automatic training, this is the default training frequency. If you frequently add new items or actions, you might want to have a higher training frequency, depending on your recipe.

- If you use User-Personalization or Next-Best-Action, the solution automatically updates to consider new items or actions for recommendations. Automatic updates aren't the same as automatic training. An automatic update doesn't create a completely new solution version, and the model doesn't learn from your latest data. To maintain your solution, your training frequency should still be at least weekly. For more formation about automatic updates, including additional guidelines and requirements, see Automatic updates.
- If you use Trending-Now, Amazon Personalize automatically identifies the top trending items in your interactions data over a configurable interval of time. Trending-Now can recommend

items added since the last training through bulk or streaming interactions data. Your training frequency should still be at least weekly. For more information, see [Trending-Now recipe].

- If you don't use a recipe with automatic updates or the Trending-Now recipe, Amazon Personalize considers new items for recommendations only after the next training. For example, if you use the Similar-Items recipe, and you add new items daily, you must use a daily training frequency for these items to appear in recommendations that same day.

You can automate and schedule retraining and data import tasks with **Maintaining Personalized Experiences with Machine Learning**, which is 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.

Recording events

An *event* is an interaction between a user and your catalog. It can be an interaction with an *item*, such as a user purchasing an item or watching a video, or an *action*, such as taking the action. For example, applying for a credit card or enrolling in a membership program.

Amazon Personalize can make recommendations based on real-time event data only, historical event data only, or a mixture of both. Record real-time events as your customers interact with action recommendations. This builds out your interactions data and keeps your data fresh. And it tells Amazon Personalize about the current interests of your user, which can improve recommendation relevance.

If your domain use case or custom recipe supports real-time personalization, Amazon Personalize uses events in real time to update and adapt recommendations according to a user's evolving interest.

How you record real-time events depends on the type of interactions data you are importing:

- For *item interactions*, you record real-time events with the PutEvents API operation. Amazon Personalize appends this data to the Item interactions dataset in your dataset group. For more information, see Recording events.
- For *action interactions*, you record real-time events with the PutActionInteractions API operation. Amazon Personalize appends this data to the Action interactions dataset in your dataset group. Only the PERSONALIZED_ACTIONS recipes use action interactions data. For more information, see Recording action interaction events.

Topics

How real-time events influence recommendations

If your recipe supports real-time personalization, after you create a recommender or custom campaign, Amazon Personalize uses new recorded event data for existing items or actions within seconds of import. The following use cases and recipes support real-time personalization:

- Recommended for you (ECOMMERCE use case)
- Top picks for you (VIDEO_ON_DEMAND use case)
- User-Personalization recipe
- Personalized-Ranking recipe
- Next-Best-Action recipe

If you use the Trending-Now recipe, Amazon Personalize automatically considers items from new events data over configurable intervals. You don't have to create a new solution version. For more information, see Trending-Now recipe.

If the item, action, or user in the event is new, how the Amazon Personalize uses the data depends on your use case or recipe. For more information, see How new data influences real-time recommendations.

Recording item interaction events

An *item interaction event* is an interaction between a user and an item in your catalog. For example, a user purchasing shoes or watching movie.

Record real-time item interaction events as you show your customer item recommendations. This builds out your interactions data and keeps your data fresh. And it tells Amazon Personalize about the current interests of your user, which can improve recommendation relevance.

You record item interaction events with the PutEvents API operation. Amazon Personalize appends the event data to the *Item interactions dataset* in your dataset group. If you record two events with exactly the same timestamp and identical properties, Amazon Personalize keeps only one of the events. You can record item interaction events using the AWS SDKs, AWS Amplify, or AWS Command Line Interface (AWS CLI).

If you use Apache Kafka, you can use the *Kafka connector for Amazon Personalize* to stream item interactions in real time to Amazon Personalize. For information see Kafka Connector for Amazon Personalize in the *personalize-kafka-connector* Github repository.

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

Topics

Requirements for recording item interaction events and training a model

To record item interaction events, you need the following:

- A dataset group that includes an `Item interactions` dataset, which can be empty. If you went through the Getting started guide, you can use the same dataset group and dataset that you created. For information on creating a dataset group and a dataset, see Step 2: Preparing and importing data.
- An event tracker.
- A call to the PutEvents API operation.

You can start out with an empty Item interactions dataset and, when you have recorded enough data, train the model using only new recorded events. For all use cases (Domain dataset groups) and recipes (Custom dataset groups), your interactions data must have the following before training:

- At minimum 1000 item interactions records from users interacting with items in your catalog. These interactions can be from bulk imports, or streamed events, or both.
- At minimum 25 unique user IDs with at least two item interactions for each.

For quality recommendations, we recommend that you have at minimum 50,000 item interactions from at least 1,000 users with two or more item interactions each.

Creating an item interaction event tracker

Before you can record item interaction events, you must create an item interaction event tracker. An *event tracker* directs new event data to the *Item interactions dataset* in your dataset group.

You create an event tracker with the Amazon Personalize console or the [CreateEventTracker](#) API operation. You pass as a parameter the Amazon Resource Name (ARN) of the dataset group that contains the target Item interactions dataset. For instructions on creating an event tracker using the Amazon Personalize console, see [Creating an event tracker (console)](#).

An event tracker 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 Item interactions dataset of the dataset group you specify in your event tracker.

> **Note**
>
> You can create only one item interaction 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;
}
```

Using the PutEvents operation

After you create an *Item interactions dataset* and an event tracker for your dataset group, you are ready to record item interaction events. To record item interaction events, you use the PutEvents API operation. The following sections show how to record a single event, how to record multiple events with event value data, and how to include impressions data with an event.

For information on how to record events for anonymous users see Recording events for anonymous users.

Topics

- Recording a single item interaction event
- Recording multiple item interaction events with event value data
- Recording impressions data

Recording a single item interaction event

The following example shows a PutEvents operation that passes one item interaction event. The corresponding schema is shown, along with an example row from the Item interactions dataset.

Your application generates a unique sessionId when a user first visits your website or uses your application. You must use the same sessionId in all events throughout the session. Amazon Personalize uses the sessionId to associate events with the user before they log in (is anonymous). For more information see Recording events for anonymous users.

The event list is an array of Event objects. An eventType is required for each event, but in this example, eventType data is not used in training because it is not included in the schema. You can provide a placeholder value to satisfy the requirement.

The trackingId comes from the event tracker you created in Creating an item interaction event tracker. 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 Schemas.

Corresponding dataset columns

```
Dataset columns: USER_ID, ITEM_ID, TIMESTAMP
Item interactions dataset data: 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.

Recording multiple item interaction events with event value data

The following example shows how to record multiple item interaction events with different event types and different event values.

When you configure a solution, if your *Item interactions dataset* includes EVENT_TYPE and EVENT_VALUE fields, you can set a specific value as a threshold to exclude records from training. For more information, seeChoosing the item interaction data used for training.

The example also shows the recording of an extra property, `numRatings`, that is used as metadata by certain recipes.

```
Dataset columns: USER_ID, ITEM_ID, TIMESTAMP, EVENT_TYPE, EVENT_VALUE, NUM_RATINGS
Item 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 event1Type,
                              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 recipe or add the IMPRESSIONS field to your schema for a dataset in a Domain dataset group, 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.

> **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 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 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 and attribution reports

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

To generate CloudWatch reports that show the impact of recommendations, create a metric attribution and record user interactions with real-time recommendations. For information on creating a metric attribution, see Measuring impact of recommendations.

For each event, include recommendation ID of the recommendations you showed the user. Or include the event source, such as a third party. Import this data to compare different campaigns, recommenders, and third parties. You can import at most 100 event attribution sources.

- If you provide a `recommendationId`, Amazon Personalize automatically determines the source campaign or recommender and identifies it in reports in an EVENT_ATTRIBUTION_SOURCE column.

- If you provide both attributes, Amazon Personalize uses only the `eventAttributionSource`.
- If you don't provide a source, Amazon Personalize labels the source `SOURCE_NAME_UNDEFINED` in reports.

The following code shows how to provide an `eventAttributionSource` for an event in a PutEvents operation.

```
response = personalize_events.put_events(
    trackingId = 'eventTrackerId',
    userId= 'userId',
    sessionId = 'sessionId123',
    eventList = [{
        'eventId': 'event1',
        'eventType': 'watch',
        'sentAt': '1667260945',
        'itemId': '123',
        'metricAttribution': {
            'eventAttributionSource': 'thirdPartyServiceXYZ'
        }
    }]
)
statusCode = response['ResponseMetadata']['HTTPStatusCode']
print(statusCode)
```

The following code shows how to provide a `recommendationId` for an event in a PutEvents operation.

```
response = personalize_events.put_events(
    trackingId = 'eventTrackerId',
    userId= 'userId',
    sessionId = 'sessionId123',
    eventList = [{
        'eventId': 'event1',
        'eventType': 'watch',
        'sentAt': '1667260945',
        'itemId': '123',
        'recommendationId': 'RID-12345678-1234-1234-1234-abcdefghijkl'
    }]
)
statusCode = response['ResponseMetadata']['HTTPStatusCode']
print(statusCode)
```

Recording action interaction events

An *action interaction* event is an interaction between a user and an *action*. For example, a user enrolling in a membership program or applying for a credit card.

If you use a PERSONALIZED_ACTIONS custom recipe, record real-time action interaction events as your customers interact with action recommendations. This builds out your interactions data and keeps your data fresh. It also tells Amazon Personalize about the current interests of your user, which can improve recommendation relevance. Only the PERSONALIZED_ACTIONS custom recipes use action interactions data.

You record action interaction events with the PutActionInteractions API operation. Amazon Personalize appends this data to the Action interactions dataset in your dataset group.

An action interaction event must have an event type attribute, which can be one of the following:

- Taken – Record *Taken* events when a user takes a recommended action.
- Not Taken – Record *Not Taken* events when your user makes a deliberate choice to not take the action after viewing it. For example, if they choose *No* when you show them the action. *Not Taken* events can indicate that the customer isn't interested in the action.
- Viewed – Record *Viewed* events when you show a user an action before they make a choice to take or not take an action. Amazon Personalize uses *View* events to learn about your users' interests. For example, if a user views an action but doesn't take it, this user might not be interested in this action in the future.

You can record real-time events using the AWS SDKs, or AWS Command Line Interface (AWS CLI). If you record two events with exactly the same timestamp and identical properties, Amazon Personalize keeps only one of the events.

Topics

- Requirements for recording action interaction events
- Finding the ID of your action interaction event tracker
- Using the PutActionInteractions operation

Requirements for recording action interaction events

To record real-time action interaction events, you need the following:

- A dataset group that includes an `Action interactions dataset`, which can be empty. For information on creating a dataset group and a dataset, see Step 2: Preparing and importing data.
- The ID of your event tracker. You specify this ID in the PutActionInteractions operation. When you create an Action interactions dataset, Amazon Personalize automatically creates an action interaction event tracker for you. For more information, see Finding the ID of your action interaction event tracker.
- A call to the PutActionInteractions operation.

Finding the ID of your action interaction event tracker

When you create an Action interactions dataset, Amazon Personalize automatically creates an *action interaction* event tracker for you. You specify the tracker's ID in the PutActionInteractions API operation. Amazon Personalize uses it to direct new data to the *Action interactions dataset* in your dataset group.

You can find your event tracker's ID on the details page of your Action interactions dataset in the Amazon Personalize console. And you can find the ID by calling the DescribeDataset API operation. The following Python code returns the tracking ID for an Action interactions dataset.

```
import boto3

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

response = personalize.describe_dataset(
  datasetArn="Action interactions dataset ARN"
)

print(response['trackingId'])
return response['trackingId']
```

Using the PutActionInteractions operation

After you create an Action interactions dataset, you are ready to record action interaction events with the PutActionInteractions operation. The following sections show how to record a single event and how to record multiple events with event value data.

Topics

- Recording a single action interaction event

- [Recording multiple action interaction events](#)

Recording a single action interaction event

The following code shows a `PutActionInteractions` operation that passes a TAKEN event. You might record this event when you show a user recommendations from Amazon Personalize and they take an action, such as applying for your credit card.

The `actionInteractions` is an array of ActionInteraction objects. The `trackingId` comes from the event tracker Amazon Personalize created when you created your Action interactions dataset. For more information, see [Finding the ID of your action interaction event tracker](#).

Your application generates a unique `sessionId` when a user first visits your website or uses your application. You must use the same `sessionId` in all events throughout the session. Amazon Personalize uses the `sessionId` to associate events with the user before they log in (is anonymous). For more information, see [Recording events for anonymous users](#).

The `userId`, `actionId`, and `sentAt` parameters map to the USER_ID, ACTION_ID, EVENT_TYPE, and TIMESTAMP fields of the Action interactions dataset.

Corresponding Action interactions dataset

```
USER_ID, ACTION_ID, TIMESTAMP, EVENT_TYPE
user123, action-xyz, 1543631760, TAKEN
```

Code example

AWS CLI

```
aws personalize-events put-action-interactions \
--tracking-id 12345678-xxxx-xxxx-xxxx-xxxxxxxxxxxx \
--action-interactions '[{
  "userId": "user123",
  "sessionId": "abcdefg",
  "timestamp": 1543631760,
  "eventType": "TAKEN",
  "actionId": "action-xyz"}]'
```

SDK for Python (Boto3)

```
import boto3
```

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

response = personalize_events.put_action_interactions(
  trackingId='12345678-xxxx-xxxx-xxxx-xxxxxxxxxxxx',
  actionInteractions=[{
    'userId': 'user123',
    'sessionId': 'abcdefg',
    'timestamp': 1543631760,
    'eventType': 'Taken',
    'actionId': 'action-xyz'
  }]
)
```

Recording multiple action interaction events

The following code shows how to record multiple action interaction events for the same user with the same sessionId.

Corresponding Action interactions dataset

```
USER_ID, ACTION_ID, EVENT_TYPE, TIMESTAMP
user123, action123, Taken, 1543531139
user123, action345, Not Taken, 1543531139
```

AWS CLI

```
aws personalize-events put-action-interactions \
--tracking-id 6ddfe6b7-cd83-4dd4-b09d-4c35ecbacfe1 \
--action-interactions '[{
  "userId": "user123",
  "sessionId": "abcdefg",
  "timestamp": 1543531139,
  "eventType": "Taken",
  "actionId": "action123"
},
{
  "userId": "user123",
  "sessionId": "abcdefg",
  "timestamp": 1543531139,
  "eventType": "Not Taken",
```

```
  "actionId": "action345"}]'
```

SDK for Python (Boto3)

```
import boto3

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

response = personalize_events.put_action_interactions(
  trackingId='12345678-xxxx-xxxx-xxxx-xxxxxxxxxxxx',
  actionInteractions=[{
    'userId': 'user123',
    'sessionId': 'abcdefg',
    'timestamp': 1697848587,
    'eventType': 'Taken',
    'actionId': 'action123'
  },
  {
    'userId': 'user123',
    'sessionId': 'abcdefg',
    'timestamp': 1697848622,
    'eventType': 'Not Taken',
    'actionId': 'action345'
  }]
)
```

Recording events for anonymous users

Important

If you don't record at minimum one event with a `sessionId` and `userId` for a user, Amazon Personalize won't use the activity tracked to only the `sessionId` when training. And after training completes, recommendations will no longer be based on activity tracked to the `sessionId`.

You can record item interaction or action interaction events for users before they create an account. Record events for anonymous users to build a continuous event history with events from before and after they log in. This provides Amazon Personalize more interactions data about the user, which can help generate more relevant recommendations.

To record events for anonymous users (users that haven't logged in), for each event specify only a `sessionId`. Your application generates a unique `sessionId` when a user first visits your website or uses your application. You must use the same `sessionId` in all events throughout the session. Amazon Personalize uses the `sessionId` to associate events with the user before they log in.

Amazon Personalize doesn't use events from anonymous users when training until you associate them with a `userId`. For more information, see Building a continuous event history for anonymous users.

To provide real-time personalization for anonymous users, specify the `sessionId` as the `userId` in your GetRecommendations or GetActionRecommendations request.

- For a code samples that show how to record item interaction events with the PutEvents operation and a sessionId and userId, see Using the PutEvents operation.
- For a code samples that show how to record action interaction events with the PutActionInteractions operation and a sessionId and userId, see Using the PutActionInteractions operation.

Building a continuous event history for anonymous users

To build an event history for an anonymous user and have Amazon Personalize use their events when training, record at minimum one event with both a `sessionId` and a `userId`. Then you can record any number of events for the `userId`. After you start providing a `userId`, the `sessionId` can change. During the next full retraining, Amazon Personalize associates the `userId` with the anonymous user history tracked to the original `sessionId`.

After retraining completes, recommendations will be based on activity tracked to both the `sessionId` from the anonymous events and any events tracked to their `userId`.

> **Note**
>
> If your user doesn't create an account and you want Amazon Personalize to use the data when training, you can use the `sessionId` as the `userId` in events. However, if the user eventually creates an account, you won't be able to associate the events from their anonymous browsing with their new `userId`.

Third-party event tracking services

The following Customer Data Platforms (CDPs) can help you collect event data from your application and send it to Amazon Personalize.

- **Amplitude** – You can use Amplitude to track user actions to help you understand your users' behavior. For information on using Amplitude and Amazon Personalize, see the following AWS Partner Network (APN) blog post: Measuring the Effectiveness of Personalization with Amplitude and Amazon Personalize.
- **mParticle** – You can use mParticle to collect event data from your app. For an example that shows how to use mParticle and Amazon Personalize to implement personalized product recommendations, see How to harness the power of a CDP for machine learning: Part 2.
- **Segment** – You can use Segment to send your data to Amazon Personalize. For more information on integrating Segment with Amazon Personalize, see Amazon Personalize Destination.

Sample implementations

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 operation, see 2.View_Campaign_And_Interactions.ipynb in the **getting_started** folder of the amazon-personalize-samples GitHub repository.

For an example that shows how to stream events from users interacting with recommendations, see streaming_events in the Amazon Personalize samples GitHub repository.

For a complete example that contains the source code and supporting files to deploy real-time APIs that sit between your Amazon Personalize resources and client applications, see Real-Time Personalization APIs in the AWS samples GitHub repository. This project includes how to implement the following:

- User context and user event collection
- Response caching
- Decorating recommendations based on item metadata
- A/B testing
- API authentication

Managing data

After you import data into a dataset, you can analyze it, export it to an Amazon S3 bucket, update it, or delete it by deleting the dataset. For Items and Users datasets, you can replace the dataset's schema to add columns of data.

Topics

Updating data

After you import data into an Amazon Personalize dataset, you can update it with bulk or individual data import operations.

If you want to update an existing dataset to add additional columns of data, you can replace the dataset's schema with a new one that has the added columns. Then you can import the new columns of data. For more information, see Replacing a dataset's schema.

Topics

How new data influences real-time recommendations

After you create a recommender or custom solution version, how new data influences real-time recommendations depends on its type, the method of import, and the domain use case or custom recipe you use. The following sections explain how new data influences real-time recommendations before the next training.

Training can be a recommender's weekly automatic training, or automatic or manual solution version creation. For manual training with User-Personalization, you must set `trainingMode` to `FULL`.

For information on how new records influence batch recommendations, see Getting batch recommendations. For information on how new records influence batch segment jobs, see Getting user segments.

Topics

- New interactions
- New items
- New users
- New actions

New interactions

New interactions are item or action interactions that you import after the latest training.

For both real-time and bulk data, if interactions involve a new item or action, Amazon Personalize might consider it for recommendations without training. For more information, see New items or New actions.

Real-time events

For use cases and recipes that feature real-time personalization, Amazon Personalize immediately uses real-time interactions between a user and existing items or actions (records present at the latest training) when generating recommendations for the same user. For more information about real-time personalization, see Real-time personalization.

For any domain use cases and custom recipes that don't feature real-time personalization, such as recommending similar items, your model learns from real-time interactions data only after training.

Bulk interactions

For *bulk interactions*, for both incremental *and* full dataset import jobs, your model learns from bulk item interaction or action interaction data only after the next training. Bulk data isn't used to update recommendations for real-time personalization.

For more information about updating existing bulk data, see Updating existing bulk data.

New items

New items are items that you import after the latest training. They can come from either interactions data or item metadata in an Items dataset.

New items are considered for recommendations as follows:

- For *Top picks for you* and *Recommended for you* domain cases or User-Personalization or Next-Best-Action recipes, Amazon Personalize automatically updates the model every two hours. After each update, Amazon Personalize considers new items for recommendations as part of exploration. When considering the new item, Amazon Personalize considers any metadata for the item. However this data will have a greater effect on recommendations only after you record interactions for the item and train a new model. For information about updates, see Automatic updates.
- If you use the *Trending now* use case, Amazon Personalize automatically evaluates your interactions data every two hours and identifies trending items. You don't have to wait for your recommender to train. If you use the *Trending-Now recipe*, Amazon Personalize automatically considers all new items over configurable intervals without training. For information about configuring intervals, see Trending-Now recipe.
- If you don't use the Trending-Now recipe or your use case or recipe doesn't support automatic updates, Amazon Personalize will consider new items only after the next training.

New users

New users are users that you import after the latest training. They can come from either interactions data or user metadata in a Users dataset. For new, anonymous users (users without a userId), you can record events for the user with a `sessionId` and Amazon Personalize will associate events with the user before they log in. For more information see Recording events for anonymous users.

Amazon Personalize generates recommendations for new users as follows:

- If you use the Trending now domain use case or Trending-Now custom recipe, new users immediately receive recommendations for the top trending items. If you use the Popularity-Count recipe, new users immediately receive recommendations for items with the most interactions.

- For recipes or use cases that provide personalized recommendations for users, recommendations for new users are based on the early interaction histories of your existing users. The first items or actions these existing users interacted with are more likely to be recommended to new users. For the User-Personalization or Personalized-Ranking recipes, if you set `recency_mask` to `true`, recommendations also include items based on the latest popularity trends in your interactions data.

The following can increase recommendation relevance for new users:

- Interactions data – The primary way to improve recommendation relevance for a new user is to import data from their interactions with your items. For information about how new interactions data influences recommendations, see New interactions.
- User metadata – Importing user metadata, such as GENDER or MEMBERSHIP_STATUS, can improve recommendations. For metadata to influence recommendations, you must wait for your domain recommender's weekly automatic retraining to complete. Or you must manually create a new solution version.
- Contextual metadata – If your use case or recipe supports contextual metadata and your Item interactions dataset has metadata fields for contextual data, you can provide the user's context in your request for recommendations. This does not require retraining. For more information, see Increasing recommendation relevance with contextual metadata.

New actions

New actions are actions that you import since the latest training. They can come from either action interaction data or actions in an Actions dataset.

With the Next-Best-Action recipe, Amazon Personalize automatically updates a solution version every two hours. After each update, Amazon Personalize considers new actions for recommendations as part of exploration. When considering the new action, Amazon Personalize considers any metadata for the action. However, this data will have a greater effect on recommendations only after you record action interactions for the action and fully retrain. For information about updates, see Automatic updates

Replacing a dataset's schema

After you create an Items or Users dataset, you can replace its schema with a new or existing one. You might replace a dataset's schema if your data structure changed after you created the dataset.

For example, you might have a new column of item metadata that you want Amazon Personalize to consider during training. Or you might want to add a column of data to use only when filtering recommendations.

When you replace a dataset's schema, you must keep all fields in the previous schema and you can't change their data types or attributes. After you replace a dataset's schema, Amazon Personalize automatically excludes any new columns from training for any existing recommenders or custom solutions. For more guidelines and requirements, see Guidelines and requirements.

You can replace a dataset's schema with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), and AWS SDKs.

Topics

Guidelines and requirements

Before you replace the schema for a dataset, make sure that you're aware of the following guidelines and requirements:

- You can't replace the schema of an Item interactions dataset, Action interactions dataset or Actions dataset.
- You can add new fields to your replacement schema, but you must keep all fields in the previous schema. And you can't change their data types or attributes. For example, if the previous schema includes a `MEMBERSHIP_STATUS` field for categorical string data, the new schema you use must include a `MEMBERSHIP_STATUS` field with these attributes and data types.
- If the current schema has a field that you want to rename, or if you want to change its data types or attributes, you can add a new field with a new name and modified types or attributes. Then include the new field in training and exclude the old field. Any new fields must support `null` data. If the old field did not support null data, when you import data, you can use placeholder data to make sure your import matches the schema. For information about configuring the columns used by a recommender, see Updating a recommender. For information about configuring the columns used by a solution, see Configuring columns used when training.

- Any new fields must support `null` data. For information about adding a null type to a field, see Schema data types.
- After you replace a dataset's schema, Amazon Personalize automatically excludes any new columns from training for any existing recommenders or custom solutions. Using the modified dataset involves the following actions:
  - To use any new columns in training, import data that aligns with the new schema. Then update any recommenders to use any new columns, or create a new custom solution and configure the columns that it uses when training.

    For information about updating the columns used by a recommender, see Updating a recommender. For information about configuring the columns used by a solution, see Configuring columns used when training.
  - To use any columns only when filtering, import data that aligns with the new schema, create a filter that uses the new data, and apply your filter to your recommendation requests. You don't need to update any recommenders, or create or update any custom resources.

Replacing a dataset's schema (console)

To replace a dataset's schema with the Amazon Personalize console, you choose the dataset to modify and choose to replace with a new schema or use an existing one.

To replace a dataset's 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 your dataset group.
3. Choose **Datasets**, and choose the radio button for the dataset that you want to modify.
4. Choose **Actions**, and choose **Replace schema**.
5. In **Schema details**, choose to replace with a new schema or a previously created one.
6. Specify the new schema to use. If you have chosen to:
   - Replace with a new schema, then give the schema a name, and in **Schema definition**, make your changes to the schema JSON.
   - Use a previously created schema, then for **Previously created schema**, choose the schema that you want to use. Only eligible schemas are listed. For information about schema requirements, see Guidelines and requirements.

7. Choose **Replace**. When the dataset is active, you can start importing data that aligns with the new schema. For more information, see Step 2: Preparing and importing data.

Replacing a dataset's schema (AWS CLI)

To replace a dataset's schema with the AWS CLI, you use the `update-dataset` command, specify the Amazon Resource Name (ARN) of the dataset to update and the ARN of the new schema to use. You can't update the schema of an Item interactions dataset, Action interactions dataset or Actions dataset.

The following code shows how to update a dataset's schema with the AWS CLI. To replace a dataset's schema with a new one, first use the `create-schema` command. Then use the following code to replace the current schema with the new one. For information about creating a schema with the AWS CLI, see Creating a dataset and a schema (AWS CLI). For information about datasets and schema requirements, see Schemas.

```
aws personalize update-dataset \
--dataset-arn Dataset ARN \
--schema-arn New schema ARN
```

When the dataset is active, you can start importing data that aligns with the new schema. For more information, see Step 2: Preparing and importing data. For information about the latest update to the dataset, you can use the DescribeDataset operation.

Replacing a dataset's schema (AWS SDKs)

To replace a dataset's schema with the AWS SDKs, you use the `UpdateDataset` API operation. Specify the Amazon Resource Name (ARN) of the dataset to update and the new schema to use. You can't update the schema of an Item interactions dataset, Action interactions dataset or Actions dataset.

The following code shows how to replace a dataset's schema with the SDK for Python (Boto3). To replace a dataset's schema with a new one, first use the CreateSchema operation. Then use the following code to replace the current schema with the new one. For information about creating a schema with the AWS SDKs, see Creating a dataset and a schema (AWS SDKs). For information on dataset and schema requirements, see Schemas.

```
import boto3
```

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

update_dataset_response = personalize.update_dataset(
    datasetArn = 'dataset_arn',
    schemaArn = 'new_schema_arn'
)

print(update_dataset_response)
```

When the dataset is active, you can start importing data that aligns with the new schema. For more information, see Step 2: Preparing and importing data. For information about the latest update to the dataset, you can use the DescribeDataset operation.

Updating existing bulk data

If you previously created a dataset import job for a dataset, you can use another import job to add to or replace the existing bulk data. By default, a dataset import job replaces any existing data in the dataset that you imported in bulk. You can instead append the new records to existing data by changing the job's import mode.

To append data to an Item interactions dataset or Action interactions dataset with a dataset import job, you must have at minimum 1000 new item interaction or action interaction records.

If you already created a recommender or deployed a custom solution version with a campaign, how new bulk records influence recommendations depends on the domain use case or recipe that you use. For more information, see How new data influences real-time recommendations.

Filter updates for bulk records

Within 20 minutes of completing a bulk import, Amazon Personalize updates any filters you created in the dataset group with your new bulk data. This update allows Amazon Personalize to use the most recent data when filtering recommendations for your users.

Topics

Import modes

To configure how Amazon Personalize adds your new records to your dataset, you specify an import mode for your dataset import job:

- To overwrite all existing bulk data in your dataset, choose **Replace existing data** in the Amazon Personalize console or specify `FULL` in the CreateDatasetImportJob API operation. This doesn't replace data you imported individually, including events recorded in real time.
- To append the records to the existing data in your dataset, choose **Add to existing data** or specify `INCREMENTAL` in the `CreateDatasetImportJob` API operation. Amazon Personalize replaces any record with the same ID with the new one.

  To append data to an Item interactions dataset or Action interactions dataset with a dataset import job, you must have at minimum 1000 new item interaction or action interaction records.

If you haven't imported bulk records, the A**dd to existing data** option is not available in the console and you can only specify `FULL` in the `CreateDatasetImportJob` API operation. The default is a full replacement.

Updating bulk records (console)

> **Important**
>
> By default, a dataset import job replaces any existing data in the dataset that you imported in bulk. You can change this by specifying the job's import mode.

To update bulk data with the Amazon Personalize console, create a dataset import job for the dataset and specify an import mode.

To update bulk records (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. From the navigation pane, choose **Datasets**.
4. On the **Datasets** page, choose the dataset you want to update.
5. In **Dataset import jobs**, choose **Create dataset import job**.

6. In **Import job details**, for **Dataset import job name**, specify a name for your import job.
7. For **Import mode**, choose how to update the dataset. Choose either **Replace existing data** or **Add to existing data**. data. For more information see Import modes.
8. In **Input source**, for **S3 Location**, specify where your data file is stored in Amazon S3. Use the following syntax:

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

   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. In **IAM role**, choose to either create a new role or use an existing one. If you completed the prerequisites, choose **Use an existing service role** and specify the role that you created in Creating an IAM role for Amazon Personalize.
10. For **Tags**, optionally add any tags. For more information about tagging Amazon Personalize resources, see Tagging Amazon Personalize resources.
11. Choose **Finish**. The data import job starts and the **Dataset overview** page displayed. The dataset import is complete when the status is ACTIVE.

Updating bulk records (AWS CLI)

> **Important**
>
> By default, a dataset import job replaces any existing data in the dataset that you imported in bulk. You can change this by specifying the job's import mode.

To update bulk data, use the `create-dataset-import-job` command. For the `import-mode`, specify `FULL` to replace existing data or `INCREMENTAL` to add to it. For more information see Import modes.

The following code shows how to create a dataset import job that incrementally imports data into a dataset.

```
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 \
```

```
--import-mode INCREMENTAL
```

Updating bulk records (AWS SDKs)

> **Important**
>
> By default, a dataset import job replaces any existing data in the dataset that you imported in bulk. You can change this by specifying the job's import mode.

To update bulk data, create a dataset import job and specify an import mode. The following code show's how to update bulk data in Amazon Personalize with the SDK for Python (Boto3) or SDK for Java 2.x.

SDK for Python (Boto3)

To update bulk data, use the `create_dataset_import_job` method. For the `import-mode`, specify `FULL` to replace existing data or `INCREMENTAL` to add to it. For more information see Import modes.

```
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',
    importMode = 'INCREMENTAL'
)
```

SDK for Java 2.x

To update bulk data, use the following `createPersonalizeDatasetImportJob` method. For the `importImport`, specify `ImportMode.FULL` to replace existing data or `ImportMode.INCREMENTAL` to add to it. For more information see Import modes.

```
public static String createPersonalizeDatasetImportJob(PersonalizeClient
 personalizeClient,
                                                      String jobName,
```

```
                                                              String datasetArn,
                                                              String s3BucketPath,
                                                              String roleArn,
                                                              ImportMode importMode) {

    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)
                .importMode(importMode)
                .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 "";
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { CreateDatasetImportJobCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// Set the dataset import job parameters.
export const datasetImportJobParam = {
  datasetArn: 'DATASET_ARN',        /* required */
  dataSource: {
    dataLocation: 's3://<name of your S3 bucket>/<folderName>/<CSVfilename>.csv'  /*
 required */
  },
  jobName: 'NAME',                  /* required */
  roleArn: 'ROLE_ARN',              /* required */
  importMode: "INCREMENTAL"         /* optional, default is FULL */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateDatasetImportJobCommand(datasetImportJobParam));
```

```
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Updating data with individual import operations

After you import data into an Amazon Personalize dataset, you can update it by importing additional individual records, including item interactions, action interactions, users, items, or actions. Importing data individually allows you to add small batches of records to your Amazon Personalize datasets as your catalog grows.

When you import records individually, Amazon Personalize appends the new records to the dataset. To update an individual item, user, or action, you can import a record with the same ID but with the modified attributes. You can import up to 10 records per individual import operation.

For more information on importing records individually, see Importing individual records.

Analyzing data in datasets

Important

You can't use the Amazon Personalize console to analyze data in an *Action interactions* or *Actions* dataset.

After you import data into an Item interactions, Users, or Items dataset, you can use the Amazon Personalize console to analyze the data. You can learn about your data through data insights and column and row statistics. And you can learn what actions you can take to improve your data. These actions can help you meet Amazon Personalize resource requirements, such as model training requirements, or they can lead to improved recommendations.

After you make any recommended changes, you can import your data again and see if you resolved any issues or improved dataset statistics. For information on updating data, see Updating data.

If you don't see any insights, your data aligns with Amazon Personalize data expectations. You can analyze data in a Domain dataset group or Custom dataset group.

When generating insights and calculating statistics, Amazon Personalize considers all bulk and streamed data from non-anonymous users. Events from anonymous users aren't considered until you associate them with a `userId`. For more information, see Recording events for anonymous users.

Topics

Required permissions for analyzing data

If you give users full access to Amazon Personalize, no permissions changes are required. If you grant your users only the permissions required to perform a task in Amazon Personalize, your AWS Identity and Access Management (IAM) policy must include the following additional data insight actions.

- personalize:CreateDataInsightsJob
- personalize:ListDataInsightsJob
- personalize:DescribeDataInsightsJob
- personalize:GetDataInsight

Data insights

The following are the possible data insights that you can generate in Amazon Personalize.

Insight	Action	Related dataset(s)
The Interactions dataset has only X interactions. Model training requires	Import Y additional unique interactions records before training a model.	Item interactions

Insight	Action	Related dataset(s)
a minimum of 1,000 interactions. We recommend at least 50,000.		
The Interactions dataset has only X unique users with two or more interactions. Model training requires at least 25 such users. We recommend at least 1,000.	Import at least 2 interactions records each for Y additional users.	Item interactions
X% of items in the Items dataset have no interactions in the Interactions dataset, so they might not be recommended.	Make sure you import all of your interactions data and check for mismatching IDs between your items and interactions datasets. Check the Dataset Statistics below for your items and interactions datasets to make sure you have imported the expected number of rows. If your use case or recipe uses exploration, modify the exploration configuration to recommend more items without interactions data.	Item interactions and Items
X% of users in the Users dataset have no interactions in the Interactions dataset. These users will receive recommendations for popular items.	Make sure you import all of your interactions data and check for mismatching IDs between your users and interactions datasets. Check the Dataset Statistics below for your users and interactions datasets to make sure you have imported the expected number of rows. Import any additional interactions so more users have interactions data.	Item interactions and Users

Insight	Action	Related dataset(s)
The <Users or Items or Interactions> dataset has X% rows with a missing value. This can negatively affect recommendations. We recommend that all required and optional fields be at least 70% percent complete.	Import additional complete records, or import data again without incomplete rows, or import data again with missing values replaced with substitute data, such as the average for numeric columns or the most common value for categorical columns.	Any
The following column(s) in the <datasetType> dataset are less than 70% complete: <ColumnName, ColumnName...>. If this data is included in training, it can negatively affect recommendations. We recommend that columns that allow null values be at least 70% complete.	Import additional complete records, or import data again without incomplete rows, or import data again with missing values replaced with substitute data, such as the average for numeric columns or the most common value for categorical columns.	Any
The following (numerical) column(s) have outliers: <ColumnName, ColumnName...>. Outliers are not always an issue, but sometimes negatively impact recommendations.	Using the Column Statistics below, check if the min and max values for these columns match your expectations. If these values are unexpected, check the data in these columns for inaccuracies and review your data collection and data processing for issues.	Any
The following column(s) have more than 1000 possible categories: <ColumnName, ColumnName...>. If this data is included in training, it can negatively impact recommendations: <ColumnName, ColumnName...>.	Check your categorical data for issues, such as duplicated categories caused by variations in spelling. Resolve any inaccuracies and import data again.	Any

Insight	Action	Related dataset(s)
The following textual metadata column(s) are less than 85% percent complete and will not be used in model training: <ColumnName, ColumnName...>.	Import additional rows or import the rows again with text data for these column(s).	Items
The Interactions dataset has more than 10 unique event types, which will cause model training to fail.	Check your event type column for inaccuracies such as duplicated event types caused by variations in spelling. Remove unnecessary event types and import data again.	Item interactions
The Interactions dataset has the same timestamp for all records. If you use a USER_SEGMENTATION recipe and all records have the same timestamp, model training will fail.	Check your data for timestamp issues and replace duplicated timestamps with unique timestamps.	Item interactions

Viewing dataset insights and statistics

To view insights and statistics on your data in Amazon Personalize datasets, navigate to your datasets in the Amazon Personalize console and choose run analysis.

To view insights and statistics

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. From the navigation pane, under **Datasets** choose **Data analysis**.
4. At the top right, choose **Run analysis**. Amazon Personalize starts analyzing your data. This can take up to 15 minutes. If successful, the results appear on this page.
5. In **Insights**, use the following to filter the insights that appear.

  - To find insights that include specific language, enter your criteria in **Find insight**. As you enter text, the list updates to include only insights with the exact string in the insight or recommended action.
  - To filter the insights by dataset type, change **All datasets** to the specific dataset type. The list updates to include only insights related to this dataset.

6. To view dataset statistics for a dataset, do the following.

  - To view general details and statistics about a dataset, such as the number of rows, unique users and unique items in an Interactions dataset, expand the section for the dataset.
  - To view detailed statistics for a column, expand the dataset section, choose **Column level statistics** and choose the radio button for the column.

7. Correct any issues in your data, import it again, and run another analysis to verify. For more information on importing data again, see Updating data.

Exporting a dataset

> **Note**
>
> You can't export data in an Action interactions dataset or Actions dataset.

After you import your data into an Amazon Personalize dataset, you can export the data to an Amazon S3 bucket. You might export data to verify and inspect the data that Amazon Personalize uses to generate recommendations, view the item interaction events that you previously recorded in real time, or 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 individually (records imported using the console or the `PutEvents`, `PutUsers`, or `PutItems` operations), or both.

For records that match exactly *for all fields*, Amazon Personalize exports just one record. If two records have the same ID but one or more fields are different, Amazon Personalize includes or removes the records depending on data you choose to export:

- If you export both bulk and incremental data, Amazon Personalize exports only the newest items with the same ID (in Items dataset exports), and only users with the same ID (in Users dataset exports). For Item interactions datasets, Amazon Personalize exports all item interactions data.
- If you export incremental data only, Amazon Personalize exports all item, user, or item interaction data that you imported individually, including items or users with the same IDs. Only records that match exactly for all fields are excluded.
- If you export bulk data only, Amazon Personalize includes all item, user, or item interaction data that you imported in bulk, including items or users with the same IDs. Only records that match exactly for all fields are excluded.

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.

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

Topics

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 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.

If you use AWS Key Management Service (AWS KMS) for encryption, you must grant Amazon Personalize and your Amazon Personalize IAM service role permission to use your key. For more information, see Giving Amazon Personalize permission to use your AWS KMS key.

Service role policy for exporting a dataset

The following example policy grants your Amazon Personalize service 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 IAM service role, see Attaching an Amazon S3 policy to your Amazon Personalize service role.

```
{
    "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 role can access and write to your output Amazon S3 bucket. See Dataset export job permissions requirements.

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 role that you created in Creating an IAM role for Amazon Personalize.
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 individually using the console or the `PutEvents`, `PutUsers`, or `PutItems` operations.
    - Choose **Both** to export all of the data in the dataset.
12. For **Tags**, optionally add any tags. For more information about tagging Amazon Personalize resources, see [Tagging Amazon Personalize resources](#).
13. 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 role can access and write to your output Amazon S3 bucket. See [Dataset export job permissions requirements](#).

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 role that you created in [Creating an IAM role for Amazon Personalize](#). 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 individually 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.

```
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 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 role can access and write to your output Amazon S3 bucket. See Dataset export job permissions requirements.

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 role that you created in Creating an IAM role for Amazon Personalize. 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 individually 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 individually 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());
    }
}
```

Deleting data

To delete data in Amazon Personalize, you delete the dataset. Amazon Personalize stores your data in datasets until you delete the datasets. You can't delete a dataset if a dataset import job or solution version is in the `CREATE PENDING` or `IN PROGRESS` state. If you use User-Personalization or Next-Best-Action recipes or *Top picks for you* and *Recommended for you* use cases, deleting a dataset halts automatic updates for any associated solution versions or recommenders.

You can delete a dataset with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs.

Topics

Deleting a dataset (console)

To delete a dataset with the Amazon Personalize console, navigate to the dataset details page and choose delete.

To delete a dataset

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 delete.
6. Choose **Delete** and confirm dataset deletion.

Deleting a dataset (AWS CLI)

The following code shows how to delete a dataset with the AWS CLI and the DeleteDataset operation.

```
aws personalize delete-dataset --dataset-arn dataset-arn
```

Deleting a dataset (AWS SDKs)

The following code shows how to delete a dataset with the AWS SDKs and the DeleteDataset operation.

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.delete_dataset(
    datasetArn = 'dataset ARN'
)
```

SDK for Java 2.x

```
public static void deleteDataset(PersonalizeClient personalizeClient,
                                 String datasetArn) {

    try {
        DeleteDatasetRequest deleteRequest = DeleteDatasetRequest.builder()
                .datasetArn(datasetArn)
                .build();

        int responseCode =
 personalizeClient.deleteDataset(deleteRequest).sdkHttpResponse().statusCode();
        System.out.println(responseCode);
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
}
```

Filtering recommendations and user segments

When getting recommendations with a domain recommender or custom campaign, 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 only items for a particular age group.

Similarly, with USER_SEGMENTATION recipes, you might not want to include certain types of users in user segments. By filtering your results, you can control the items that will be recommended to users or the users that will be included in user segments.

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 real-time recommendations, you apply a filter and specify any filter parameter values when you call the GetRecommendations, GetActionRecommendations, or GetPersonalizedRanking operations. You can also apply a filter when you get recommendations from a campaign or a recommender in the console.

  When you get real-time item recommendations with personalized or related items recipes or use cases, you can specify a promotion in your request. A *promotion* uses a filter to define additional business rules that apply to a configurable subset of recommended items. For more information see Promoting items in recommendations.
- For batch workflows, you include any filter parameter values in your input JSON. Then you specify the filter's Amazon Resource Name (ARN) when you create a batch inference job or batch segment job. For more information see Filtering batch recommendations and user segments (custom resources).

Filter updates for new records

For data that you import with the PutEvents or PutActionInteractions operations, Amazon Personalize updates any filters in the dataset group with the new data within seconds of import. For example, if your filter removes purchased items from recommendations, and you record a purchase event for a user with the PutEvents operation, this item would be removed from future recommendations for this user within seconds of recording the event.

For all other data imported in bulk or individually, Amazon Personalize updates any filters in the dataset group with the new data within 20 minutes from the last import.

Topics

Filter expressions

To configure filters, you must use a properly formatted *filter expression*. Filter expressions are composed of dataset and field identifiers in `dataset.field` format, along with logical operators, keywords, and values. For values, you can specify fixed values or add placeholder parameters set the filter criteria when you get recommendations.

You can use filter expressions to filter items, users, or actions from recommendations based on data from the following datasets:

- **Item interactions**: You can use filter expressions to include or exclude items or users based on interactions data. For example, you can exclude items that a user has clicked (for item recommendations), or include only users who have rated items (for the Item-Affinity recipe). For all recipe types, you can filter only based on event type. You can't filter based on other interaction metadata, such as contextual metadata. You can't use item interactions filters with the Item-Attribute-Affinity recipe.

  Amazon Personalize considers up to 100 of the most recent interactions per user per event type. This is an adjustable quota. You can request a quota increase using the Service Quotas console.
- **Action interactions**: Use filter expressions to include or exclude actions that a user has interacted with based on event type. For example, you might exclude actions that a user has already taken. You can't filter based on other action interaction metadata.

  Amazon Personalize considers up to 300 of the most recent action interactions per user per event type. This is an adjustable quota. You can request a quota increase using the Service Quotas console.
- **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. If your domain use case or custom recipe generates related items recommendations, such as the Similar-Items recipe or the *More Like X* domain use case, you can use filter expressions to include or exclude items based on the properties of the item you specify in your recommendation request.

- **Users**: For *item* and *action* recommendations, if you have a Users dataset, you can exclude or include items or actions based on a `CurrentUser`. For personalized recommendations, popular items, and action recommendations, this is the user you are getting recommendations for. For related items, this an optional user you can specify in your recommendation request.

  For *user segments*, you can use filter expressions to include or exclude users from user segments based on attributes, such as `Users.MEMBERSHIP_STATUS`.
- **Actions**: Use filter expressions to include or exclude actions based on specific action conditions. Amazon Personalize automatically excludes actions based on `Action expiration timestamp` and `Repeat frequency` data. You can't create additional custom filters that filter based on this data.

For a complete list of filter expression elements, see Filter expression elements. For examples of filter expressions, see Filter expression examples.

Topics

Guidelines and requirements

When creating a filter expression, note the following guidelines and requirements:

- You can't use filters to include or exclude items based on unstructured textual item metadata such as product descriptions.
- If you are filtering based on item or action interactions data, you can only filter based on event type. You can't filter based on other interaction metadata, such as contextual metadata.
- Amazon Personalize ignores case only when matching event types.
- You can't use Item Interaction and Item datasets in 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.
- You can't use Item interaction and Action datasets in one expression. To create a filter that filters by Item interaction and then Action datasets (or the opposite), you must chain two or more expressions together. For more information, see Combining multiple expressions.

- You can't use item interactions filters with the Item-Attribute-Affinity recipe.
- 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`.
- The maximum number of distinct dataset fields for a filter, either in one expression or across multiple expressions chained together, is **5**. The maximum number of distinct dataset fields across all filters in a dataset group is **10**.
- You can apply a filter with the CurrentItem element only if your domain use case or custom recipe generates related items recommendations, such as the Similar-Items recipe or the *More Like X* domain use case.
- You can't use placeholder parameters in a filter expression that uses the NOT_IN operator. Instead, use the IN operator and use the opposite Action. For example, use Include instead of Exclude (or the reverse).
- You can't create filters that filter based on `Action expiration timestamp` and `Repeat frequency` data. Amazon Personalize automatically filters action recommendations based on this data.

Filter expression structure and elements

This section includes information about the structure of filter expressions and their elements.

Topics

- Filter expression structure
- Filter expression elements

Filter expression structure

The general structure of a filter expression is as follows:

```
EXCLUDE/INCLUDE ItemID/ActionID/UserID WHERE dataset type.field 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 in the console.

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/ActionID/UserID

Use one of these elements after the `INCLUDE` or `EXCLUDE` element. The element you use depends on whether you are filtering items (for item recommendations), actions (for action recommendations), or users (for user segments).

WHERE

Use `WHERE` to check conditions for items, actions, or users. You must use the `WHERE` element after the `ItemID`, `ActionID`, or `UserID`.

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 fields of the dataset used in the first condition.

Dataset.field

Provide the dataset and the metadata field that you want to filter recommendations by in `dataset.field` format. For example, to filter item recommendations based on the genres field in your Items dataset, you would use Items.genres in your filter expression.

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.attribute

To filter item recommendations based on the user you are getting recommendations for, in *only* an IF condition, use `CurrentUser` and provide the user field. For example, `CurrentUser.AGE`.

CurrentItem.attribute

For only related items recipes and use cases, use `CurrentItem.attribute` to filter items based on an attribute of the item you specify in your request for related items recommendations. For example, `CurrentItem.GENRE` or `CurrentItem.PRICE`.

You can apply a filter with the CurrentItem element only if your domain use case or custom recipe generates related items recommendations, such as the Similar-Items recipe or the *More Like X* domain use case. The first time you create a filter with a `CurrentItem` element, filter creation can a few minutes. If you use AWS KMS for encryption, filter creation can take up to 15 minutes.

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 =, <, <=, >, >=, and != operators to test numerical data, including data passed in a placeholder parameter, for equality.

Asterisk (*) character

Use `*` to include or exclude interactions of all types. Use `*` *only* for filter expressions that use the `EVENT_TYPE` field of an `Interactions` dataset.

Pipe separator

Use the pipe separator (`|`) to chain multiple expressions together. For more information, see [Combining multiple expressions](#).

Parameters

For expressions that use comparison operators or the `IN` operator, 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.

> **Note**
> You define a parameter name when you add it to an expression. The parameter name does not have to match the field name. We recommend that you use a parameter name

that is similar to the field name and easy to remember. You use the parameter name (case sensitive) when you apply the filter to recommendations requests. For an example that shows how to apply a filter with placeholder parameters when using the AWS SDKS, see Applying a filter (AWS SDKs).

Filter expression examples

Use the filter expressions in the following sections to learn how to build your own filter expressions.

Topics

- Item recommendation filter expression examples
- User segment filter expressions
- Action recommendation filter expression examples
- Combining multiple expressions

Item recommendation filter expression examples

The following filter expressions show how to filter item recommendations based on item interactions, item metadata, and user metadata. They are organized by data type.

Topics

- Item interaction data
- Item data
- User data

Item interaction data

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 only items that the user has clicked.

```
INCLUDE ItemID WHERE Interactions.EVENT_TYPE IN ("click")
```

Item data

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 includes only items that are cheaper than the current item (the item you specify in the request for related items recommendations), and created by the same studio as the current item. You can apply a filter with the CurrentItem element only if your domain use case or custom recipe generates related items recommendations.

```
INCLUDE ItemID WHERE Items.PRICE < CurrentItem.PRICE AND Items.GENRE IN
 CurrentItem.GENRE
```

The following expression excludes items based on multiple levels of categorical fields. It excludes items with a CATEGORY_L1 value of `shoe` that *do not* have a CATEGORY_L2 value of `boot`.

```
EXCLUDE ItemID WHERE Items.CATEGORY_L1 IN ("shoe") AND Items.CATEGORY_L2 NOT IN
 ("boot")
```

The following expression includes only items with a price less than or equal to the price that you specify when you get recommendations using the `$PRICE` parameter.

```
INCLUDE ItemID WHERE Items.PRICE <= $PRICE
```

The following expression includes only items that have been created earlier than a timestamp (in Unix epoch time) that you specify when you get recommendations.

```
INCLUDE ItemID WHERE Items.CREATION_TIMESTAMP < $DATE
```

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)
```

The following expression includes only items that are more expensive than the current item *and* created more recently than a timestamp (in Unix epoch time) that you specify. You might use this filter if you are getting related item recommendations, and want to apply some specific business rules based on price and a varying creation date.

```
INCLUDE ItemID WHERE Items.PRICE < CurrentItem.PRICE AND Items.CREATION_TIMESTAMP >
 $DATE
```

User data

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 with `watch` for CATEGORY_L1 and `luxury` for CATEGORY_L2, if the current user's age is over 18.

```
INCLUDE ItemID WHERE Items.CATEGORY_L1 IN ("watch") AND Items.CATEGORY_L2 IN ("luxury")
 IF CurrentUser.AGE > 18
```

User segment filter expressions

The following filter expressions show how to filter user segments based on item interactions data and user metadata. They are organized by data type.

User data

The following filter expression includes only users with a membership status equal to the value that you specify when you get user segments.

```
INCLUDE UserID WHERE Users.MEMBERSHIP_STATUS IN ($MEMBERSHIP)
```

The following filter expression excludes users with an AGE less than a value you specify when you get user segments.

```
EXCLUDE UserID WHERE Users.AGE < $AGE
```

Item interaction data

The following filter expression includes only users who have clicked or rated items.

```
INCLUDE UserID WHERE Interactions.EVENT_TYPE IN ("click", "rating")
```

The following filter expression excludes users from user segments who have item interactions with an event type you specify when you get user segments.

```
EXCLUDE UserID WHERE Interactions.EVENT_TYPE IN ($EVENT_TYPE)
```

Action recommendation filter expression examples

The following filter expression examples show how to filter actions based on action interactions data, action data. and user data. They are organized by data type.

Topics

Action interaction data

The following filter expression includes only actions in recommendations that the user has interacted with, when those interactions have an event type that you specify when you get recommendations.

```
INCLUDE ActionID WHERE Action_Interactions.EVENT_TYPE IN ($EVENT_TYPE)
```

The following filter expression excludes actions that the user has not taken based on event type.

```
EXCLUDE ActionID WHERE Action_Interactions.EVENT_TYPE IN ("NOT_TAKEN")
```

Action data

The following expression excludes actions based on a category or categories that you specify when you get recommendations using the `$CATEGORY` parameter.

```
EXCLUDE ActionID WHERE Actions.CATEGORY IN ($CATEGORY)
```

The following expression includes only actions with a value greater than a value that you specify when you get recommendations.

```
INCLUDE ActionID WHERE Actions.VALUE > ($VALUE)
```

User data

The following expression includes only actions for premium members if the current user has a premium membership.

```
INCLUDE ActionID WHERE Action.MEMBERSHIP_LEVEL IN ("Premium") IF CurrentUser.MEMBERSHIP
 = $PREMIUM
```

The following expression excludes actions with a `VALUE` less than a value that you specify when you get recommendations if the current user is a premium member.

```
EXCLUDE ActionID WHERE Actions.VALUE < ($VALUE) IF CurrentUser.MEMBERSHIP = $PREMIUM
```

Combining multiple expressions

To combine multiple expressions together you use a pipe separator (`|`). Use a combination of expressions when you want to use a single filter and filter on Items and Item interactions datasets, or Action and Action interactions datasets. Each expression is first evaluated independently and the result is either the union or the intersection of the two results. The following examples show how to create expressions for Items and Item interactions datasets, but the same rules apply when working with Actions and Action interactions.

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)`

Expression order does not matter: If the EXCLUDE expression comes before the INCLUDE expression, the result is the same.

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.

When you get personalized item recommendations or similar items, you can specify a promotion in your request. A *promotion* uses a filter to define additional business rules that apply to a configurable subset of recommended items. For more information see [Promoting items in recommendations](#).

Topics

- [Filtering real-time recommendations (console)](#)
- [Filtering real-time recommendations (AWS CLI)](#)
- [Filtering real-time recommendations (AWS SDKs)](#)

Filtering real-time recommendations (console)

To filter real-time recommendations using the console, create a filter and then apply it to a recommendation request.

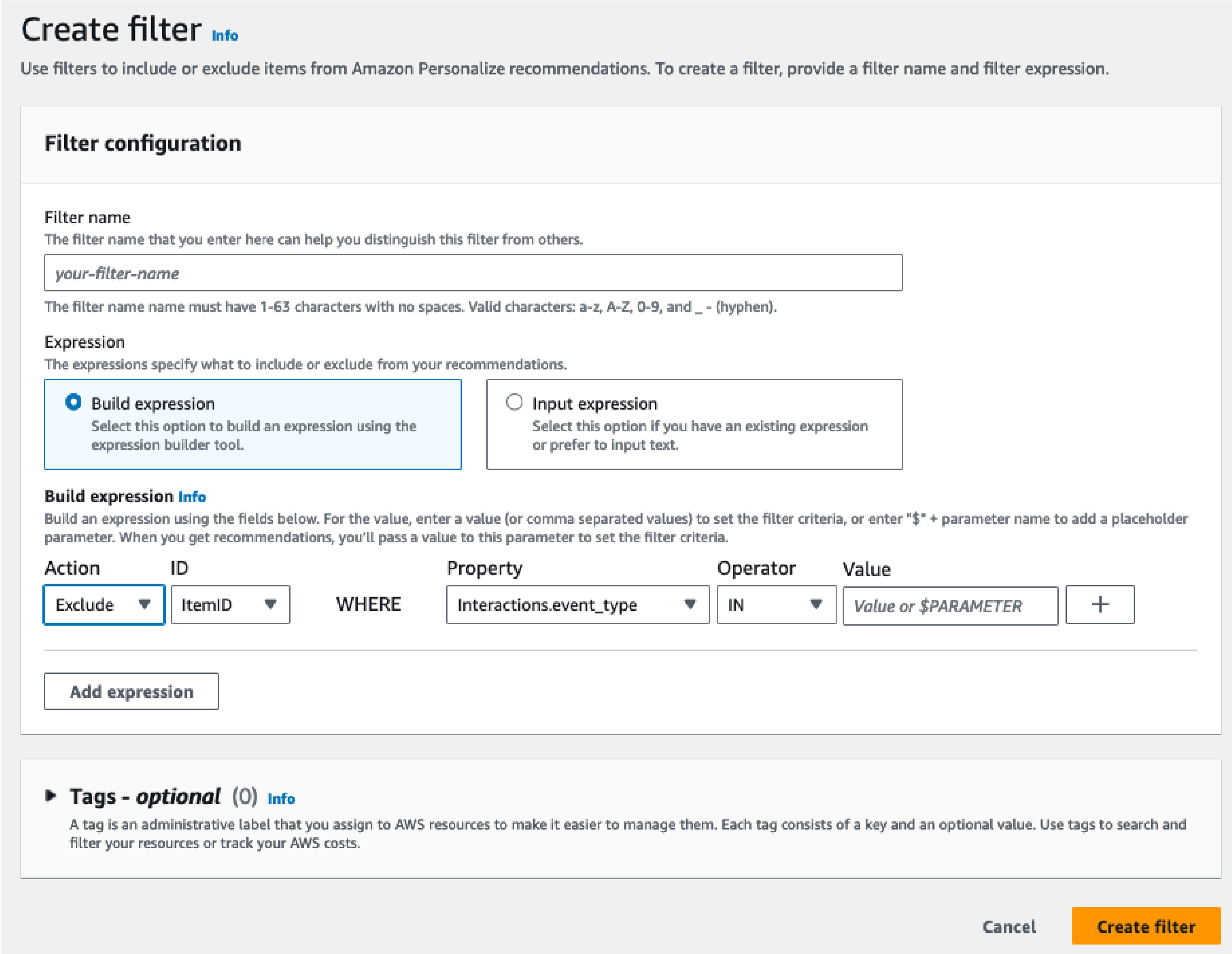
> **Note**
>
> To filter recommendations using a filter with parameters and a campaign deployed before November 10, 2020, you must redeploy the campaign by using the [UpdateCampaign](#) operation or create a new campaign.

Creating a filter (console)

To create a filter in the console, choose the dataset group that contains the campaign or recommender you want to use to get filtered recommendations. 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 or recommender that you want to use to get filtered recommendations.
3. In the navigation pane, choose **Filters** and then choose **Create new filter**. The **Create filter** page displays.



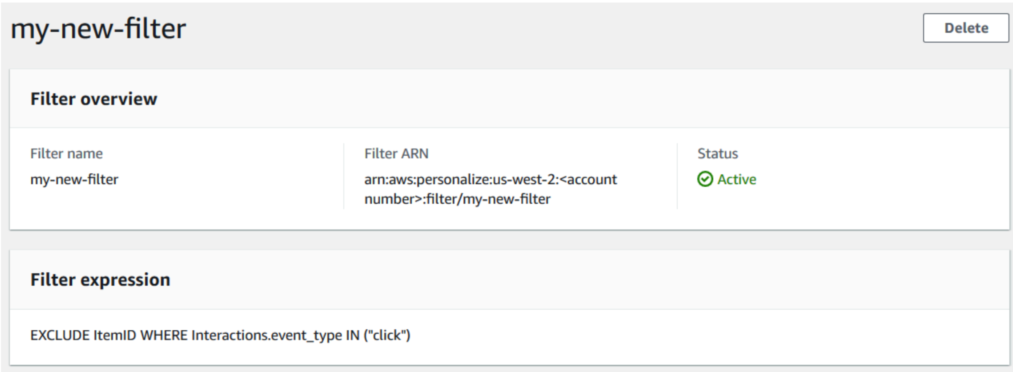
4. For **Filter name**, enter a name for your filter. You will choose the filter by this name when you apply it to a recommendation request.

5. For **Expression**, choose either **Build expression** or **Add expression manually** and build or insert your expression:

   - To use the expression builder, choose **Build expression**. The expression builder provides structure, fields, and guidelines for building correctly formatted filter expressions. For more information, see Using the filter expression builder.
   - To input your own expression, choose **Add expression manually**. For more information, see Filter expression elements.

6. Choose **Finish**. The filter's overview page shows the filter’s Amazon Resource Name (ARN), status, and full filter expression. To delete the filter, choose **Delete**. For information about

finding and deleting filters after you have left the overview page, see [Deleting a filter (console)](#).



Applying a filter (console)

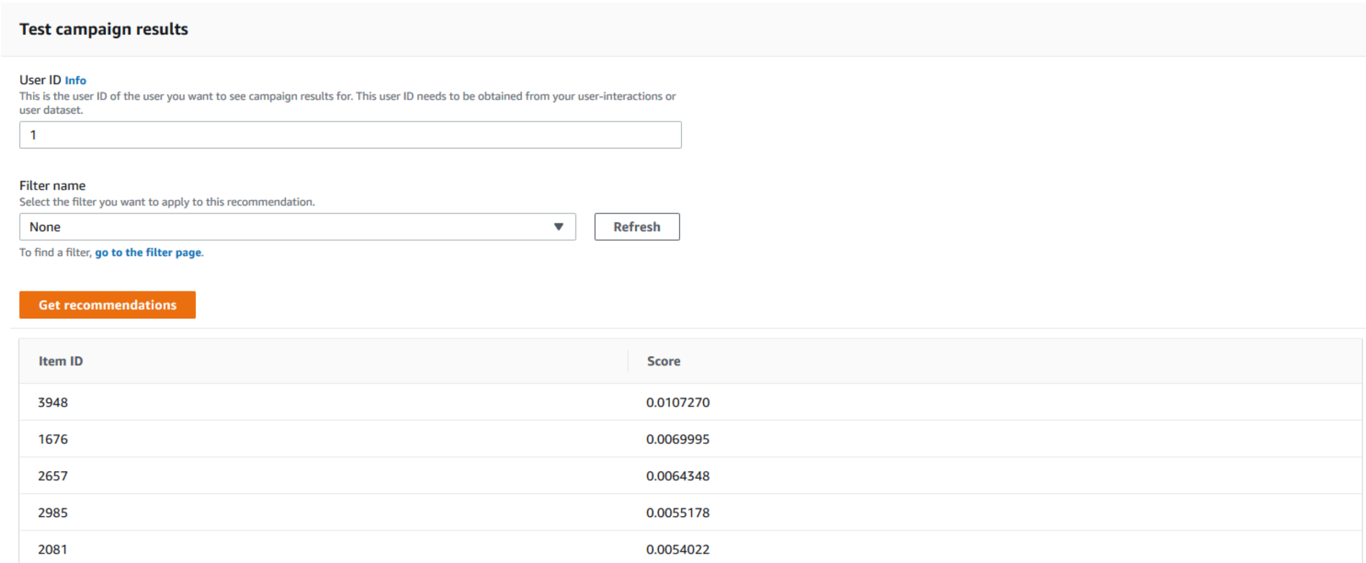
To apply a filter, in **Test recommender** (for recommenders) or **Test campaign results** (for custom campaigns), choose the filter and enter any filter parameter values. Then get recommendations for a user.

> **Important**
>
> For filter expressions that use an `INCLUDE` element, you must provide values for all parameters that are defined in the expression. For filters with expressions that use an `EXCLUDE` element, you can omit the `filter-values`. In this case, Amazon Personalize doesn't use that portion of the expression to filter recommendations.

To apply a filter (console)

1. 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 or recommender that you want to use to get filtered recommendations.
3. Depending on your dataset group type or resource type, do either of the following:
   a. For a Domain dataset group, in the navigation pane choose **Recommenders**.
   b. For a Custom dataset group or custom resources, in the navigation pane choose **Custom resources** then **Campaigns**.
4. On the **Recommenders** or **Campaigns** page, choose the target recommender or campaign.

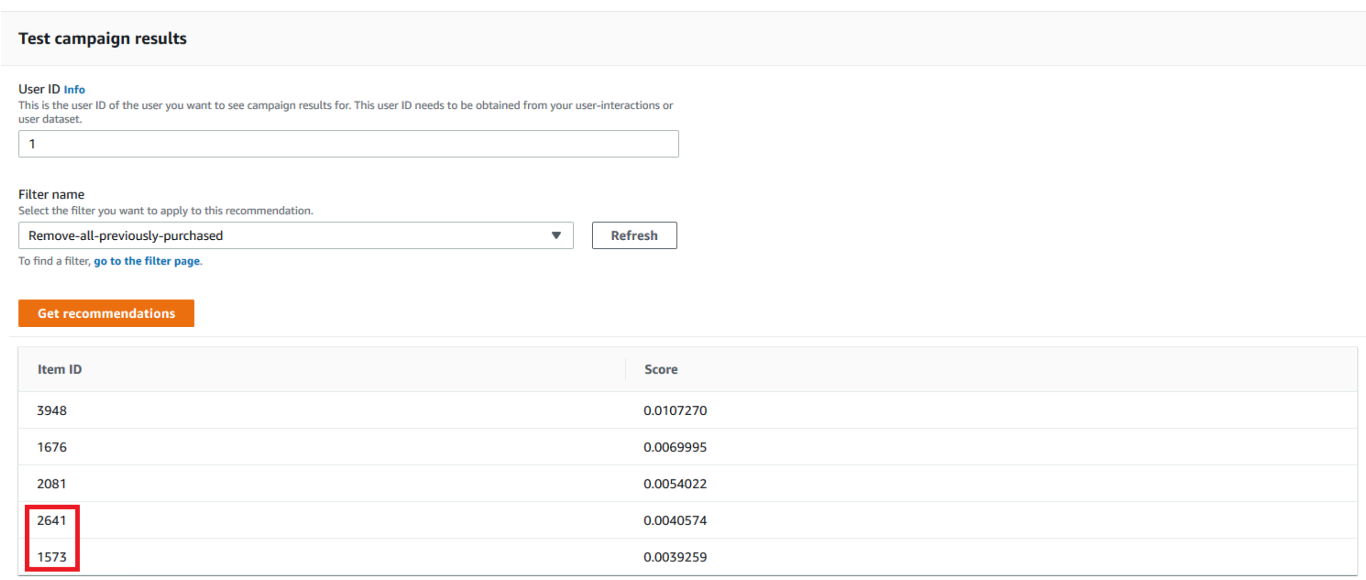
5. For comparison, start by getting recommendations without applying a filter. Under **Test recommender / Test campaign results**, enter the ID of a user that you want to get recommendations for, or the ID of the item for related items, and choose **Get recommendations**. A table containing the top recommendations appears.



6. 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.

7. 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.

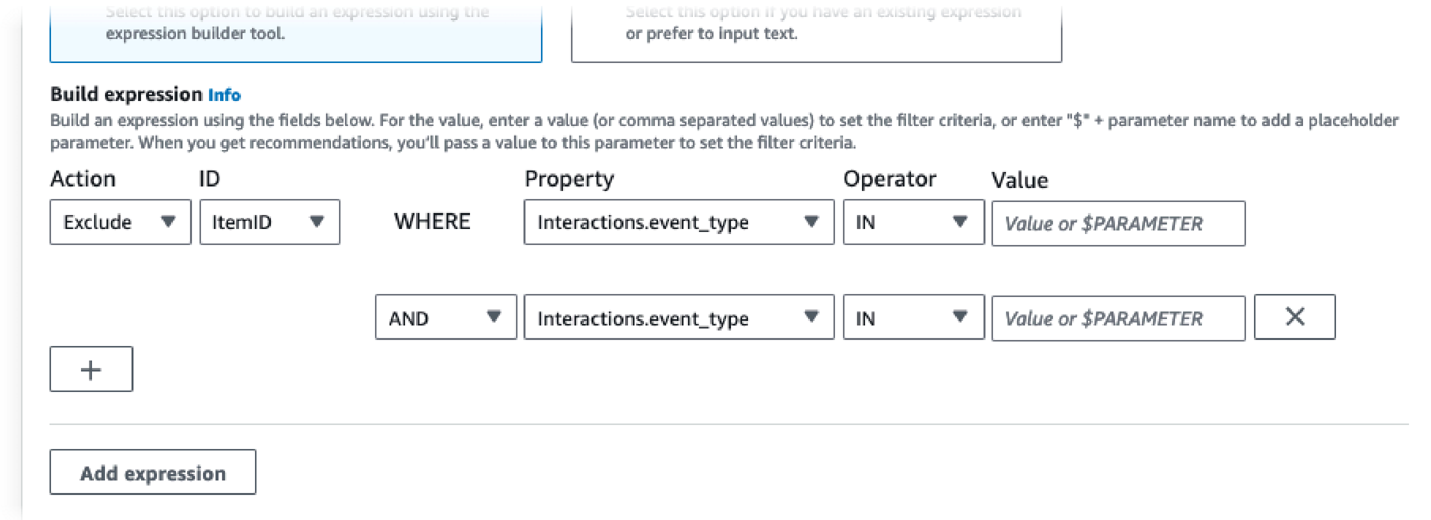
8. Using the same `User ID` or `Item ID` as in the earlier step, choose **Get recommendations**. The recommendations table appears.



For example, 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 didn't buy (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



To build a filter expression:

- Use the **Type**, **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.field` 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.

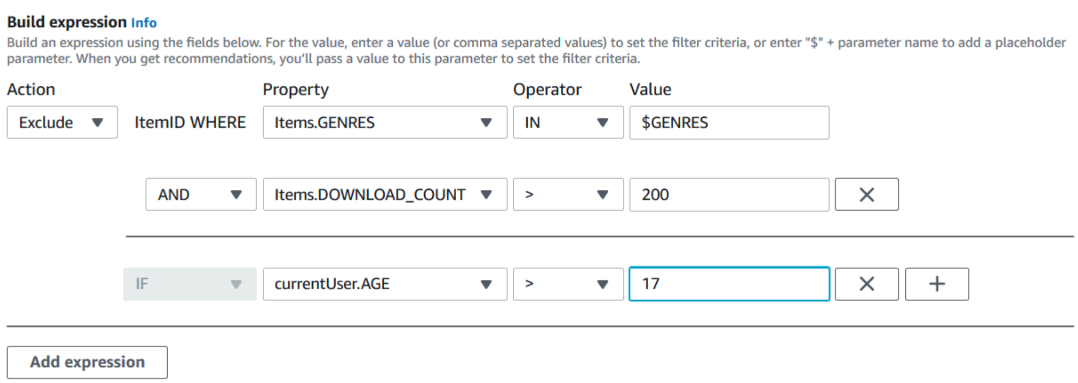
  For `IF` conditions:

  - 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. 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 Item interaction datasets, or Action and Action interactions 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). The filter also excludes items with a `DOWNLOAD_COUNT` of more than `200`, but only if the current user's age is greater than 17.



Deleting a filter (console)

Deleting a filter removes the filter from the list of filters for a dataset group.

Important

You can't delete a filter while a batch inference job is in progress.

To delete a filter (console)

1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/home and sign into your account.
2. From the **Dataset groups** list, choose the dataset group that contains the filter that you want to delete.
3. In the navigation pane, choose **Filters**.
4. From the list of filters, choose the filter that you want to delete and choose **View Details**. The filter details page appears.
5. Choose **Delete** and confirm the deletion in the confirmation dialog box.

Filtering real-time recommendations (AWS CLI)

To filter recommendations using the AWS CLI, you create a filter and then apply it by specifying the filter ARN in a GetRecommendations or GetPersonalizedRanking 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 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 operation before you use the filter.

For more information about the API, see CreateFilter. For more information about filter expressions, including examples, see Filter expression structure and elements.

Applying a filter (AWS CLI)

When you use the `get-recommendations`, `get-action-recommendations` or `get-personalized-ranking` operations, you 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 you're getting recommendations from a recommender instead of a campaign, use `recommender-arn` instead of `--campaign-arn` and provide the ARN for the recommender.

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 or GetPersonalizedRanking 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 call or create a new campaign.

Creating a filter (AWS SDKs)

Create a new filter with the CreateFilter operation. The following code shows how to create a filter. Specify the filter name, Amazon Resource Name (ARN) of your dataset group, and provide your filter expression.

SDK for Python (Boto3)

```
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)
```

SDK for Java 2.x

```
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 "";
}
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { CreateFilterCommand } from
  "@aws-sdk/client-personalize";
import { personalizeClient } from "./libs/personalizeClients.js";
// Or, create the client here.
// const personalizeClient = new PersonalizeClient({ region: "REGION"});

// Set the filter's parameters.
export const createFilterParam = {
  datasetGroupArn: 'DATASET_GROUP_ARN', /* required */
  name: 'NAME', /* required */
  filterExpression: 'FILTER_EXPRESSION' /*required */
}
```

```
export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateFilterCommand(createFilterParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Record the filter ARN for later use. To verify that the filter is active, use the [DescribeFilter](#) operation before using the filter. For more information about the API, see [CreateFilter](#). For more information about filter expressions, including examples, see [Filter expression structure and elements](#).

Applying a filter (AWS SDKs)

When you use the GetRecommendations, GetActionRecommendations, or GetPersonalizedRanking operations, apply a filter by passing a `filterArn` and any filter values as parameters.

The following code shows how to get filtered Amazon Personalize item recommendations for a user. Specify the ID of the user you want to get recommendations for, the Amazon Resource Name (ARN) of your campaign, and the ARN of your filter. If you're getting recommendations from a recommender instead of a campaign, use `recommenderArn` instead of `campaignArn` and provide the ARN for the recommender.

For `filterValues`, for each optional parameter in your filter expression, provide the parameter name (case sensitive) and the value or values. For example, if your filter expression has a `$GENRES` parameter, provide *"GENRES"* as the key, and a genre or genres, such as `"\"Comedy"\"`, as the value. For multiple values, separate each value with a comma. For example, `"\"comedy\",\"drama\",\"horror\""`.

Important

For filter expressions that use an `INCLUDE` element to include items, you must provide values for all parameters that are defined in the expression. For filters with expressions that use an `EXCLUDE` element to exclude items, you can omit the `filter-values`. In this case, Amazon Personalize doesn't use that portion of the expression to filter recommendations.

SDK for Python (Boto3)

```
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

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);
    }
}
```

SDK for JavaScript v3

```
// Get service clients module and commands using ES6 syntax.
import { GetRecommendationsCommand } from
  "@aws-sdk/client-personalize-runtime";
import { personalizeRuntimeClient } from "./libs/personalizeClients.js";
// Or, create the client here:
// const personalizeRuntimeClient = new PersonalizeRuntimeClient({ region:
 "REGION"});

// Set recommendation request parameters.
export const getRecommendationsParam = {
  campaignArn: 'CAMPAIGN_ARN', /* required */
  userId: 'USER_ID',      /* required */
  numResults: 15,    /* optional */
  filterArn: 'FILTER_ARN',   /* required to filter recommendations */
  filterValues: {
    "PROPERTY": "\"VALUE\""  /* Only required if your filter has a placeholder
 parameter */
  }
```

```
}

export const run = async () => {
  try {
    const response = await personalizeRuntimeClient.send(new
 GetRecommendationsCommand(getRecommendationsParam));
    console.log("Success!", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

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 and user segments (custom resources)

Filtering batch recommendations and user segments works nearly the same as filtering real-time recommendations. It follows the same workflow described in Batch recommendations and user segments (custom resources). To filter batch recommendations or user segments, you do the following:

1. Create a filter just like you would for real-time recommendations. For more information see Filtering real-time recommendations.
2. Prepare your input data and upload it to Amazon S3 as described in Preparing input data for batch recommendations or Preparing input data for user segments. If your filter uses

placeholder parameters, you must add an additional `filterValues` object. For more information, see Providing filter values in your input JSON. If your filter doesn't use placeholder parameters, your input data can follow the examples in Batch inference job input and output JSON examples Batch segment job input and output JSON examples

3. Create a separate location for your output data, either a folder or a different Amazon S3 bucket.
4. Create a batch inference job or a batch segment job. When you create the job, specify the Amazon Resource Name (ARN) of your filter.
5. When the batch inference or batch segment job is complete, retrieve the recommendations or user segments from your output location in Amazon S3.

Topics

- Providing filter values in your input JSON
- Filtering batch workflows (console)
- Filtering batch workflows (AWS SDKs)

Providing filter values in your input JSON

For filters with placeholder parameters, such as `$GENRE`, you must provide the values for the parameters in a `filterValues` object in your input JSON. For a `filterValues` object, each key is a parameter name. Each value is the criteria that you are passing as a parameter. Surround each value with escaped quotes: `"filterValues":{"GENRES":"\"drama\""}`. For multiple values, separate each value with a comma: `"filterValues":{"GENRES":"\"horror\",\"comedy\",\"drama\""}`

Batch inference job input JSON example

The following is an example of the first few lines of a JSON input file for a *batch inference job*. The example includes the `filterValues` object. The `GENRES` key corresponds to a `$GENRES` placeholder in the filter expression. The job in this example uses the User-Personalization recipe. For RELATED_ITEMS recipes, provide an itemId instead of the userId. For PERSONALIZED_RANKING recipes provide the userID and an itemList.

```
{"userId": "5","filterValues":{"GENRES":"\"horror\",\"comedy\",\"drama\""}}
{"userId": "3","filterValues":{"GENRES":"\"horror\",\"comedy\""}}
{"userId": "34","filterValues":{"GENRES":"\"drama\""}}
```

For more examples of batch inference job input data by recipe see Batch inference job input and output JSON examples. You can use these examples as a starting point and add the `filterValues` object from the above example.

Batch segment job input JSON example

The following is an example of the first few lines of a JSON input file with filter values for a *batch segment job*. The `GENRES` key corresponds to a `$GENRES` placeholder in the filter expression.

```
{"itemAttributes": "ITEMS.genres = \"Comedy\" AND ITEMS.genres = \"Action
\"","filterValues":{"COUNTRY":"\"Japan\""}}
{"itemAttributes": "ITEMS.genres = \"Horror\"","filterValues":{"COUNTRY":"\"United
 States\"\""}}
{"itemAttributes": "ITEMS.genres = \"Action\" AND ITEMS.genres = \"Adventure
\"","filterValues":{"COUNTRY":"\"England\""}}
```

For more examples of batch inference job input data by recipe see Batch segment job input and output JSON examples. You can use these examples as a starting point and add the `filterValues` object from the above example.

Filtering batch workflows (console)

To filter batch workflows with the Amazon Personalize console, you create a filter and then you create a batch inference job or batch segment job and choose the filter. For complete step by step instructions, see Creating a batch inference job (console) and Creating a batch segment job (console).

Filtering batch workflows (AWS SDKs)

To filter batch recommendations with the AWS SDKs, create a filter and include the `FilterArn` parameter in the CreateBatchInferenceJob or CreateBatchSegmentJob request.

The following code shows how to create a batch inference job with a filter using the AWS SDK for Python (Boto3). We recommend using a different location for your output data (either a folder or a different Amazon S3 bucket). For complete explanation of all fields, see see Creating a batch inference job (AWS SDKs).

```
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 = e
        {"S3DataDestination": {"path": "S3 output path"}}
)
```

Measuring impact of recommendations

As your customers interact with recommendations, you can measure how the recommendations are helping you achieve your goals. You can identify which campaigns and recommenders have the most impact, such as which resource generates the most minutes watched or the most clicks. And you can compare the performance of Amazon Personalize recommendations to those generated by third-party services.

The following can help you measure the impact of recommendations:

- Metric attribution: An Amazon Personalize metric attribution creates reports based on metrics that you specify and the item interactions and items data that you import. For example, the total length of movies watched by users, or the total number of click events.
- A/B testing: Performing an A/B test consists of running an experiment with multiple variations and comparing the results. You can use A/B testing to help compare and evaluate different recommendation strategies, and measure the impact of the recommendations.

Topics

Measuring recommendation impact with a metric attribution

To measure the impact of item recommendations, you can create a metric attribution. A *metric attribution* creates reports based on the item interactions and items data that you import, and the metrics that you specify. For example, the total length of movies watched by users, or the total number of click events. Amazon Personalize aggregates calculations over a 15-minute window. For PutEvents and incremental bulk data, Amazon Personalize automatically sends metric reports to Amazon CloudWatch. For bulk data, you can choose to publish reports to an Amazon S3 bucket.

For each interaction that you import, include source data to compare different campaigns, recommenders, and third parties. You can include the recommendation ID of the recommendations you showed the user or the event source, such as a third party.

For example, you might have a video streaming app that shows movie recommendations from two different Amazon Personalize recommenders. If you wanted to see which recommender generates the most watch events, you could create a metric attribution that tracks the total number of watch events. Then you could record watch events as users interact with recommendations, and include the `recommendationId` in each event. Amazon Personalize uses the `recommendationId` to identify each recommender. As you record events, you can view the watch event totals aggregated over every 15 minutes for both recommenders in CloudWatch. For code samples that show how to include a `recommendationId` or an `eventAttributionSource` for an event, see Event metrics and attribution reports.

Topics

- Guidelines and requirements
- Creating a metric attribution
- Managing a metric attribution
- Publishing and viewing results

Guidelines and requirements

Amazon Personalize starts calculating and reporting the impact of recommendations only after you create a metric attribution. To build the most complete history, we recommend creating a metric attribution before you import your interactions data. When you create a dataset import job for an Item interactions dataset with the Amazon Personalize console, you have the option to create a metric attribution in a new tab. Then you can return to the import job to complete it.

After you create a metric attribution and record events or import incremental bulk data, you will incur some monthly CloudWatch cost per metric. For information about CloudWatch pricing, see the Amazon CloudWatch pricing page. To stop sending metrics to CloudWatch, delete the metric attribution.

To see the impact of recommendations over time, keep importing data as customers interact with recommendations. If you have already imported data, you can still create a metric attribution and start measuring recommendation impact. However, Amazon Personalize won't report on data that you imported before you created it.

The following are guidelines and requirements for generating reports with a metric attribution:

- You must grant Amazon Personalize permission to access and put data in CloudWatch. For policy examples, see Giving Amazon Personalize access to CloudWatch.

- To publish metrics to Amazon S3, give Amazon Personalize permission to write to your bucket. You also must provide the bucket path in your metric attribution. For policy examples, see Giving Amazon Personalize access to your Amazon S3 bucket.
- To publish metrics to CloudWatch, records must be less than 14 days old. If your data is older, these records won't be included in calculations or reports.
- Importing duplicate events (events that match for all attributes exactly) can lead to unexpected behavior including inaccurate metrics. We recommend that you remove duplicate records from any bulk data before import, and avoid importing duplicate events with the `PutEvents` operation.
- Your Item interactions dataset must have an `EVENT_TYPE` column.
- You can't create metric reports for data in a Action interactions dataset.
- You can create at most one metric attribution per dataset group. Each metric attribution can have at most 10 metrics.

To compare sources, each interaction event must include a `recommendationId` or `eventAttributionSource`. You can provide at most 100 unique event attribution sources. For `PutEvents` code samples, see Event metrics and attribution reports.

- If you provide a `recommendationId`, Amazon Personalize automatically determines the source campaign or recommender and identifies it in reports in an EVENT_ATTRIBUTION_SOURCE column.
- If you provide both attributes, Amazon Personalize uses only the `eventAttributionSource`.
- If you don't provide a source, Amazon Personalize labels the source `SOURCE_NAME_UNDEFINED` in reports.

Topics

Giving Amazon Personalize access to CloudWatch

> **Important**
>
> When you grant permissions, Amazon Personalize places and validates a small amount of data in CloudWatch. This will incur a one-time cost of less than $0.30. For more information about CloudWatch pricing, see the [Amazon CloudWatch pricing](#) page.

To give Amazon Personalize access to CloudWatch, attach a new AWS Identity and Access Management (IAM) policy to your Amazon Personalize service role that grants the role permission to use the `PutMetricData` Action for CloudWatch. The following policy example grants `PutMetricData` permissions.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "*"
    }
  ]
}
```

Giving Amazon Personalize access to your Amazon S3 bucket

To give Amazon Personalize access to your Amazon S3 bucket:

- Attach an IAM policy to your Amazon Personalize service role that grants the role permission to use the `PutObject` Action on your bucket.

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

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

- Attach a bucket policy to your output Amazon S3 bucket that grants the Amazon Personalize principle permission to use the `PutObject` Actions.

  If you use AWS Key Management Service (AWS KMS) for encryption, you must grant Amazon Personalize and your Amazon Personalize IAM service role permission to use your key. For more information, see Giving Amazon Personalize permission to use your AWS KMS key.

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

Creating a metric attribution

> **Important**
>
> After you create a metric attribution and record events or import incremental bulk data, you will incur some monthly CloudWatch cost per metric. For information about CloudWatch pricing, see the Amazon CloudWatch pricing page. To stop sending metrics to CloudWatch, delete the metric attribution.

To start generating metric reports, you create a metric attribution and import interactions data. When you create a metric attribution, you specify a list of event types to report on. For each event type, you specify a function that Amazon Personalize applies as it collects the data. Available functions include `SUM(DatasetType.COLUMN_NAME)` and `SAMPLECOUNT()`.

For example, you might have an online video streaming app and want to track two metrics: the click-through rate for recommendations, and the total length of movies watched, where each video in the Items dataset includes a `LENGTH` attribute. You would create a metric attribution and add two metrics, each with an event type and function. The first might be for the `Click` event type with a `SAMPLECOUNT()` function. The second might be for the `Watch` event type with a `SUM(Items.LENGTH)` function.

You can apply `SUM()` functions to only numeric columns of Items and Item interactions datasets. To apply a `SUM()` function to a column in an Items dataset, you must first import item metadata.

You can create a metric attribution with the Amazon Personalize console, AWS Command Line Interface, or AWS SDKS.

Topics

Creating a metric attribution (console)

To create a metric attribution with the Amazon Personalize console, you navigate to the **Metric attribution** page and choose **Create metric attribution**. When you create a metric attribution, you specify an optional Amazon S3 bucket path, your Amazon Personalize IAM service role, and a list of metrics to report on.

When you create an Item interactions dataset import job with the Amazon Personalize console, you have the option to create a metric attribution in a new tab. Then you can return to the import job to complete it. If you're already on the **Configure metric attribution** page, you can skip to step 4.

To create a metric attribution

1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/home and sign into your account.
2. Choose your dataset group.
3. In the navigation pane, under **Custom resources**, choose **Metric attribution**.
4. In **Metric attribution details**, choose **Create metric attribution**.
5. On the **Configure metric attribution** page, give the metric attribution a name.
6. If you want to publish metrics to Amazon S3 for **Amazon S3 data output path**, enter the destination Amazon S3 bucket. This enables the option to publish metrics each time you create a dataset import job. Use the following syntax:

   ```
   s3://<name of your S3 bucket>/<folder> path>
   ```
7. If you are using AWS KMS for encryption, for **KMS key ARN**, enter the Amazon Resource Name (ARN) for the AWS KMS key. You must grant Amazon Personalize and your Amazon Personalize IAM service role permission to use your key. For more information, see Giving Amazon Personalize permission to use your AWS KMS key.
8. In **IAM role**, choose to create a new service role or use an existing one. The role you choose must have `PutMetricData` permissions for CloudWatch. If you want to publish to Amazon S3, the role must have `PutObject` permissions for your Amazon S3 bucket.

   To use the role that you created in Creating an IAM role for Amazon Personalize, you might have to add policies for CloudWatch and Amazon S3.

   For policy examples, see Giving Amazon Personalize access to CloudWatch and Giving Amazon Personalize access to your Amazon S3 bucket.
9. Choose **Next**.
10. On the **Define metric attributes** page, choose how to define metrics. Choose **Build metric attributes** to use the builder tool. Choose **Input metric attributes** to enter metrics in JSON format.

- If you choose **Build metric attributes**, for each metric provide a name, event type, and choose a function. For `SUM()` functions, choose the column name. Choose **Add metric attribute** to add additional metrics.
- If you choose **Input metric attributes**, enter each metric in JSON format. The following shows how to format a metric.

```
{
    "EventType": "watch",
    "MetricName": "MinutesWatchedTracker",
    "MetricMathExpression": "SUM(Items.LENGTH)"
}
```

11. Choose **Next**.
12. On the **Review and create page**, review the details for the new metric attribution. To make changes, choose **Previous**. To create the metric attribution, choose **Create**. When the metric attribution is active, you can start importing data and view the results. For information on viewing results, see Publishing and viewing results.

Creating a metric attribution (AWS CLI)

The following code shows how to create a metric attribution with the AWS Command Line Interface. The role you specify must have `PutMetricData` permissions for CloudWatch and, if publishing to Amazon S3, `PutObject` permissions for your Amazon S3 bucket. To use the role that you created in Creating an IAM role for Amazon Personalize, you might have to add policies for CloudWatch and Amazon S3. For policy examples, see Giving Amazon Personalize access to CloudWatch and Giving Amazon Personalize access to your Amazon S3 bucket.

For each metric specify a name, event type, and expression (a function). Available functions include `SUM(DatasetType.COLUMN_NAME)` and `SAMPLECOUNT()`. For SUM() functions, specify the dataset type and column name. For example, `SUM(Items.LENGTH)`. For information on each parameter, see CreateMetricAttribution.

```
aws personalize create-metric-attribution \
--name metric attribution name \
--dataset-group-arn dataset group arn \
--metrics-output-config "{\"roleArn\": \"Amazon Personalize service role ARN\",
 \"s3DataDestination\":{\"kmsKeyArn\":\"kms key ARN\",\"path\":\"s3://bucket-
name/folder-name/\"}}" \
```

```
--metrics "[{
  \"eventType\": \"event type\",
  \"expression\": \"SUM(DatasetType.COLUMN_NAME)\",
  \"metricName\": \"metric name\"
}]"
```

Creating a metric attribution (AWS SDKs)

The following code shows how to create a metric attribution with the SDK for Python (Boto3). The role you specify must have `PutMetricData` permissions for CloudWatch and, if publishing to Amazon S3, `PutObject` permissions for your Amazon S3 bucket. To use the role that you created in Creating an IAM role for Amazon Personalize, you might have to add policies for CloudWatch and Amazon S3. For policy examples, see Giving Amazon Personalize access to CloudWatch and Giving Amazon Personalize access to your Amazon S3 bucket.

For each metric specify a name, event type, and expression (a function). Available functions include `SUM(DatasetType.COLUMN_NAME)` and `SAMPLECOUNT()`. For SUM() functions, specify the dataset type and column name. For example, `SUM(Items.LENGTH)`. For information on each parameter, see CreateMetricAttribution.

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

metricsList = [{
      "eventType": "event type",
      "expression": "SUM(DatasetType.COLUMN_NAME)",
      "metricName": "metric name"
}]

outputConfig = {
  "roleArn": "Amazon Personalize service role ARN",
  "s3DataDestination": {
    "kmsKeyArn": "key ARN",
    "path": "s3://<name of your S3 bucket>/<folder>"
  }
}
response = personalize.create_metric_attribution(
  name = 'metric attribution name',
  datasetGroupArn = 'dataset group arn',
```

```
  metricsOutputConfig = outputConfig,
  metrics = metricsList
)

metric_attribution_arn = response['metricAttributionArn']

print ('Metric attribution ARN: ' + metric_attribution_arn)

description = personalize.describe_metric_attribution(
    metricAttributionArn = metric_attribution_arn)['metricAttribution']

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

SDK for Java 2.x

```
public static String createMetricAttribution(PersonalizeClient personalizeClient,
                                             String eventType,
                                             String expression,
                                             String metricName,
                                             String metricAttributionName,
                                             String roleArn,
                                             String s3Path,
                                             String kmsKeyArn,
                                             String datasetGroupArn) {
    String metricAttributionArn = "";

    try {

        MetricAttribute attribute = MetricAttribute.builder()
                .eventType(eventType)
                .expression(expression)
                .metricName(metricName)
                .build();

        ArrayList<MetricAttribute> metricAttributes = new ArrayList<>();
        metricAttributes.add(attribute);

        S3DataConfig s3DataDestination = S3DataConfig.builder()
                .kmsKeyArn(kmsKeyArn)
                .path(s3Path)
                .build();
```

```
        MetricAttributionOutput outputConfig = MetricAttributionOutput.builder()
                .roleArn(roleArn)
                .s3DataDestination(s3DataDestination)
                .build();

        CreateMetricAttributionRequest createMetricAttributionRequest =
 CreateMetricAttributionRequest.builder()
                .name(metricAttributionName)
                .datasetGroupArn(datasetGroupArn)
                .metrics(metricAttributes)
                .metricsOutputConfig(outputConfig)
                .build();
        CreateMetricAttributionResponse createMetricAttributionResponse =
 personalizeClient.createMetricAttribution(createMetricAttributionRequest);

        metricAttributionArn =
 createMetricAttributionResponse.metricAttributionArn();
        System.out.println("Metric attribution ARN: " + metricAttributionArn);
        return metricAttributionArn;
    } catch (PersonalizeException e) {
        System.out.println(e.awsErrorDetails().errorMessage());
    }
    return "";
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { CreateMetricAttributionCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// set the metric attribution param
export const createMetricAttributionParam = {
  name: "METRIC_ATTRIBUTION_NAME",            /* required */
  datasetGroupArn: "DATASET_GROUP_ARN",       /* required */
  metricsOutputConfig: {
    roleArn: "ROLE_ARN",                      /* required */
```

```
    s3DataDestination: {
      kmsKeyArn: "KEY_ARN",                                              /*
 optional */
      path: "s3://<name of your output S3 bucket>/<folderName>/",    /* optional */
    },
  },
  metrics: [
    {
      eventType: "EVENT_TYPE",                        /* required for each metric */
      expression: "SUM(DatasetType.COLUMN_NAME)",   /* required for each metric */
      metricName: "METRIC_NAME",                      /* required for each metric */
    }
  ]
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new CreateMetricAttributionCommand(createMetricAttributionParam)
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Managing a metric attribution

After you create a metric attribution, you can update it or delete it. If you delete a metric attribution, Amazon Personalize stops sending reports related to PutEvents and incremental imports to CloudWatch.

Topics

- Updating a metric attribution
- Deleting a metric attribution

Updating a metric attribution

When you update a metric attribution, you can add and remove metrics and modify its output configuration. You can update a metric attribution with the Amazon Personalize console, AWS Command Line Interface, or AWS SDKS.

Topics

Updating a metric attribution (console)

To update a metric attribution with the Amazon Personalize console, you make your changes on the **Metric attribution** page.

To update a metric attribution

1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/home and sign into your account.
2. Choose your dataset group.
3. In the navigation pane, choose **Metric attribution**.
4. In the bottom section, choose the **Metric attributes** tab or **Metric attribution configuration** tab to start making changes.

   - To add or remove metrics, choose the **Metric attributes** tab and choose **Edit attributes**. Make your changes on the **Edit metric attributes** page and choose **Update** to save your changes.
   - To make changes to the Amazon S3 output bucket or IAM service role, choose the **Edit metric attribution configuration** tab and make changes on the **Edit attribution configuration** page. Choose **Update** to save your changes.

Updating a metric attribution (AWS CLI)

After you create a metric attribution, you can use the AWS Command Line Interface (AWS CLI) to add and remove metrics and modify its output configuration. The following code shows how to remove metrics with the `update-metric-attribution` command:

```
aws personalize update-metric-attribution \
--metric-attribution-arn metric attribution arn \
--remove-metrics metricName1 metricName2
```

The following code shows how to add an additional metric and specify a new output configuration:

```
aws personalize update-metric-attribution \
--metric-attribution-arn metric attribution arn \
--metrics-output-config "{\"roleArn\": \"new role ARN\", \"s3DataDestination\":
{\"kmsKeyArn\":\"kms key ARN\",\"path\":\"s3://new-bucket-name/new-folder-name/\"}}" \
--add-metrics "[{
  \"eventType\": \"event type\",
  \"expression\": \"SUM(DatasetType.COLUMN_NAME)\",
  \"metricName\": \"metric name\"
}]"
```

If successful, Amazon Personalize returns the ARN of the metric attribution you updated. For a complete listing of all parameters, see UpdateMetricAttribution.

Updating a metric attribution (AWS SDK)

After you create a metric attribution, you can add or remove metrics and modify its output configuration. The following code shows how to remove metrics from a metric attribution.

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

metricsToRemove = ["metricName1", "metricName2"]

response = personalize.update_metric_attribution(
  metricAttributionArn = "metric attribution ARN",
  removeMetrics = metricsToRemove
)
```

SDK for Java 2.x

```
public static void removeMetrics(PersonalizeClient client,
                                 String metricAttributionArn,
```

```
                                    String metric1Name,
                                    String metric2Name) {

    ArrayList<String> metricsToRemove = new ArrayList<>(Arrays.asList(metric1Name,
 metric2Name));

    try {

        UpdateMetricAttributionRequest request =
 UpdateMetricAttributionRequest.builder()
                .metricAttributionArn(metricAttributionArn)
                .removeMetrics(metricsToRemove)
                .build();

        UpdateMetricAttributionResponse response =
 client.updateMetricAttribution(request);
        System.out.println(response);

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

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import {UpdateMetricAttributionCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// set the update request param
export const updateMetricAttributionParam = {
  metricAttributionArn: "METRIC_ATTRIBUTION_ARN",    /* required */
  removeMetrics: ["METRIC_NAME_1", "METRIC_NAME_2"]    /* specify list of names of
 metrics to delete */
};
export const run = async () => {
  try {
    const response = await personalizeClient.send(
```

```
      new UpdateMetricAttributionCommand(updateMetricAttributionParam)
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

The following code shows how to add an additional metric and specify a new output configuration:

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

newMetrics = [{
      "eventType": "event type",
      "expression": "SUM(DatasetType.COLUMN_NAME)",
      "metricName": "metric name"
}]

newOutputConfig = {
  "roleArn": "Amazon Personalize service role ARN",
  "s3DataDestination": {
    "kmsKeyArn": "key ARN",
    "path": "s3://<name of your S3 bucket>/<folder>"
  }
}

response = personalize.update_metric_attribution(
  metricAttributionArn = "metric attribution arn",
  metricsOutputConfig = newOutputConfig,
  addMetrics = newMetrics
)
```

SDK for Java 2.x

```
public static void addMetricsAndUpdateOutputConfig(PersonalizeClient
 personalizeClient,
```

```
                                                  String metricAttributionArn,
                                                  String newMetric1EventType,
                                                  String newMetric1Expression,
                                                  String newMetric1Name,
                                                  String newMetric2EventType,
                                                  String newMetric2Expression,
                                                  String newMetric2Name,
                                                  String roleArn,
                                                  String s3Path,
                                                  String kmsKeyArn) {
    try {

        MetricAttribute newAttribute = MetricAttribute.builder()
                .eventType(newMetric1EventType)
                .expression(newMetric1Expression)
                .metricName(newMetric1Name)
                .build();

        MetricAttribute newAttribute2 = MetricAttribute.builder()
                .eventType(newMetric2EventType)
                .expression(newMetric2Expression)
                .metricName(newMetric2Name)
                .build();

        ArrayList<MetricAttribute> newAttributes = new
 ArrayList<>(Arrays.asList(newAttribute, newAttribute2));

        S3DataConfig newDataDestination = S3DataConfig.builder()
                .kmsKeyArn(kmsKeyArn)
                .path(s3Path)
                .build();

        MetricAttributionOutput newOutputConfig = MetricAttributionOutput.builder()
                .roleArn(roleArn)
                .s3DataDestination(newDataDestination)
                .build();

        UpdateMetricAttributionRequest request =
 UpdateMetricAttributionRequest.builder()
                .metricAttributionArn(metricAttributionArn)
                .metricsOutputConfig(newOutputConfig)
                .addMetrics(newAttributes)
                .build();
```

```
        UpdateMetricAttributionResponse response =
 personalizeClient.updateMetricAttribution(request);
        System.out.println("New metrics added!");
        System.out.println(response);

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

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import {UpdateMetricAttributionCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

export const updateMetricAttributionParam = {
  metricAttributionArn: "METRIC_ATTRIBUTION_ARN",
  addMetrics: [
    {
      eventType: "EVENT_TYPE",                      /* required for each metric */
      expression: "SUM(DatasetType.COLUMN_NAME)",   /* required for each metric */
      metricName: "METRIC_NAME",                    /* required for each metric */
    }
  ],
  metricsOutputConfig: {
    roleArn: "ROLE_ARN",                      /* required */
    s3DataDestination: {
      kmsKeyArn: "KEY_ARN",                                                   /*
 optional */
      path: "s3://<name of your output S3 bucket>/<folderName>/",    /* optional */
    },
  }
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(
```

```
      new UpdateMetricAttributionCommand(updateMetricAttributionParam)
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

If successful, Amazon Personalize returns the ARN of the metric attribution you updated. For a complete listing of all parameters, see UpdateMetricAttribution.

Deleting a metric attribution

If you no longer want to generate reports, you can delete a metric attribution. Deleting a metric attribution deletes all of its metrics and output configuration.

If you delete a metric attribution, Amazon Personalize stops automatically sending reports related to PutEvents and incremental bulk data to CloudWatch. Data already sent to CloudWatch or published to Amazon S3 is not affected. You can delete a metric attribution with the Amazon Personalize console, AWS Command Line Interface, or AWS SDKS.

Topics

- Deleting a metric attribution (console)
- Deleting a metric attribution (AWS CLI)
- Deleting a metric attribution (AWS SDKs)

Deleting a metric attribution (console)

You delete a metric attribution on the overview page for your metric attribution.

To delete a metric attribution

1. Open the Amazon Personalize console at https://console.aws.amazon.com/personalize/home and sign into your account.
2. Choose your dataset group.
3. In the navigation pane, choose **Metric attribution**.

4. Choose **Delete** and then confirm the deletion.

Deleting a metric attribution (AWS CLI)

To delete a metric attribution with the AWS CLI, use the `delete-metric-attribution` command as follows.

```
aws personalize delete-metric-attribution --metric-attribution-arn metric attribution
 ARN
```

Deleting a metric attribution (AWS SDKs)

The following code shows how to delete a metric attribution with the SDK for Python (Boto3):

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.delete_metric_attribution(
  metricAttributionArn = 'metric attribution ARN'
)
```

SDK for Java 2.x

```
public static void deleteMetricAttribution(PersonalizeClient client, String
 metricAttributionArn) {

    try {

        DeleteMetricAttributionRequest request =
 DeleteMetricAttributionRequest.builder()
                .metricAttributionArn(metricAttributionArn)
                .build();

        DeleteMetricAttributionResponse response =
 client.deleteMetricAttribution(request);
        if (response.sdkHttpResponse().statusCode() == 200) {
            System.out.println("Metric attribution deleted!");
        }
```

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

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { DeleteMetricAttributionCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

export const deleteMetricAttributionParam = {
  metricAttributionArn: "METRIC_ATTRIBUTION_ARN",
};
export const run = async () => {
  try {
    const response = await personalizeClient.send(
      new DeleteMetricAttributionCommand(deleteMetricAttributionParam)
    );
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Publishing and viewing results

Amazon Personalize sends the reports for each metric to CloudWatch or Amazon S3:

- For PutEvents data and incremental bulk data, Amazon Personalize automatically sends metrics to CloudWatch. For information on viewing and identifying reports in CloudWatch, see Viewing metrics in CloudWatch.

- For all bulk data, if you provide an Amazon S3 bucket when you create your metric attribution, you can choose to publish metric reports to your Amazon S3 bucket each time you create a dataset import job for interactions data.

  For information publishing metric reports to Amazon S3, see Publishing metrics to Amazon S3.

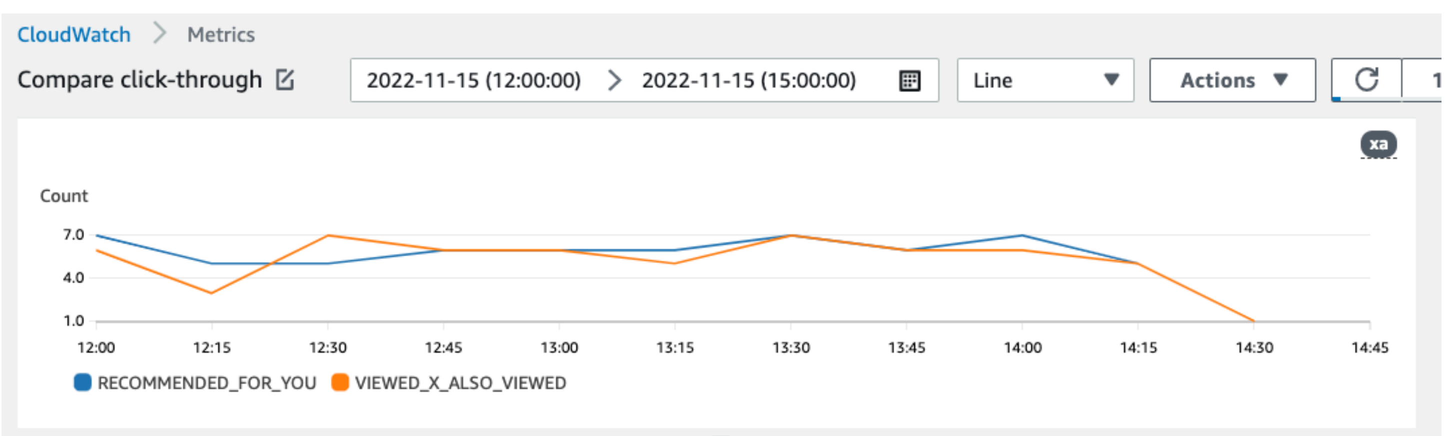
Topics

Viewing metrics in CloudWatch

> **Important**
>
> After you create a metric attribution and record events or import incremental bulk data, you will incur some monthly CloudWatch cost per metric. For information about CloudWatch pricing, see the Amazon CloudWatch pricing page. To stop sending metrics to CloudWatch, delete the metric attribution.

To view metrics in CloudWatch, complete the procedure found in Graphing a metric. The minimum **Period** you can graph is 15 minutes. For the search term, specify the name you gave the metric when you created the metric attribution.

The following is an example of how a metric might appear in CloudWatch. The metric shows the click-through rate for every 15 minutes for two different recommenders.



Publishing metrics to Amazon S3

To publish metrics to Amazon S3, you provide a path to your Amazon S3 bucket in your metric attribution. Then you publish reports to Amazon S3 when you create a dataset import job.

When the job completes, you can find the metrics in your Amazon S3 bucket. Each time you publish metrics, Amazon Personalize creates a new file in your Amazon S3 bucket. The file name includes the import method and date as follows:

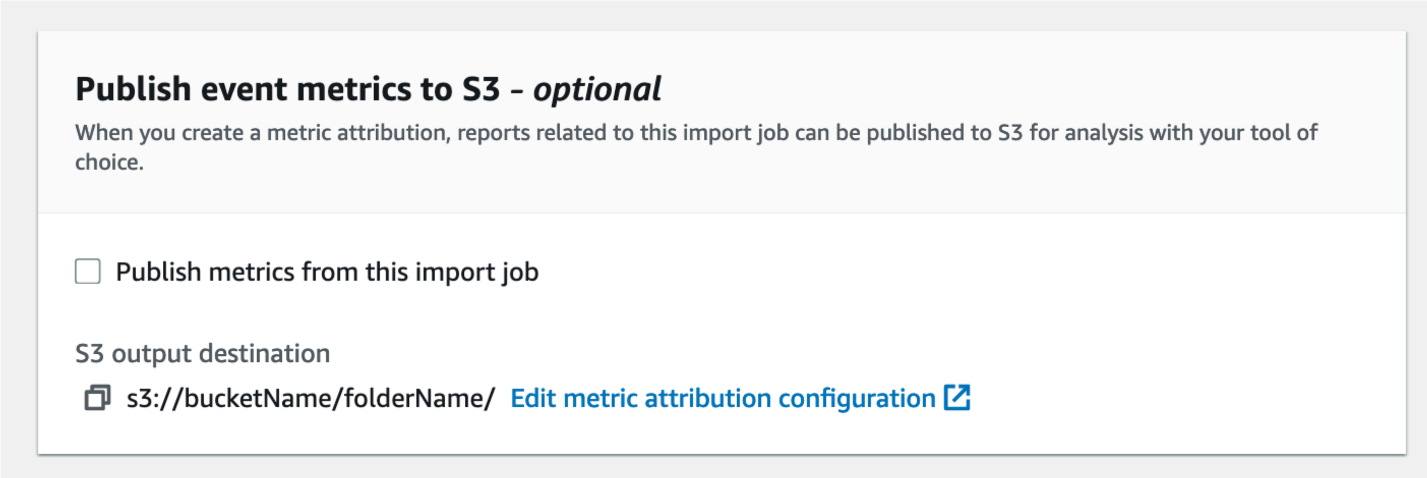
`AggregatedAttributionMetrics -` **`ImportMethod`** `-` **`Timestamp`**`.csv`

The following is an example of how the first few rows of a metric report CSV file might appear. The metric in this example reports on the total clicks from two different recommenders over 15 minute intervals. Each recommender is identified by its Amazon Resource Name (ARN) in the EVENT_ATTRIBUTION_SOURCE column.

```
METRIC_NAME,EVENT_TYPE,VALUE,MATH_FUNCTION,EVENT_ATTRIBUTION_SOURCE,TIMESTAMP
COUNTWATCHES,WATCH,12.0,samplecount,arn:aws:personalize:us-west-2:acctNum:recommender/
recommender1Name,1666925124
COUNTWATCHES,WATCH,112.0,samplecount,arn:aws:personalize:us-west-2:acctNum:recommender/
recommender2Name,1666924224
COUNTWATCHES,WATCH,10.0,samplecount,arn:aws:personalize:us-west-2:acctNum:recommender/
recommender1Name,1666924224
COUNTWATCHES,WATCH,254.0,samplecount,arn:aws:personalize:us-west-2:acctNum:recommender/
recommender2Name,1666922424
COUNTWATCHES,WATCH,112.0,samplecount,arn:aws:personalize:us-west-2:acctNum:recommender/
recommender1Name,1666922424
COUNTWATCHES,WATCH,100.0,samplecount,arn:aws:personalize:us-west-2:acctNum:recommender/
recommender2Name,1666922424
......
.....
```

Publishing metrics for bulk data to Amazon S3 (console)

To publish metrics to an Amazon S3 bucket with the Amazon Personalize console, create a dataset import job and choose **Publish metrics for this import job** in **Publish event metrics to S3**.



For step-by-step instructions, see Importing bulk records (console).

Publishing metrics for bulk data to Amazon S3 (AWS CLI)

To publish metrics to an Amazon S3 bucket with the AWS Command Line Interface (AWS CLI), use the following code to create a dataset import job and provide the `publishAttributionMetricsToS3` flag. If you don't want to publish metrics for a particular job, omit the flag. For information on each parameter, see CreateDatasetImportJob.

```
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 \
--import-mode INCREMENTAL \
--publish-attribution-metrics-to-s3
```

Publishing metrics for bulk data to Amazon S3 (AWS SDKs)

To publish metrics to an Amazon S3 bucket with the AWS SDKs, create a dataset import job and set `publishAttributionMetricsToS3` to true. For information on each parameter, see CreateDatasetImportJob.

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

```

```
response = personalize.create_dataset_import_job(
    jobName = 'YourImportJob',
    datasetArn = 'dataset_arn',
    dataSource = {'dataLocation':'s3://bucket/file.csv'},
    roleArn = 'role_arn',
    importMode = 'INCREMENTAL',
    publishAttributionMetricsToS3 = True
)

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

```
public static String createPersonalizeDatasetImportJob(PersonalizeClient
 personalizeClient,
                                                      String jobName,
                                                      String datasetArn,
                                                      String s3BucketPath,
                                                      String roleArn,
                                                      ImportMode importMode,
                                                      boolean publishToS3) {

  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)
                .importMode(importMode)
                .publishAttributionMetricsToS3(publishToS3)
                .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 "";
}
```

SDK for JavaScript v3

```
// Get service clients and commands using ES6 syntax.
import { CreateDatasetImportJobCommand, PersonalizeClient } from
  "@aws-sdk/client-personalize";

// create personalizeClient
const personalizeClient = new PersonalizeClient({
  region: "REGION"
});

// Set the dataset import job parameters.
export const datasetImportJobParam = {
  datasetArn: 'DATASET_ARN', /* required */
  dataSource: {
    dataLocation: 's3://<name of your S3 bucket>/<folderName>/<CSVfilename>.csv'  /*
 required */
  },
  jobName: 'NAME',                        /* required */
  roleArn: 'ROLE_ARN',                    /* required */
  importMode: "FULL",                     /* optional, default is FULL */
  publishAttributionMetricsToS3: true     /* set to true to publish metrics to
 Amazon S3 bucket */
};

export const run = async () => {
  try {
    const response = await personalizeClient.send(new
 CreateDatasetImportJobCommand(datasetImportJobParam));
    console.log("Success", response);
    return response; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
};
run();
```

Measuring recommendation impact with A/B testing

Performing an A/B test consists of running an experiment with multiple variations and comparing the results. Performing A/B testing with Amazon Personalize recommendations involves showing different groups of users different types of recommendations and then comparing the results. You can use A/B testing to help compare and evaluate different recommendation strategies, and measure the impact of the recommendations.

For example, you might use A/B testing to see if Amazon Personalize recommendations increase click-through rate. To test this scenario, you might show one group of users recommendations that are not personalized, such as featured products. And you might show another group personalized recommendations generated by Amazon Personalize. As your customers interact with items, you can record the outcomes and see which strategy results in the highest click-through rate.

The workflow for performing A/B testing with Amazon Personalize recommendations is as follows:

1. **Plan your experiment** – Define a quantifiable hypothesis, identify business goals, define experiment variations, and determine your experiment time frame.
2. **Split your users** – Split users into two or more groups, with a control group and one or more experiment groups.
3. **Run your experiment** – Show the users in the experiment group modified recommendations. Show the users in the control group recommendations with no changes. Record their interactions with recommendations to track results.
4. **Evaluate results** – Analyze experiment results to determine if the modification made a statistically significant difference for the experiment group.

You can use Amazon CloudWatch Evidently to perform A/B testing with Amazon Personalize recommendations. With CloudWatch Evidently, you can define your experiment, track key performance indicators (KPIs), route recommendation request traffic to the relevant Amazon Personalize resource, and evaluate experiment results. For more information, see A/B testing with CloudWatch Evidently.

Topics

A/B testing best practices

Use the following best practices to help you design and maintain A/B tests for Amazon Personalize recommendations.

- Identify a quantifiable business goal. Verify that the different recommendations that you want to compare both align with this business goal and are not related to different or non-quantifiable objectives.
- Define a quantifiable hypothesis that aligns with your business goal. For example, you might predict that a promotion for your own custom made content will result in 20% more clicks from these items. Your hypothesis determines the modification that you make for your experiment group.
- Define relevant key performance indicators (KPIs) related to your hypothesis. You use KPIs to measure the outcome of your experiments. These might be the following:
  - Click-through rate
  - Watch time
  - Total price
- Verify that the total number of users in the experiment is large enough to reach a statistically significant result, depending on your hypothesis.
- Define your traffic splitting strategy before you start your experiment. Avoid changing traffic splitting while the experiment is running.
- Keep the user experience of your application or website the same for both your experiment group and control group, except for modifications related to your experiment (for example, model). Variations in user experience, such as the UI or latency, can lead to misleading results.
- Control external factors, such as holidays, ongoing marketing campaigns, and browser limitations. These external factors can lead to misleading results.
- Avoid changing Amazon Personalize recommendations unless directly related to your hypothesis or business requirements. Changes like applying a filter or manually changing the order can lead to misleading results.
- When you evaluate results, make sure that the results are statistically significant before drawing conclusions. The industry standard is a 5% significance level. For more information about statistical significance, see A Refresher on Statistical Significance .

A/B testing with CloudWatch Evidently

After you create a recommender or deploy a custom solution version with a campaign, you can perform A/B tests with Amazon Personalize recommendations and Amazon CloudWatch Evidently. The following video describes the process of using CloudWatch Evidently to perform A/B testing with Amazon Personalize recommendations. For step-by-step instructions, see Performing an A/B test with CloudWatch Evidently.

Perform AB Testing with Amazon Personalize and CloudWatch Evidently

Topics

Performing an A/B test with CloudWatch Evidently

To perform an A/B test with Amazon Personalize and Amazon CloudWatch Evidently, create a CloudWatch Evidently project, define a feature and its variations, update your application to support your experiment, and create and run the experiment. As the experiment runs, you can view results in CloudWatch Evidently.

To perform an A/B test with Amazon Personalize and CloudWatch Evidently

1. Create a CloudWatch Evidently project. A project is a logical grouping of CloudWatch resources. Within the project, you create features that have variations that you want to test or launch. For step-by-step instructions, see Create a new project in the *Amazon CloudWatch User Guide*.
2. Add a feature to your project and define its variations. For this experiment, your feature should represent the recommendation scenario that you want to test, such as the click-through rate.

   When you add a feature, specify identifiers to map the different variations of your scenario to Amazon Personalize recommenders or custom campaigns. For each variation, specify the **Variation type**, such as String, give the variation a name, and give it a value.

   When your experiment runs, your application uses the value of variation to determine what Amazon Personalize resource to use for recommendations. For example, if you're testing two VIDEO_ON_DEMAND recommenders, one created for the *Top picks for you* use case and one created for the *Trending now* use case, you might set the following JSON as the **Value** for each variation.

   ```
   {"type":"top-picks-recommendations","arn":"arn:aws:personalize:us-west-2:<acct-
   id>:recommender/top-picks-recommender"}
   ```

```
{"type":"trending-recommendations","arn":"arn:aws:personalize:us-west-2:<acct-id>:recommender/trending-now-recommender"}
```

You can specify any identifier, as long as your application can use it to identify the relevant resource. For example, you might specify only the name of the recommender or campaign, and construct the Amazon Resource Name (ARN) of the resource in your application.

For step-by-step instructions to add a feature, see Add a feature to a project in the *Amazon CloudWatch User Guide*.

3. Update your application to support your experiment:

   - **Feature evaluation** – Use the CloudWatch Evidently `EvaluateFeature` API operation to assign variations to each user session. The `EvaluateFeature` response includes the variation value that you specified in the previous step. In this case, it's a JSON object with the type of recommender and it's the ARN of the recommender. Update your recommendation request code to get recommendations from this resource.

     For information about evaluating a feature, see Using EvaluateFeature in the *Amazon CloudWatch User Guide*.
   - **Record outcomes** – Add code to your application to track results from users' interactions with recommendations.

     To track metrics for your experiments in CloudWatch Evidently, use the CloudWatch Evidently `PutProjectEvents` API operation to record outcomes for each user. For example, if a user in an experiment clicks a recommended item, you would send details for this event to CloudWatch Evidently.

     For information about sending events to CloudWatch Evidently, see Using PutProjectEvents in the *Amazon CloudWatch User Guide*.

     To improve Amazon Personalize recommendation relevance, you can record outcome events with the Amazon Personalize `PutEvents` API operation. If your domain use case or custom recipe supports real-time updates to recommendations, Amazon Personalize can learn from your user's most recent activity and update recommendations as they use your application. If it doesn't support updates, Amazon Personalize uses this data during the next full retraining of your model and then it impacts recommendations.

     For information about streaming events to Amazon Personalize, see Recording events.

4. Create and start an experiment. When you create an experiment, specify the following:

   - **Feature** – Choose the feature to be tested in the experiment.
   - **Audience** – Configure how many of your users will participate, and configure how to split traffic between feature variations.
   - **Metrics** – Specify the metrics that determine the success of the experiment. For example, the number of clicks.

   After you finish creating the experiment, specify its duration and start the experiment. For step-by-step instructions to create and start experiments in CloudWatch Evidently, see Create an experiment in the *Amazon CloudWatch User Guide*.
5. As you run your experiment, you can view results in the CloudWatch Evidently experiment dashboard. For information about viewing experiment results, see View experiment results in the dashboard in the *Amazon CloudWatch User Guide*.

Sample implementations

The following sample implementations show how to implement A/B testing with CloudWatch Evidently.

- For a complete example of real-time APIs that include source code for implementing A/B tests, see Real-Time Personalization APIs in the AWS samples GitHub repository.
- For a sample retail web application that includes a workshop on personalization and A/B testing, see the Retail Demo Store in the AWS samples GitHub repository. For a notebook that describes how to create an A/B experiment with CloudWatch Evidently and the Retail Demo Store, see Retail Demo Store Experimentation Workshop - CloudWatch Evidently .
- For a tutorial that describes how to use A/B testing with CloudWatch Evidently and a sample react application, see Tutorial: A/B testing with the Evidently sample application in the *Amazon CloudWatch User Guide*.

Personalizing search results from OpenSearch

You can use Amazon Personalize to personalize results from open source OpenSearch or Amazon OpenSearch Service for your users.

OpenSearch is a self-managed, open source search service based on the Apache 2.0 License. Amazon OpenSearch Service is a managed service that helps you deploy, operate, and scale OpenSearch resources in the AWS Cloud. When you use Amazon OpenSearch Service, OpenSearch retrieves and ranks results.

When ranking query results, OpenSearch uses a probabilistic ranking framework called BM-25 to calculate relevance scores. If a distinctive keyword appears more frequently in a document, BM-25 assigns a higher relevance score to that document. OpenSearch ranking doesn't take into account user behavior like click-through data.

When you use Amazon Personalize with OpenSearch, Amazon Personalize re-ranks OpenSearch results based on a user's past behavior, any metadata about the items, and any metadata about the user. OpenSearch then incorporates the re-ranking before returning the search response to your application. You control how much weight OpenSearch gives the ranking from Amazon Personalize when applying it to OpenSearch results.

With this re-ranking, results can be more engaging and relevant to a user's interests. This can lead to an increase in the click-through rate and conversion rate for your application. For a use case example that describes how personalized search can improve results for an ecommerce application, see Use case example.

Before you start personalizing OpenSearch results, review the requirements listed in Guidelines and requirements.

Topics

Use case example

When you use Amazon Personalize to re-rank OpenSearch results, the search results can be more relevant for your users. For example, you might have an ecommerce application that sells cars. If your user enters a query for Toyota cars and you don't personalize results, OpenSearch would return a list of cars made by Toyota based on keywords in your data. This list would be ranked in the same order for all users.

But if you use Amazon Personalize to personalize results, OpenSearch re-ranks these cars in order of relevance for the specific user based on their behavior—for example, their clicks. The car that the user is most likely to click is ranked first.

When you personalize OpenSearch results, you control how much weight (emphasis) OpenSearch gives the ranking from Amazon Personalize. Continuing with this example, if a user searches for a specific type of car from a specific year (such as a 2008 Toyota Prius), you might want to put more emphasis on the original ranking from OpenSearch.

However, for more generic queries that result in a wide range of results (such as a search for all Toyota vehicles), you might put a high emphasis on personalization. This way, the cars at the top of the list are more relevant to the particular user.

Personalized search workflow

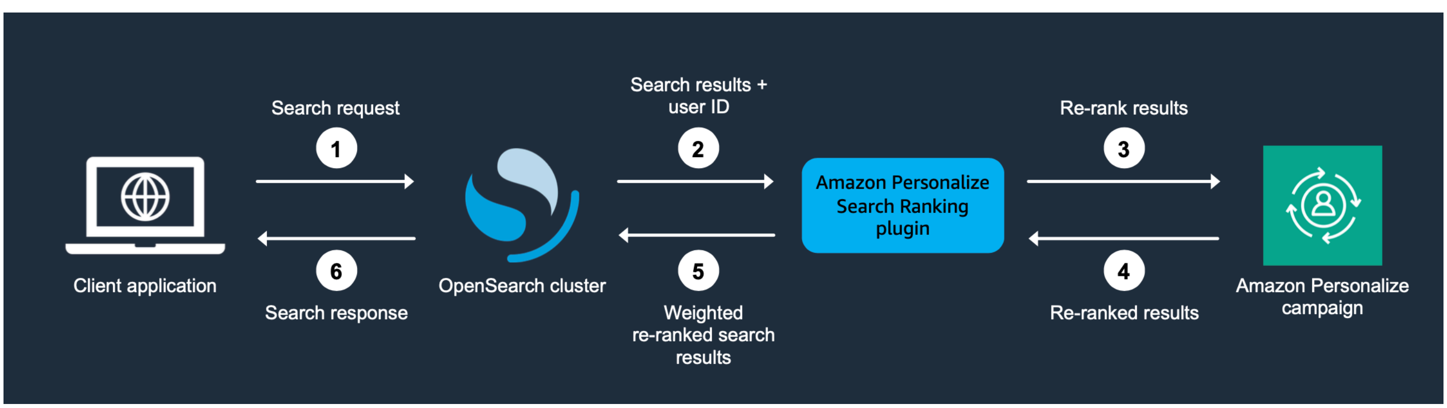
To personalize OpenSearch results, you do the following:

1. **Set up Amazon Personalize** – If you haven't already, complete the steps in Setting up Amazon Personalize to set up your credentials and set up permissions for Amazon Personalize. You don't need to set up the AWS SDKs to personalize OpenSearch results.
2. **Complete the Amazon Personalize workflow** – Complete the Amazon Personalize workflow to import data, create a solution with the Personalized-Ranking recipe, train a custom solution version, and deploy it in a campaign. You can only use the Personalized-Ranking recipe. You must create an Item interactions dataset. A Users dataset and an Items dataset are optional. For more information, see Amazon Personalize workflow.

3. **Set up OpenSearch and install the Amazon Personalize Search Ranking plugin –** If you haven't already, set up your OpenSearch Service domain or open source OpenSearch cluster. Then install the Amazon Personalize Search Ranking plugin. This plugin handles communication with Amazon Personalize and re-ranking results. For more information, see Setting up OpenSearch and installing the plugin.
4. **Configure the Amazon Personalize Search Ranking plugin –** To configure the plugin, you create search pipelines. *Search pipelines* are sets of request and response processors. When you create a pipeline for the plugin, you specify your Amazon Personalize resources in a `personalized_search_ranking` response processor. You also configure how much weight the plugin gives the results from Amazon Personalize when it re-ranks results. For more information, see Configuring the plugin.
5. **Apply the Amazon Personalize Search Ranking plugin to OpenSearch queries –** You can apply the Amazon Personalize Search Ranking plugin to all queries and responses for an OpenSearch index. You can also apply the plugin to individual OpenSearch queries. For more information, see Applying the plugin to OpenSearch queries.
6. **Compare results –** The Amazon Personalize Search Ranking plugin re-ranks the search results in the OpenSearch query response. It considers both the ranking from Amazon Personalize and the ranking from OpenSearch. To understand how results are re-ranked, you can compare results from queries that use personalization and those that don't. For more information, see Comparing OpenSearch results with results from the plugin.
7. **Monitor the Amazon Personalize Search Ranking plugin –** As you apply the Amazon Personalize Search Ranking plugin to search queries, you can monitor the plugin by getting metrics for your search pipelines. For more information, see Monitoring the plugin.

How the Amazon Personalize Search Ranking plugin works

The following diagram shows how the Amazon Personalize Search Ranking plugin works.



1. You submit your customer's query to your OpenSearch Service domain or your open source OpenSearch cluster.
2. OpenSearch sends the query response (list of items that are relevant to the query) and the user's ID to the Amazon Personalize Search Ranking plugin.
3. The plugin sends the items and user in the response to your Amazon Personalize campaign for ranking. It uses the recipe and campaign Amazon Resource Name (ARN) values in your search pipeline to get a personalized ranking for the user. It uses the GetPersonalizedRanking API operation for recommendations. In the request, it passes the userId of the user making the query and the items returned from the OpenSearch query in the `inputList`.
4. Amazon Personalize returns the re-ranked results to the plugin.
5. The plugin rearranges and returns the search results to your OpenSearch Service domain or open source OpenSearch cluster. It re-ranks the results based on the response from your Amazon Personalize campaign and the emphasis on personalization that you specify during setup.
6. Your open source OpenSearch cluster or OpenSearch Service domain returns the final results to your application.

Additional information

The following resources provide additional information about using OpenSearch.

- For information about getting started with open source OpenSearch, see Quickstart.
- For information about getting started with OpenSearch Service, see Getting started with Amazon OpenSearch Service in the *Amazon OpenSearch Service Developer Guide*.
- For information about the Personalized-Ranking recipe in Amazon Personalize, see Personalized-Ranking recipe.

Guidelines and requirements

This section includes requirements for using the Amazon Personalize Search Ranking plugin. It also describes how to set up permissions for Amazon OpenSearch Service or open source OpenSearch.

Topics

- Setting up open source OpenSearch permissions

Plugin requirements

Before you start personalizing results from OpenSearch, note the following guidelines and requirements for the Amazon Personalize Search Ranking plugin:

- You must use OpenSearch version 2.9.0 or later. If you use Amazon OpenSearch Service, your domain must use version 2.9 or later.
- If you haven't already, complete the instructions in Setting up permissions to grant your users permission to access Amazon Personalize and give Amazon Personalize permission to access your resources in Amazon Personalize.
- You must be able to access your Amazon Personalize resources from your OpenSearch Service domain or open source OpenSearch cluster.
  - For information about granting access for an OpenSearch Service domain, see Setting up Amazon OpenSearch Service permissions.
  - For information about granting access for an OpenSearch cluster, see Setting up open source OpenSearch permissions.
- You can use only custom Amazon Personalize resources. If you created a Domain dataset group, you can still add custom resources.
- You can only use the custom recipe Personalized-Ranking. For more information about this recipe, see Personalized-Ranking recipe.
- You must create an Item interactions dataset in Amazon Personalize. Items and Users datasets are optional.
- You can't apply Amazon Personalize filters when you're using the Amazon Personalize Search Ranking plugin.
- By default, the plugin assumes that the `_id` for an indexed document in OpenSearch matches the itemId in your Amazon Personalize data. If your OpenSearch data uses a different field that corresponds with your Amazon Personalize itemIds, you must specify the name of the field when you configure the plugin.
- The userId that you use for a user making a query must match their userId in the data you import into Amazon Personalize.
- The plugin re-ranks only the top 500 search results from OpenSearch. The remaining items are not re-ranked and end up at the bottom of the list.

Setting up Amazon OpenSearch Service permissions

If you use Amazon OpenSearch Service, you must be able to access your Amazon Personalize resources from your OpenSearch Service domain.

To set up permissions

1. Depending on if your resources are in the same or different accounts, create one or more IAM service roles with permission to access your resources.
   - If your OpenSearch Service and Amazon Personalize resources are in the same account, you create an IAM service role for OpenSearch Service and grant it permission to get a personalized ranking from your Amazon Personalize campaign. For more information, see Configuring permissions when resources are in the same account.
   - If your OpenSearch Service and Amazon Personalize resources are in separate accounts, you create two IAM service roles. You create one in the account with your OpenSearch Service resources and grant it access to your OpenSearch Service resources. And you create one in the account with your Amazon Personalize resources and grant it permission to get a personalized ranking from your Amazon Personalize campaign. For more information, see Configuring permissions when resources are in different accounts.
2. Grant the user or role that's accessing your OpenSearch Service domain `PassRole` permissions for the IAM service role that you created for OpenSearch Service. For more information, see Configuring Amazon OpenSearch Service domain security.

Topics

Configuring permissions when resources are in the same account

If your OpenSearch Service and Amazon Personalize resources are in the same account, you must create an IAM service role for OpenSearch Service. This role must have permission to get a personalized ranking from your Amazon Personalize campaign. The following is required to grant your OpenSearch Service service role permission to get a personalized ranking from your Amazon Personalize campaign:

- The role's trust policy must grant `AssumeRole` permissions for OpenSearch Service. For a trust policy example, see Trust policy example.
- The role must have permission to get a personalized ranking from your Amazon Personalize campaign. For a policy example, see Permissions policy example.

For information about creating an IAM role, see Creating IAM roles in the *IAM User Guide*. For information on attaching an IAM policy to role, see Adding and removing IAM identity permissions in the *IAM User Guide*.

After you create an IAM service role for OpenSearch Service, you must grant the user or role that's accessing your OpenSearch Service domain `PassRole` permissions for the OpenSearch Service service role. For more information, see Configuring Amazon OpenSearch Service domain security.

Topics

- Trust policy example
- Permissions policy example

Trust policy example

The following trust policy example grants `AssumeRole` permissions for OpenSearch Service.

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

Permissions policy example

The following policy example grants the role the minimum permissions to get a personalized ranking from your Amazon Personalize campaign. For `Campaign ARN`, specify the Amazon Resource Name (ARN) of your Amazon Personalize campaign.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "personalize:GetPersonalizedRanking"
            ],
            "Resource": "Campaign ARN"
        }
    ]
}
```

Configuring permissions when resources are in different accounts

If your OpenSearch Service and Amazon Personalize resources are in separate accounts, you create an IAM role in each account and grant the role access to the resources in the account.

To set up permissions for multiple accounts

1. In the account where your Amazon Personalize campaign exists, create an IAM role that has permission to get a personalized ranking from your Amazon Personalize campaign. When you configure the plugin, you specify the ARN for this role in the `external_account_iam_role_arn` parameter of the `personalized_search_ranking` response processor. For more information, see Configuring the plugin.

   For a policy example, see Permissions policy example.

2. In the account where your OpenSearch Service domain exists, create a role with a trust policy that grants OpenSearch Service `AssumeRole` permissions. When you configure the plugin, you specify the ARN for this role in the `iam_role_arn` parameter of the `personalized_search_ranking` response processor. For more information, see Configuring the plugin.

   For a trust policy example, see Trust policy example.

3. Modify each role to grant the other role `AssumeRole` permissions. For example, for the role that has access to your Amazon Personalize resources, its IAM policy would grant the role in the account with the OpenSearch Service domain assume role permissions as follows:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "",
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "arn:aws:iam::<Account number for role with access to
 OpenSearch Service domain>:role/roleName"

    }]
}
```

4. In the account where your OpenSearch Service domain exists, grant the user or role that's accessing your OpenSearch Service domain `PassRole` permissions for the OpenSearch Service service role you just created. For more information, see [Configuring Amazon OpenSearch Service domain security](#).

Configuring Amazon OpenSearch Service domain security

To use the plugin with OpenSearch Service, the user or role that's accessing your domain must have `PassRole` permissions for the [IAM service role for OpenSearch Service](#) you just created. Also, the user or role must have permission to perform the `es:ESHttpGet` and `es:ESHttpPut` actions.

For information about configuring access to OpenSearch Service, see [Security in Amazon OpenSearch Service](#) in the *Amazon OpenSearch Service Developer Guide*. For policy examples, see [Policy examples for OpenSearch Service user or role](#).

Policy examples for OpenSearch Service user or role

The following IAM policy example grants a user or role `PassRole` permissions for the IAM service role that you created for OpenSearch Service in [Configuring permissions when resources are in the same account](#).

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

```
        {
            "Sid": "",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "OpenSearch Service role ARN"
        }
    ]
}
```

The following IAM policy grants the minimum permissions to create pipelines and search queries with OpenSearch Service.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "es:ESHttpGet",
                "es:ESHttpPut"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:ResourceTag/environment": [
                        "production"
                    ]
                }
            }
        }
    ]
}
```

Setting up open source OpenSearch permissions

If you use open source OpenSearch, you must be able to access your Amazon Personalize resources from your open search cluster. To grant access, do the following:

- If you're setting up OpenSearch from scratch, you can use a quick start bash script to run an OpenSearch cluster in a Docker container. The script uses the default credentials in your AWS profile. You can specify an alternate profile when you run the script.

These credentials must be associated with a user or role that has permission to perform the GetPersonalizedRanking action for your Amazon Personalize campaign. For an example of an IAM policy, see IAM policy examples. Alternatively, the credentials must have permission to assume a role that has these permissions. You can provide the Amazon Resource Name (ARN) for this role when you create a pipeline for the Amazon Personalize Search Ranking plugin.

- If you don't use the quick start bash script, you can manually add your credentials to your OpenSearch keystore. These credentials must correspond with a user or role that has permission to perform the GetPersonalizedRanking action for your Amazon Personalize campaign.

  To manually add your AWS credentials to your OpenSearch keystore, run the following command where your OpenSearch cluster is running (such as a Docker container). Then provide each credential. If you don't use a session token, you can omit the final line in the command.

```
opensearch-keystore add \
personalized_search_ranking.aws.access_key \
personalized_search_ranking.aws.secret_key \
personalized_search_ranking.aws.session_token
```

- If you run your OpenSearch cluster on an Amazon EC2 instance, you can grant permissions with an IAM instance profile. The policy attached to the role must grant it permission to perform the GetPersonalizedRanking action for your Amazon Personalize campaign. It must also grant Amazon EC2 permissions to assume the role.

  For information about Amazon EC2 instance profiles, see Using instance profiles. For a policy example, see IAM policy examples.

IAM policy examples

The following policy example grants a user or role the minimum permissions to get a personalized ranking from your Amazon Personalize campaign. For `Campaign ARN`, specify the Amazon Resource Name (ARN) of your Amazon Personalize campaign.

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

```
                "personalize:GetPersonalizedRanking"
            ],
            "Resource": "Campaign ARN"
        }
    ]
}
```

Additionally, if you run your OpenSearch cluster on an Amazon EC2 instance and grant permissions with an IAM instance profile, the trust policy for the role must grant Amazon EC2 `AssumeRole` permissions as follows. For information about Amazon EC2 instance profiles, see Using instance profiles.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Setting up OpenSearch and installing the plugin

The Amazon Personalize Search Ranking plugin handles communication with Amazon Personalize from your OpenSearch Service domain or open source OpenSearch cluster. It also handles re-ranking results. Depending on how you access OpenSearch, you set up OpenSearch and install the plugin as follows:

- If you use Amazon OpenSearch Service, you set up OpenSearch by creating a domain in OpenSearch Service, ingesting data, and installing the plugin.
- If you use open source OpenSearch, you create an OpenSearch cluster, ingest data, and install the plugin.

Topics

- Setting up open source OpenSearch

Setting up Amazon OpenSearch Service

After you complete the Amazon Personalize workflow and meet the requirements listed in Guidelines and requirements, you're ready to set up Amazon OpenSearch Service and install the Amazon Personalize Search Ranking plugin.

To set up Amazon OpenSearch Service, you create a domain, ingest your data, and install the plugin. If you have already created a domain and ingested your data, you can skip to step 3.

To set up OpenSearch Service

1. If you haven't already, complete the steps in Setting up Amazon OpenSearch Service permissions so you can access your Amazon Personalize resources from your OpenSearch Service domain.
2. If you haven't already, create an OpenSearch Service domain. An *OpenSearch Service domain* is synonymous with an open source OpenSearch cluster. Domains are clusters with the settings, instance types, instance counts, and storage resources that you specify.
    - For a concise tutorial for configuring a test domain, see Step 1: Create an Amazon OpenSearch Service domain in the "Getting started" section of the *Amazon OpenSearch Service Developer Guide*.
    - For more detailed steps, see Creating and managing Amazon OpenSearch Service domains.
3. If you haven't already, ingest your items into OpenSearch Service.
    - For a concise tutorial for uploading a small amount of test data to OpenSearch Service, see Step 2: Upload data to Amazon OpenSearch Service for indexing in the "Getting started" section of the *Amazon OpenSearch Service Developer Guide*.
    - For complete information about ingesting data, see Indexing data in Amazon OpenSearch Service in the *Amazon OpenSearch Service Developer Guide*.
4. Associate the `Amazon_Personalize_Search_Ranking_Plugin` plugin with your domain. The plugin is preinstalled, and you don't have to import it from Amazon S3. You associate the plugin the same way that you associate an OpenSearch Service package.

    For information about associating an OpenSearch Service package, see Custom packages for Amazon OpenSearch Service.

After you create a domain, ingest data, and install the Amazon Personalize Search Ranking plugin, you're ready to configure the plugin. You configure it by creating a search pipeline and specifying a `personalized_search_ranking` response processor. For more information, see Configuring the plugin.

Setting up open source OpenSearch

After you complete the Amazon Personalize workflow and meet the requirements listed in Guidelines and requirements, you're ready to set up open source and install the Amazon Personalize Search Ranking plugin.

If you already have an OpenSearch cluster running, you can manually install the plugin. If you don't have a cluster running, you can install OpenSearch and the plugin from scratch with a bash script.

Topics

Manually installing the plugin on an existing OpenSearch cluster

If you already have an OpenSearch cluster, you can manually install the plugin on your cluster directly from the OpenSearch GitHub repository.

To manually install the plugin

1. Use the following command to start your OpenSearch cluster:

   ```
   bin/opensearch
   ```

2. If you haven't already, upload your catalog data to your OpenSearch cluster. When you upload your data, you create an OpenSearch index and define your field mappings. Then you upload your data to that index. For an example, see Create an index and field mappings using sample data.

3. Use the following command to install the plugin:

   ```
   bin/opensearch-plugin install https://github.com/opensearch-project/search-processor/releases/download/2.9.0/opensearch-search-processor-2.9.0.0.zip
   ```

   For more information about installing plugins, see Installing plugins.

After you install the Amazon Personalize Search Ranking plugin, you're ready to configure it. You configure the plugin by creating a search pipeline and specifying a `personalized_search_ranking` response processor. For more information, see Configuring the plugin.

Setting up your cluster and installing the plugin with a quickstart script

If you haven't created an OpenSearch cluster, you can use a quickstart bash script to create one. This script sets up an OpenSearch cluster in a Docker container, sets up credentials using your default AWS profile, and installs the Amazon Personalize Search Ranking plugin.

For information about manually creating an OpenSearch cluster, see the Quickstart instructions in the OpenSearch documentation.

To install the plugin with a quickstart bash script

1. Before you run the script, download and install Docker Desktop for your operating system.
2. Download the quick start bash script from GitHub.
3. In your working directory, run the script with the following command.

```
sh personalized_search_ranking_quickstart.sh
```

   With this command, the script uses the credentials in your default AWS profile. To provide an alternate profile, use the `--profile` argument.

```
sh personalized_search_ranking_quickstart.sh --profile profile-name
```

   After you run the script, you can find more information about the script in the README file that's located in the unique directory created by the script. This directory stores the Dockerfile and docker-compose.yml files that the script uses. For example: `../opensearch-personalize-intelligent-ranking-docker.1234/README`.
4. Upload your catalog data to your OpenSearch cluster. When you upload your data, you create an OpenSearch index and define your field mappings. Then you upload your data to that index. For an example, see Create an index and field mappings using sample data.

After you set up OpenSearch and install the Amazon Personalize Search Ranking plugin, you're ready to configure it. You configure the plugin by creating a search pipeline and specifying a personalized_search_ranking response processor. For more information, see Configuring the plugin.

Configuring the plugin

After you install the Amazon Personalize Search Ranking plugin, you're ready to configure it by creating an OpenSearch search pipeline.

A *search pipeline* is a set of request and response processors that run sequentially in the order that you create them. When you create a search pipeline for the plugin, you specify a personalized_search_ranking response processor. For information about search pipelines, see Search pipelines.

Topics

- Fields for the personalized_search_ranking response processor
- Creating a pipeline with Amazon OpenSearch Service
- Creating a pipeline with open source OpenSearch

Fields for the personalized_search_ranking response processor

For the personalized_search_ranking response processor, you specify the following fields:

- **campaign_arn (required)** – Specify the Amazon Resource Name (ARN) of the Amazon Personalize campaign to use to personalize results.
- **item_id_field (optional)** – If the _id field for an indexed document in OpenSearch doesn't correspond with your Amazon Personalize itemIds, specify the name of the field that does. By default, the plugin assumes that the _id data matches the itemId in your Amazon Personalize data.
- **recipe (required)** – Specify the name of the Amazon Personalize recipe to use. You can specify only aws-personalized-ranking.
- **weight (required)** – Specify the emphasis that the response processor puts on personalization when it re-ranks results. Specify a value within a range of 0.0–1.0. The closer to 1.0 that it is, the more likely it is that results from Amazon Personalize rank higher. If you specify 0.0, no personalization occurs and OpenSearch takes precedence.
- **tag (optional)** – Specify an identifier for the processor.

- **iam_role_arn (required for OpenSearch Service, optional for open source OpenSearch)** – For OpenSearch Service, provide the Amazon Resource Name (ARN) for the role that you created when setting up permissions for OpenSearch Service to access your Amazon Personalize resources. If your OpenSearch Service and Amazon Personalize resources exist in different accounts, specify the role that grants `AssumeRole` permissions for OpenSearch Service. For more information, see Configuring permissions when resources are in different accounts.

  For open source OpenSearch, if you use multiple roles to restrict permissions for different groups of users in your organization, specify the ARN of the role that has permission to access Amazon Personalize. If you use only the AWS credentials in your OpenSearch keystore, you can omit this field.
- **aws_region (required)** – The AWS Region where you created your Amazon Personalize campaign.
- **ignore_failure (optional)** – Specify whether the plugin ignores any processor failures. For values, specify `true` or `false`. For your production environments, we recommend that you specify `true` to avoid any interruptions for query responses. For test environments, you can specify `false` to view any errors that the plugin generates.
- **external_account_iam_role_arn** – If you use OpenSearch Service, and your Amazon Personalize and OpenSearch Service resources exist in different accounts, specify the ARN of the role that has permission to access your Amazon Personalize resources. This role must exist in the same account as your Amazon Personalize resources. For more information, see Configuring permissions when resources are in different accounts.

Creating a pipeline with Amazon OpenSearch Service

You can use the following Python code to create a search pipeline with a `personalized_search_ranking` response processor on an OpenSearch Service domain. Replace `domain endpoint` with your domain endpoint URL. For example: `https://<domain name>.<AWS region>.es-staging.amazonaws.com`.

```
import requests
from requests_auth_aws_sigv4 import AWSSigV4

domain_endpoint = 'domain endpoint'
pipeline_name = 'pipeline name'
url = f'{domain_endpoint}/_search/pipeline/{pipeline_name}'
auth = AWSSigV4('es')

headers = {'Content-Type': 'application/json'}
```

```
body = {
  "description": "A pipeline to apply custom re-ranking from Amazon Personalize",
  "response_processors": [
    {
      "personalized_search_ranking" : {
        "campaign_arn" : "Amazon Personalize Campaign ARN",
        "item_id_field" : "productId",
        "recipe" : "aws-personalized-ranking",
        "weight" : "0.3",
        "tag" : "personalize-processor",
        "iam_role_arn": "Role ARN",
        "aws_region": "AWS region",
        "ignore_failure": true
    }
  ]
}
try:
    response = requests.put(url, auth=auth, json=body, headers=headers, verify=False)
    print(response.text)
except Exception as e:
    print(f"Error: {e}")
```

After you create a search pipeline with a `personalized_search_ranking` response processor, you're ready to start applying the plugin to OpenSearch queries. You can apply it to an OpenSearch index or an individual OpenSearch query. For more information, see [Applying the plugin to OpenSearch queries](#).

Creating a pipeline with open source OpenSearch

You can use the following curl command to create a search pipeline with a `personalized_search_ranking` response processor on an open source OpenSearch cluster.

```
curl -X PUT "http://localhost:9200/_search/pipeline/pipeline-name" -ku 'admin:admin' --
insecure -H 'Content-Type: application/json' -d'
{
  "description": "A pipeline to apply custom re-ranking from Amazon Personalize",
  "response_processors" : [
    {
      "personalized_search_ranking" : {
        "campaign_arn" : "Amazon Personalize Campaign ARN",
        "item_id_field" : "productId",
```

```
        "recipe" : "aws-personalized-ranking",
        "weight" : "0.3",
        "tag" : "personalize-processor",
        "iam_role_arn": "Role ARN",
        "aws_region": "AWS region",
        "ignore_failure": true
      }
    }
  ]
}'
```

After you create a search pipeline with a `personalized_search_ranking` response processor, you're ready to start applying the plugin to OpenSearch queries. You can apply it to an OpenSearch index or an individual OpenSearch query. For more information, see Applying the plugin to OpenSearch queries.

Applying the plugin to OpenSearch queries

After you configure a search pipeline with a `personalized_search_ranking` response processor, you're ready to apply the Amazon Personalize Search Ranking plugin to your OpenSearch queries and view the re-ranked results.

As you apply the plugin to OpenSearch queries, you can monitor the plugin by getting metrics for your search pipeline. For more information, see Monitoring the plugin.

Topics

Applying the plugin to Amazon OpenSearch Service queries

You can apply the Amazon Personalize Search Ranking plugin to all queries and responses for an index. You can also apply the plugin to individual queries and responses.

- You can use the following Python code to apply a search pipeline to an index. With this approach, all searches using this index use the plugin to apply personalization to search results.

```
import requests
from requests_auth_aws_sigv4 import AWSSigV4
```

```
domain_endpoint = 'domain endpoint'
index = 'index name'
url = f'{domain_endpoint}/{index}/_settings/'
auth = AWSSigV4('es')
headers = {'Content-Type': 'application/json'}
body = {
    "index.search.default_pipeline": "pipeline name"
}
try:
    response = requests.put(url, auth=auth, json=body, headers=headers)
    print(response.text)
except Exception as e:
    print(f"Error: {e}")
```

- You can use the following Python code to apply a search pipeline to an individual query for Toyota brand cars.

  Update the code to specify your domain endpoint, your OpenSearch Service index, the name of your pipeline, and your query. For `user_id`, specify the ID of the user that you're getting search results for. This user must be in the data that you used to create your Amazon Personalize solution version. If the user wasn't present, Amazon Personalize ranks the items based on their popularity.

  For `context`, if you use contextual metadata, provide the user's contextual metadata, such as their device type. The `context` field is optional. For more information, see [Increasing recommendation relevance with contextual metadata](#).

```
import requests
from requests_auth_aws_sigv4 import AWSSigV4

domain_endpoint = 'domain endpoint'
index = 'index name'
url = f'{domain_endpoint}/{index}/_search/'

auth = AWSSigV4('es')
headers = {'Content-Type': 'application/json'}
params = {"search_pipeline": "pipeline-name"}
body = {
    "query": {
        "multi_match": {
            "query": "Toyota",
```

```
            "fields": ["BRAND"]
        }
    },
    "ext": {
        "personalize_request_parameters": {
            "user_id": "USER ID"
            "context": { "DEVICE" : "mobile phone" }
        }
    }
}
try:
    response = requests.post(url, auth=auth, params=params, json=body,
 headers=headers, verify=False)
    print(response)
except Exception as e:
    print(f"Error: {e}")
```

Applying the plugin to queries in open source OpenSearch

You can apply the Amazon Personalize Search Ranking plugin to all queries and responses for an OpenSearch index. You can also apply the plugin to individual OpenSearch queries and responses.

- The following curl command applies a search pipeline to an OpenSearch index in an open source OpenSearch cluster running locally. With this approach, all searches at this index use the plugin to apply personalization to search results.

```
curl -XGET "https://localhost:9200/index/_settings" -ku 'admin:admin' --insecure -H
 'Content-Type: application/json' -d'
{
  "index.search.default_pipeline" : "pipeline-name"
}
'
```

- The following curl command applies a search pipeline to an individual query for Toyota brand cars on an index in an open source OpenSearch cluster running locally.

  For `user_id`, specify the ID of the user that you're getting search results for. This user must be in the data that you used to create your Amazon Personalize solution version. If the user wasn't present, Amazon Personalize ranks the items based on their popularity. For `context`, if you use contextual metadata, provide the user's contextual metadata, such as their device type. The context field is optional. For more information, see [Increasing recommendation relevance with contextual metadata].

```
curl -XGET "http://localhost:9200/index/_search?search_pipeline=pipeline-name" -ku
 'admin:admin' --insecure -H 'Content-Type: application/json' -d'
{
  "query": {
    "multi_match": {
      "query": "Toyota",
      "fields": ["BRAND"]
    }
  },
  "ext": {
    "personalize_request_parameters": {
      "user_id": "user ID",
      "context": { "DEVICE" : "mobile phone" }
    }
  }
}
'
```

To understand how results are re-ranked, you can use OpenSearch Dashboards to compare OpenSearch results against re-ranked results with the plugin. For more information, see Comparing OpenSearch results with results from the plugin.

As you apply the plugin to OpenSearch queries, you can monitor the plugin by getting metrics for your OpenSearch pipeline. For more information, see Monitoring the plugin.

Comparing OpenSearch results with results from the plugin

The Amazon Personalize Search Ranking plugin rearranges search results based on both the ranking from Amazon Personalize and the ranking from OpenSearch. The way that the plugin re-ranks the results depends on how you configured the personalized_search_ranking response processor in your pipelines.

To understand how results are re-ranked, you can run queries with and without personalization, and compare the results.

Topics

- Comparing results with Amazon OpenSearch Service

- Comparing results with open source OpenSearch

Comparing results with Amazon OpenSearch Service

To understand how results are ranked, you can run queries with and without personalization, and compare the results. You can use the following Python code to run two different queries and output the results to two JSON files. The first method runs a query that uses the plugin to re-rank results. The second runs a method that generates results without personalization.

```
import json
import requests
from requests_auth_aws_sigv4 import AWSSigV4


# Returns re-ranked OpenSearch results using the Amazon Personalize Search Ranking
 plugin.
def get_personalized_results(pipeline_name):
    url = f'{domain}/{index}/_search/'
    auth = AWSSigV4('es')
    headers = {'Content-Type': 'application/json'}
    params = {"search_pipeline": pipeline_name}
    body = {
        "query": {
            "multi_match": {
                "query": "Toyota",
                "fields": ["BRAND"]
            }
        },
        "ext": {
            "personalize_request_parameters": {
                "user_id": "1"
            }
        }
    }
    try:
        response = requests.post(url, auth=auth, params=params, json=body,
 headers=headers, verify=False)
    except Exception as e:
        return f"Error: {e}"
    return response.text
```

```
# Returns OpenSearch results without personalization.
def get_opensearch_results():
    url = f'{domain}/{index}/_search/'
    auth = AWSSigV4('es')
    headers = {'Content-Type': 'application/json'}
    body = {
        "query": {
            "multi_match": {
                "query": "Toyota",
                "fields": ["BRAND"]
            }
        }
    }
    try:
        response = requests.post(url, auth=auth, json=body, headers=headers,
 verify=False)
    except Exception as e:
        return f"Error: {e}"
    return response.text


def print_results(file_name, results):
    results_file = open(file_name, 'w')
    results_file.write(json.dumps(results, indent=4))
    results_file.close()


# specify domain endpoint
domain = "DOMAIN_ENDPOINT"

# specify the region where you created your Amazon Personalize resources and Amazon
 OpenSearch domain
aws_region = "REGION"

# specify the name of the pipeline that uses the Amazon Personalize plugin
pipeline_name = "PIPELINE_NAME"

# specify your Amazon OpenSearch index
index = "INDEX"

# specify names for json files for comparison
personalized_results_file = "personalized_results.json"
opensearch_results_file = "opensearch_results.json"
```

```
# get personalized results
personalized_results = json.loads(get_personalized_results(pipeline_name))

# get OpenSearch results without personalization
opensearch_results = json.loads(get_opensearch_results())

# print results to files
print_results(personalized_results_file, personalized_results)
print_results(opensearch_results_file, opensearch_results)
```

Comparing results with open source OpenSearch

To understand how results are re-ranked, you can run queries with the Dev Tools console in two separate browser windows. Then you can compare results for queries with and without personalization.

To compare results with the Dev Tools console

1. If you haven't already, follow the steps in Setting up OpenSearch and installing the plugin and Configuring the plugin.
2. Make sure OpenSearch Dashboards is installed. The quickstart bash script installs OpenSearch Dashboards. If you don't use the script or already have a cluster running, you must install OpenSearch Dashboards. For more information, see Installing OpenSearch Dashboards.
3. Launch OpenSearch Dashboards. Open `http://localhost:5601` from a browser and sign in to OpenSearch Dashboards. The default credentials are username 'admin' and password 'admin'.
4. Choose **Dev Tools** under the Management menu on the OpenSearch Dashboards home page.
5. Open a separate browser window and open the Dev Tools console again. You can use the URL from the previous window.
6. In one window, enter a query that doesn't use any re-ranking for personalization.
   In the other window, enter a curl command that uses a pipeline with the `personalized_search_ranking` response processor. If you paste a curl command directly into the console, the command is automatically converted into the format that the console uses. For a command example, see Applying the plugin to OpenSearch queries.
7. Run both queries and compare the results.

Monitoring the plugin

If you use OpenSearch Service, you can monitor the plugin through metrics in Amazon CloudWatch. For more information, see Monitoring Amazon OpenSearch Service domains.

As you apply the Amazon Personalize Search Ranking plugin to OpenSearch queries, you can monitor the plugin by getting metrics for your search pipelines. Pipeline metrics include statistics like the number of failed requests for the `personalized_search_ranking` response processor.

Topics

- Monitoring the plugin with Amazon OpenSearch Service
- Monitoring the plugin with open source OpenSearch
- Pipeline metrics example

Monitoring the plugin with Amazon OpenSearch Service

You can use the following Python code to get metrics for all of your pipelines. For an example of pipeline metrics, see Pipeline metrics example.

```
import requests
from requests_auth_aws_sigv4 import AWSSigV4

domain_endpoint = 'domain endpoint'
url = f'{domain_endpoint}/_nodes/stats/search_pipeline'

auth = AWSSigV4('es')
headers = {'Content-Type': 'application/json'}
try:
    response = requests.get(url, auth=auth, headers=headers, verify=False)
    print(response.text)
except Exception as e:
    print(f"Error: {e}")
```

Monitoring the plugin with open source OpenSearch

You can use the following code to get metrics for all of your pipelines. The response contains statistics for all search pipelines. For an example of pipeline metrics, see Pipeline metrics example.

```
curl -XGET "https://localhost:9200/_nodes/stats/search_pipeline?pretty" -ku
 'admin:admin'
```

Pipeline metrics example

The following code shows an excerpt of the pipeline metrics that are returned from OpenSearch. It shows only the `pipelines` object that contains statistics for two different pipelines. For each pipeline, you can find Amazon Personalize Search Ranking plugin metrics in the `personalized_search_ranking` response processor list. For a complete example of all metrics, see [Search pipeline metrics](#).

```
{
....
....
  "pipelines": {
    "pipelineA": {
      "request": {
        "count": 0,
        "time_in_millis": 0,
        "current": 0,
        "failed": 0
      },
      "response": {
        "count": 6,
        "time_in_millis": 2246,
        "current": 0,
        "failed": 0
      },
      "request_processors": [],
      "response_processors": [
        {
          personalized_search_ranking": {
            "type": "personalized_search_ranking",
            "stats": {
              "count": <number of requests>,
              "time_in_millis": <time>,
              "current": 0,
              "failed": <number of failed requests>
            }
          }
        }
      ]
```

```
    },
    "pipelineB": {
      "request": {
        "count": 0,
        "time_in_millis": 0,
        "current": 0,
        "failed": 0
      },
      "response": {
        "count": 8,
        "time_in_millis": 2248,
        "current": 0,
        "failed": 0
      },
      "request_processors": [],
      "response_processors": [
        {
          "personalized_search_ranking": {
            "type": "personalized_search_ranking",
            "stats": {
              "count": <number of requests>,
              "time_in_millis": <time>,
              "current": 0,
              "failed": <number of failed requests>
            }
          }
        }
      ]
    }
  }
....
....
}
```

Tagging Amazon Personalize resources

A *tag* is a label that you optionally define and associate with AWS resources, including certain types of Amazon Personalize resources. A resource can have as many as 50 tags.

Tags can help you categorize and manage resources in different ways, such as by purpose, environment, or other criteria. For example, you can use tags to split revenue between different functions, or identify development environments for different resources.

To retrieve Amazon Personalizes resources by tag, you can use the filters in GetResources operation of Resource Groups Tagging API. For more information, see GetResources in the *Resource Groups Tagging API* API Reference guide.

You can add tags to the following types of Amazon Personalize resources:

- Batch inference jobs
- Batch segment jobs
- Campaigns
- Datasets
- Dataset groups
- Dataset import and export jobs
- Event trackers
- Filters
- Recommenders
- Solutions
- Solution versions

Topics

Guidelines and requirements

Each tag consists of a required tag key and an optional tag value, both of which you define. A tag key is a general label that acts as a category for more specific tag values. A tag value acts as a descriptor for a tag key.

For example, if you have two versions of an Amazon Personalize dataset group (one for internal testing and another for production), you might assign an `Environment` tag key to both projects. The tag value of the `Environment` tag might be `Test` for one version of the dataset group and `Production` for the other version.

The following restrictions apply to tags:

- Maximum number of tags per resource – 50
- Maximum key length – 128 Unicode characters in UTF-8
- Maximum value length – 256 Unicode characters in UTF-8
- Tag keys and values can contain the following characters: A-Z, a-z, 0-9, space, and _ . : / = + @ – (hyphen). This is the standard set of characters available across AWS services that support tags. Some services support additional symbols.
- Tag keys and tag values are case sensitive.
- For each associated resource, each tag key must be unique and it can have only one tag value.
- Your tag keys and tag values can't start with `aws:`. AWS services apply tags that start with `aws:`, and those tags can't be modified. They don't count towards tag limits.
- You can't update or delete a resource based only on its tags. You must also specify the Amazon Resource Name (ARN) or resource ID, depending on the operation that you use.

Additional information

For more information about tagging, see the following resources.

- AWS Tagging Principles in the *AWS General Reference*
- AWS Tagging Strategies (downloadable PDF)
- AWS Access Control in the *AWS IAM User Guide*
- AWS Tagging Policies in the *AWS Organizations User Guide*

Adding tags to Amazon Personalize resources

You can add, display, update, and remove tag keys and values from Amazon Personalize resources with the Amazon Personalize console, AWS Command Line Interface (AWS CLI), or AWS SDKs. The following examples show how to add a tag to Amazon Personalize dataset group. You can add tags to other Amazon Personalize resources in the same way.

Topics

Adding tags (console)

When you create a resource in Amazon Personalize, you can add optional tags with the Amazon Personalize console. The following example adds a tag to a dataset group.

To add tags to a new 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. For **Name**, enter a name.
4. For **Domain**, choose a domain.
5. Expand the **Tags** section and choose **Add new tag**.
6. For **Key** and **Value**, enter appropriate values.

   For example, **`Environment`** and **`Test`**, respectively.
7. To add more tags, choose **Add new tag**.

   You can add up to 50 tags to a resource.
8. Choose **Next** to continue creating your resource.

Adding tags to an existing resource is similar: Choose your resource and use the **Tags** fields to add your tags.

Adding tags (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to add tags when you create a resource or add tags to an existing resource.

Topics

Adding tags when you create a resource

To create a new resource and add a tag to it with the AWS CLI, use the appropriate `create` command for the resource and include the `tags` parameter and values. For example, the following command creates a new Domain dataset group named `myDatasetGroup` for the ECOMMERCE domain, and adds the following tags: An `Environment` tag key with a `Test` tag value, and a `Owner` tag key and a `xyzCorp` value.

```
aws personalize create-dataset-group \
--name myDatasetGroup \
--domain ECOMMERCE \
--tags tagKey=Environment,tagValue=Test tagKey=Owner,tagValue=xyzCorp
```

For information about the commands that you can use to create an Amazon Personalize resource, see the Amazon Personalize AWS CLI Command Reference.

Adding tags to an existing resource

To add a tag to an existing resource, use the `tag-resource` command. Specify the ARN of the resource and provide the tag key and value in the `tags` parameter.

```
aws personalize tag-resource \
--resource-arn resource ARN \
--tags tagKey=key,tagValue=value
```

Adding tags (AWS SDKs)

You can use the AWS SDKs to add tags when you create a resource, or to add tags to an existing resource.

Topics

Adding tags when you create a resource

To create a new resource and add a tag to it with the AWS SDKs, use the appropriate `create` method. Use the `tags` parameter to specify the key-value pairs for each of your tags. For example, the following code creates a new Domain dataset group named `myDatasetGroup` for the ECOMMERCE domain and adds the following tags: An `Environment` tag key with a `Test` tag value, and a `Owner` tag key and a `xyzCorp` value.

SDK for Python (Boto3)

```
import boto3

personalize = boto3.client('personalize')

response = personalize.create_dataset_group(
  name = 'myDatasetGroup'
  domain = 'ECOMMERCE'
  tags = [
    {
      'tagKey': 'Environment',
      'tagValue': 'Test'
    },
    {
      'tagKey': 'Owner',
      'tagValue': 'xyzCorp'
    }
  ]
)
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 String createDomainDatasetGroup(PersonalizeClient personalizeClient,
                                              String datasetGroupName,
                                              String domain) {

    try {

        ArrayList <Tag> tags = new ArrayList<>();

        Tag tag1 = Tag.builder()
                .tagKey("Environment")
                .tagValue("Test")
                .build();
        tags.add(tag1);
        Tag tag2 = Tag.builder()
                .tagKey("Owner")
                .tagValue("xyzCorp")
                .build();
        tags.add(tag2);

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

Adding tags to an existing resource

The following code shows how to add a tag to an existing Amazon Personalize resource. Specify the Amazon Resource Name (ARN) of the resource that you want to add tags to and specify key-value pairs for each of your tags.

SDK for Python (Boto3)

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

add_tags_response = personalize.tag_resource(
  resourceArn = "resourceArn",
  tags = [
    {
      'tagKey': 'Environment',
      'tagValue': 'Test'
    },
    {
      'tagKey': 'Owner',
      'tagValue': 'xyzCorp'
    }
  ]
)
```

SDK for Java 2.x

```
public static void tagResource(PersonalizeClient personalizeClient,
                                              String resourceArn,
                                              String domain) {

    try {

         ArrayList <Tag> tagList = new ArrayList<>();

          Tag tag1 = Tag.builder()
                  .tagKey("Environment")
                  .tagValue("Test")
                  .build();
          tags.add(tag1);
          Tag tag2 = Tag.builder()
                  .tagKey("Owner")
                  .tagValue("xyzCorp")
                  .build();
          tags.add(tag2);

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                .resourceArn(resourceArn)
                .tags(tagList)
```

```
                .build();

        personalizeClient.tagResource(tagResourceRequest);
        System.out.println("Tags have been added to "+ resourceArn);

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

Removing tags from Amazon Personalize resources

You can remove tags from Amazon Personalize resources with the Amazon Personalize console or the UntagResource API operation with the AWS Command Line Interface (AWS CLI) or AWS SDKs. The following examples show how to remove a tag from an Amazon Personalize dataset group. You can remove tags from other Amazon Personalize resources in the same way.

Topics

- Removing tags (console)
- Removing tags (AWS CLI)
- Removing tags (AWS SDKs)

Removing tags (console)

After you add tags to a resource in Amazon Personalize, you can remove the tags with the Amazon Personalize console. The following example removes a tag from a dataset group

To remove tags from 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 your dataset group.
3. At the bottom of the page, choose the **Tags** tab and choose **Manage tags**.
4. For each tag that you want to remove, choose **Remove**.
5. Choose **Save** to remove the tags.

Removing tags (AWS CLI)

To remove tags from an existing resource with the AWS CLI, use the following `untag-resource` command. For `resource-arn`, specify the Amazon Resource Name (ARN) of the resource. For `tag-keys`, specify the keys of the tags to be removed.

```
aws personalize untag-resource \
--resource-arn resource ARN \
--tag-keys key1 key2
```

Removing tags (AWS SDKs)

To remove tags from an existing Amazon Personalize resource with the AWS SDKs, use the UntagResource API operation. The following code shows how to remove multiple tags from a dataset group with the SDK for Python (Boto3). For `resourceArn`, specify the Amazon Resource Name (ARN) of the resource. For `tagKeys`, specify the keys of the tags to be removed.

```
import boto3

personalize = boto3.client('personalize')

response = personalize.untag_resource(
    resourceArn="Resource ARN",
    tagKeys=["tag1Key", "tag2Key"]
)
```

Using tags in IAM policies

After you start implementing tags, you can apply tag-based, resource-level permissions to AWS Identity and Access Management (IAM) policies and API operations. This includes operations that support adding tags to resources when resources are created. By using tags in this way, you can implement granular control of which groups and users in your AWS account have permission to create and tag resources, and which groups and users have permission to create, update, and remove tags more generally.

For example, you can create a policy that allows a user to have full access to all of the Amazon Personalize resources where their name is a value in the `Owner` tag for the resource.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
       {
          "Sid": "ModifyResourceIfOwner",
          "Effect": "Allow",
          "Action": "personalize:*",
          "Resource": "*",
          "Condition": {
             "StringEqualsIgnoreCase": {
                "aws:ResourceTag/Owner": "${aws:username}"
             }
          }
       }
    ]
}
```

The following example shows how to create a policy to allow creating and deleting a dataset. These operations are allowed only if the user name is `johndoe`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "personalize:CreateDataset",
                "personalize:DeleteDataset"
            ],
            "Resource": "arn:aws:personalize:*:*:dataset/*",
            "Condition": {
                "StringEquals": {"aws:username" : "johndoe"}
            }
        },
        {
            "Effect": "Allow",
            "Action": "personalize:DescribeDataset",
            "Resource": "*"
        }
    ]
}
```

If you define tag-based, resource-level permissions, the permissions take effect immediately. This means that your resources are more secure as soon as they're created, and you can quickly start enforcing the use of tags for new resources. You can also use resource-level permissions to control which tag keys and values can be associated with new and existing resources. For more information, see Controlling Access Using Tags in the *AWS IAM User Guide*.

Troubleshooting

The following topics provide answers to common questions and troubleshooting advice for error messages that you might encounter with Amazon Personalize. For a quick reference to help you determine if Amazon Personalize fits your use case, see the Amazon Personalize Cheat Sheet in the Amazon Personalize samples repository.

Topics

- Frequently asked questions
- Error messages

Frequently asked questions

The following are answers to frequently asked questions related to importing data, training, model deployment, recommendations, and filters in Amazon Personalize.

For more questions and answers, see the Amazon Personalize Cheat Sheet in the Amazon Personalize samples repository.

Topics

- Data import and management
- Creating a custom solution and solution version
- Model deployment (custom campaigns)
- Recommendations
- Filtering recommendations

Data import and management

What format should my bulk data be in?

Your bulk data must be in comma-separated values (CSV) format. The first row of your CSV file must contain column headers. The column headers in your CSV file need to map to the schema to create the dataset. If your data includes any non-ASCII encoded characters, your CSV file must be encoded in UTF-8 format. Don't enclose headers in quotation marks ("). `TIMESTAMP` and `CREATION_TIMESTAMP` data must be in *UNIX epoch* time format. For more information on timestamp data, see Timestamp data. For more information about schemas, see Schemas.

For complete data format guidelines, see Data format guidelines. If you're not sure how to format your data, you can use Amazon SageMaker Data Wrangler (Data Wrangler) to prepare your data. For more information, see Preparing and importing data using Amazon SageMaker Data Wrangler.

How much training data do I need?

For all use cases (Domain dataset groups) and custom recipes, your interactions data must have the following:

- At minimum 1000 item interactions records from users interacting with items in your catalog. These interactions can be from bulk imports, or streamed events, or both.
- At minimum 25 unique user IDs with at least two item interactions for each.

For quality recommendations, we recommend that you have at minimum 50,000 item interactions from at least 1,000 users with two or more item interactions each.

You can start out with an empty Item interactions dataset and, when you have recorded enough data, create your recommender (Domain dataset group) or custom solution version using only new recorded events. Some recipes and use cases may have additional data requirements. For information on use case requirements, see Choosing a use case. For information on recipe requirements, see Choosing a recipe.

How do I update an item or user's attributes?

Use the Amazon Personalize console or the PutItems or PutUsers operations to import an item or user with the same item ID but with the modified attributes.

How do I delete an item or user?

Amazon Personalize doesn't support deleting a specific item or user. To make sure that an item or user doesn't appear in recommendations, use a filter to exclude items. For more information, see Filtering recommendations and user segments.

How do I delete a schema?

You can delete a schema only with the DeleteSchema operation. You can't use the Amazon Personalize console to delete a schema.

Creating a custom solution and solution version

What recipe should I use?

The Amazon Personalize recipe that you use depends on your use case. For information on matching use cases to recipes, see Choosing a recipe. The Amazon Personalize Cheat Sheet also includes use case and recipe information.

How often should I train?

We recommend using automatic training with at least a weekly training frequency. Automatic training makes it easier for you to maintain recommendation relevance. Your training frequency depends on your business requirements, the recipe that you use, and how frequently you import data. For more information, see Configuring automatic training. For information about maintaining relevance, see Maintaining recommendation relevance.

Should I use AutoML?

No, instead we recommend that you match your use case to different Amazon Personalize recipes and choose a recipe. For information on matching use cases to recipes, see Choosing a recipe.

Model deployment (custom campaigns)

What should I set for my campaign's minProvisionedTPS?

A high `minProvisionedTPS` will increase your cost. We recommend starting with 1 for `minProvisionedTPS` (the default). Track your usage using Amazon CloudWatch metrics, and increase the `minProvisionedTPS` as necessary.

How do I monitor the cost of my campaigns?

The Amazon Personalize Monitor project provides a CloudWatch dashboard, custom metrics, utilization alarms, and cost optimization functions for Amazon Personalize campaigns. See the Amazon Personalize Monitor in the Amazon Personalize samples repository.

How do I set a maximum transaction throughput for a campaign?

You can only set the *minimum* throughput for a campaign. When you create an Amazon Personalize campaign, you specify a dedicated transaction capacity for creating real-time recommendations for your application users. If your TPS increases beyond `minProvisionedTPS`, Amazon Personalize auto-scales the provisioned capacity up and down, but never below the `minProvisionedTPS`. For more information, see Minimum provisioned transactions per second and auto-scaling.

Recommendations

How can I tell if my Amazon Personalize model is generating quality recommendations?

Evaluate the performance of your solution version with offline and online metrics (see Evaluating a solution version with metrics) and online testing (such as A/B testing). For more information about A/B testing, see Measuring recommendation impact with A/B testing.

How do I delete my batch inference job and why is its status "active"?

You can't delete batch inference jobs. When a batch inference job's status is *active*, the job is complete. You can access your recommendations in the output Amazon S3 bucket or folder. You won't incur additional cost from the batch inference job once the job is complete. However you may incur additional charges from other services such as Amazon S3 for input and output data storage.

Why does my SIMS-backed campaign recommend items that are not similar based on metadata?

SIMS uses your Item interactions dataset to determine similarity; not item metadata such as color or price. SIMS identifies the co-occurrence of the item in user histories in your Interaction dataset to recommend similar items. For more information, see SIMS recipe.

Can I get more than 500 items from a single GetRecommendations API operation?

500 is the maximum number of items that you can retrieve in a single GetRecommendations. This value cannot be increased.

Filtering recommendations

Why aren't my recommendations filtered as expected?

This can occur for a variety of reasons:

- There may be issue with the format or syntax of your filter expression. For examples of correctly formatted filter expressions, see Filter expression examples.

- Amazon Personalize considers up to 100 of the most recent interactions per user per event type. This is an adjustable quota. You can request a quota increase using the Service Quotas console.

For more information, see Filtering recommendations and user segments.

How can I remove already purchased items from recommendations?

For ECOMMERCE Domain dataset groups, if you create a recommender with the Recommended for you or Customers who viewed X also viewed use case, Amazon Personalize automatically filters items the user purchased based on the userId that you specify and `Purchase` events.

For other Domain dataset group use cases or custom resources, use a filter to remove purchased items. Add a `Purchased` event type attribute to your data, record *Purchase* events with the `PutItems` operation, and create a filter that removes purchased items from recommendations. For example:

```
EXCLUDE ItemID WHERE Interactions.EVENT_TYPE IN ("purchased")
```

For more information, see Filtering recommendations and user segments.

Error messages

The following sections list and explain some of the messages that you might encounter when using Amazon Personalize.

Topics

- Data import and management
- Creating a solution and solution version (custom resources)
- Model deployment (custom campaigns)
- Recommenders (Domain dataset groups)
- Recommendations
- Filtering recommendations

Data import and management

Error message: *Invalid Data location.*

Make sure you used the correct syntax for your Amazon S3 bucket location. For dataset import jobs, use the following syntax for the location of your data in Amazon S3:

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

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

Error message: *An error occurred (LimitExceededException) when calling the CreateDatasetImportJob operation: More than 5 resources with PENDING or IN_PROGRESS status.*

You can have a total of 5 pending or in progress dataset import jobs per region. This quota is not adjustable. For a complete list of quotas for Amazon Personalize, see Amazon Personalize endpoints and quotas.

Error message: *Failed to create a data import job for <dataset type> dataset....Insufficient privileges for accessing data in Amazon S3.*

Give Amazon Personalize access to your Amazon S3 resources by attaching access policies to your Amazon S3 bucket and your Amazon Personalize service role. See Giving Amazon Personalize access to Amazon S3 resources.

If you use AWS Key Management Service (AWS KMS) for encryption, you must grant Amazon Personalize and your Amazon Personalize IAM service role permission to use your key. For more information, see Giving Amazon Personalize permission to use your AWS KMS key.

Error message: *Failed to create a data import job <dataset type> dataset...Input CSV is missing the following columns:[COLUMN_NAME, COLUMN_NAME].*

The data that you import into Amazon Personalize, including attribute names and data types, must match the destination dataset's schema. For more information, see Schemas.

Error message: *Length cannot be more than <character limit> characters for <COLLUMN_NAME>. If no values exceed the character limit, make sure your data follows the formatting guidelines listed in https://docs.aws.amazon.com/personalize/latest/dg/data-prep-formatting.html.*

Check to make sure all values in this column don't exceed the character limit. If no values exceed the character limit, check any preceding textual fields for the following:

- Make sure any textual data is wrapped in double quotes. Use the \ character to escape any double quotes or \ characters in your data.

- Makes sure each record in your CSV file is on a single line.

Creating a solution and solution version (custom resources)

Error message: *Create failed. Dataset has fewer than 25 users with at least 2 interactions each.*

You must import more data before you can train the model. The minimum data requirements to train a model are:

- At minimum 1000 item interactions records from users interacting with items in your catalog. These interactions can be from bulk imports, or streamed events, or both.
- At minimum 25 unique user IDs with at least two item interactions for each.

For real-time recommendations, import more data with a dataset import job or record more interaction *events* for your users with an event tracker and the PutEvents operation. For more information on recording real-time events, see Recording events.

For batch recommendations, import your data with a dataset import job when you have more data. For more information, about importing bulk data see Step 2: Preparing and importing data.

Model deployment (custom campaigns)

Error: *Cannot create a campaign. More than 5 resources in ACTIVE state. Please delete some and try again.*

You can have a total of 5 active Amazon Personalize campaigns per dataset group. This quota is adjustable and you can request a quota increase using the Service Quotas console. For a complete list of limits and quotas for Amazon Personalize, see Amazon Personalize endpoints and quotas.

Recommenders (Domain dataset groups)

Error: *Dataset has fewer than 1000 interactions after filtering by event type: <event type>*

Different use cases require different event types. Your data must have at minimum 1000 events with the required type for your use case. For more information, see Choosing a use case

Recommendations

Batch inference job error message: *Invalid S3 input path* or *Invalid S3 output path*

Make sure you use the correct syntax for your Amazon S3 input or output locations. Also make sure that your output location is different from your input data. It should be a folder in the same Amazon S3 bucket or a different bucket.

Use the following syntax for the *input* file location in Amazon S3: **`s3://<name of your S3 bucket>/<folder name>/<input JSON file name>`**

Use the following syntax for the *output* folder in Amazon S3: **`s3://<name of your S3 bucket>/<output folder name>/`**

Filtering recommendations

Error message: *Could not create filter. Invalid input symbol: $parameterName. Placeholders are not allowed with NOT_IN operator.*

You can't use placeholder parameters in a filter expression that uses the NOT_IN operator. Instead, use the IN operator and use the opposite Action: use Include instead of Exclude (or the reverse).

For example, if you want to use `INCLUDE ItemID WHERE Items.GENRE NOT IN ($GENRE)`, you can use `EXCLUDE ItemID WHERE Items.GENRE IN ($GENRE)` and get the same results.

For more information about filters, see Filter expression elements.

Error message: *Could not create filter. Invalid Expression...* when filtering on Boolean type fields

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`.

For more information about filters, see Filter expression elements.

Specifying resources with AWS CloudFormation

Amazon Personalize is integrated with AWS CloudFormation, a service that helps you to model and set up your AWS resources so that you can spend less time creating and managing your resources and infrastructure. You create a template that describes all the AWS resources that you can specify (such as Amazon Personalize dataset groups). AWS CloudFormation then provisions and configures those resources for you.

When you use AWS CloudFormation, you can reuse your template to set up your Amazon Personalize resources consistently and repeatedly. Describe your resources once, and then provision the same resources over and over in multiple AWS accounts and Regions.

Topics

Amazon Personalize and AWS CloudFormation templates

To provision and configure resources for Amazon Personalize and related services, you must understand AWS CloudFormation templates. Templates are formatted text files in JSON or YAML. These templates describe the resources that you want to provision in your AWS CloudFormation stacks. If you're unfamiliar with JSON or YAML, you can use AWS CloudFormation Designer to help you get started with AWS CloudFormation templates. For more information, see What is AWS CloudFormation Designer? in the *AWS CloudFormation User Guide*.

Amazon Personalize supports specifying datasets, dataset groups, dataset import jobs, schemas, and solutions in AWS CloudFormation. For more information, see the Amazon Personalize resource type reference in the *AWS CloudFormation User Guide*.

Example AWS CloudFormation templates for Amazon Personalize resources

The following AWS CloudFormation template examples show you how to specify different Amazon Personalize resources.

Topics

- CreateDatasetGroup
- CreateDataset
- CreateSchema
- CreateSolution

CreateDatasetGroup

JSON

```
{
   "AWSTemplateFormatVersion":"2010-09-09",
   "Resources":{
      "MyDatasetGroup": {
            "Type": "AWS::Personalize::DatasetGroup",
            "Properties": {
               "Name": "my-dataset-group-name"
            }
      }
   }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyDatasetGroup:
    Type: 'AWS::Personalize::DatasetGroup'
    Properties:
      Name: my-dataset-group-name
```

CreateDataset

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
```

```
    "MyDataset": {
      "Type": "AWS::Personalize::Dataset",
      "Properties": {
        "Name": "my-dataset-name",
        "DatasetType": "Interactions",
        "DatasetGroupArn": "arn:aws:personalize:us-west-2:123456789012:dataset-
group/dataset-group-name",
        "SchemaArn": "arn:aws:personalize:us-west-2:123456789012:schema/schema-
name",
        "DatasetImportJob": {
          "JobName": "my-import-job-name",
          "DataSource": {
            "DataLocation": "s3://bucket-name/file-name.csv"
          },
          "RoleArn": "arn:aws:iam::123456789012:role/personalize-role"
        }
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyDataset:
    Type: 'AWS::Personalize::Dataset'
    Properties:
      Name: my-dataset-name
      DatasetType: Interactions
      DatasetGroupArn: 'arn:aws:personalize:us-west-2:123456789012:dataset-group/
dataset-group-name'
      SchemaArn: 'arn:aws:personalize:us-west-2:123456789012:schema/schema-name'
      DatasetImportJob:
        JobName: my-import-job-name
        DataSource:
          DataLocation: 's3://bucket-name/file-name.csv'
        RoleArn: 'arn:aws:iam::123456789012:role/personalize-role'
```

CreateSchema

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "MySchema": {
            "Type": "AWS::Personalize::Schema",
            "Properties": {
                "Name": "my-schema-name",
                "Schema": "{\"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\"}"
            }
        }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MySchema:
    Type: AWS::Personalize::Schema
    Properties:
      Name: "my-schema-name"
      Schema: >-
        {"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"}
```

CreateSolution

JSON

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
```

```
        "MySolution": {
            "Type": "AWS::Personalize::Solution",
            "Properties": {
               "Name": "my-solution-name",
               "DatasetGroupArn": "arn:aws:personalize:us-
west-2:123456789012:dataset-group/my-dataset-group-name",
               "RecipeArn": "arn:aws:personalize:::recipe/aws-user-personalization",
               "SolutionConfig": {
                  "EventValueThreshold" : ".05"
                }
            }
         }
    }
}
```

YAML

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MySolution:
    Type: 'AWS::Personalize::Solution'
    Properties:
      Name: my-solution-name
      DatasetGroupArn: >-
        arn:aws:personalize:us-west-2:123456789012:dataset-group/my-dataset-group-
name
      RecipeArn: 'arn:aws:personalize:::recipe/aws-user-personalization'
      SolutionConfig:
        EventValueThreshold: '.05'
```

Learn more about AWS CloudFormation

To learn more about AWS CloudFormation, see the following resources:

- AWS CloudFormation
- AWS CloudFormation user guide
- AWS CloudFormation API reference
- AWS CloudFormation Command Line Interface user guide

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. 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

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. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 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 text 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 text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Data encryption

The following information explains where Amazon Personalize uses data encryption to protect your data.

Encryption at rest

Any data stored within Amazon Personalize is always encrypted at rest with Amazon Personalize managed AWS Key Management Service (AWS KMS) keys. If you provide your own AWS KMS key during resource creation, Amazon Personalize uses the key to encrypt your data and store it. For example, if you provide a AWS KMS ARN in the CreateDatasetGroup operation, Amazon Personalize uses the key to encrypt and store data you import into any datasets that you create in that dataset group.

You must grant Amazon Personalize and your Amazon Personalize IAM service role permission to use your key. For more information, see Giving Amazon Personalize permission to use your AWS KMS key.

For information about data encryption in Amazon S3 see Protecting data using encryption in the *Amazon Simple Storage Service User Guide*. For information about managing your own AWS KMS key, see Managing keys in the *AWS Key Management Service Developer Guide*.

Encryption in transit

Amazon Personalize uses TLS with AWS certificates to encrypt any data sent to other AWS services. Any communication with other AWS services happens over HTTPS, and Amazon Personalize endpoints support only secure connections over HTTPS.

Amazon Personalize copies data out of your account and processes it in an internal AWS system. When processing data, Amazon Personalize encrypts data with either a Amazon Personalize AWS KMS key or any AWS KMS key you provide.

Key management

AWS manages any default AWS KMS keys. It is your responsibility to manage any AWS KMS keys that you own. You must grant Amazon Personalize and your Amazon Personalize IAM service role permission to use your key. For more information, see Giving Amazon Personalize permission to use your AWS KMS key.

For information about managing your own AWS KMS key, see Managing keys in the *AWS Key Management Service Developer Guide*.

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

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.

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 service users 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.

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 Identity-based policy examples for Amazon Personalize.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might 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 Multi-factor authentication in the *AWS IAM Identity Center User Guide* and Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one 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 don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

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:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **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.
  - **Forward access sessions (FAS)** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.
  - **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 AWS 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 AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. 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**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

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 an 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, learn what IAM features are available to use with Amazon Personalize.

IAM features you can use with Amazon Personalize

IAM feature	Amazon Personalize support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	No
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Principal permissions	Yes
Service roles	Yes
Service-linked roles	No

To get a high-level view of how Amazon Personalize and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for Amazon Personalize

Supports identity-based policies	Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for Amazon Personalize

To view examples of Amazon Personalize identity-based policies, see Identity-based policy examples for Amazon Personalize.

Resource-based policies within Amazon Personalize

Supports resource-based policies	No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Policy actions for Amazon Personalize

Supports policy actions	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. 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.

To see a list of Amazon Personalize actions, see [Actions defined by Amazon Personalize](#) in the *Service Authorization Reference*.

Policy actions in Amazon Personalize use the following prefix before the action:

```
personalize
```

To specify multiple actions in a single statement, separate them with commas.

```
"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 view examples of Amazon Personalize identity-based policies, see [Identity-based policy examples for Amazon Personalize](#).

Policy resources for Amazon Personalize

Supports policy resources	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. 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": "*"
```

To see a list of Amazon Personalize resource types and their ARNs, see Resources defined by Amazon Personalize in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by Amazon Personalize.

To view examples of Amazon Personalize identity-based policies, see Identity-based policy examples for Amazon Personalize.

Policy condition keys for Amazon Personalize

Supports service-specific policy condition keys	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (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*.

To see a list of Amazon Personalize condition keys, see Condition keys for Amazon Personalize in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by Amazon Personalize.

To view examples of Amazon Personalize identity-based policies, see Identity-based policy examples for Amazon Personalize.

ACLs in Amazon Personalize

Supports ACLs	No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with Amazon Personalize

Supports ABAC (tags in policies)	Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

For more information about tagging Amazon Personalize resources, see Tagging Amazon Personalize resources.

To view an example identity-based policy for limiting access to a resource based on the tags on that resource, see Using tags in IAM policies.

Using temporary credentials with Amazon Personalize

Supports temporary credentials	Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for Amazon Personalize

Supports forward access sessions (FAS)	Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

Service roles for Amazon Personalize

Supports service roles	Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break Amazon Personalize functionality. Edit service roles only when Amazon Personalize provides guidance to do so.

Service-linked roles for Amazon Personalize

Supports service-linked roles	No

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Cross-service confused deputy prevention

The confused deputy problem is a security issue where an entity that doesn't have permission to perform an action can coerce a more-privileged entity to perform the action. In AWS, cross-service impersonation can result in the confused deputy problem. Cross-service impersonation can occur when one service (the *calling service*) calls another service (the *called service*). The calling service can be manipulated to use its permissions to act on another customer's resources in a way it should not otherwise have permission to access. To prevent this, AWS provides tools that help you protect your data for all services with service principals that have been given access to resources in your account.

We recommend using the `aws:SourceArn` and `aws:SourceAccount` global condition context keys in resource policies to limit the permissions that Amazon Personalize gives another service to the resource.

To prevent the confused deputy problem in roles assumed by Amazon Personalize, in the role's trust policy set the value of `aws:SourceArn` to `arn:aws:personalize:region:accountNumber:*`. The wildcard (*) applies the condition for all Amazon Personalize resources.

The following trust relationship policy grants Amazon Personalize access to your resources and uses the `aws:SourceArn` and `aws:SourceAccount` global condition context keys to prevent the confused deputy problem. Use this policy when you create a role for Amazon Personalize (Creating an IAM role for Amazon Personalize).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "personalize.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "accountNumber"
```

```
        },
        "StringLike": {
          "aws:SourceArn": "arn:aws:personalize:region:accountNumber:*"
        }
      }
    }
  ]
}
```

Identity-based policy examples for Amazon Personalize

By default, users and roles don't have permission to create or modify Amazon Personalize resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

For details about actions and resource types defined by Amazon Personalize, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon Personalize in the *Service Authorization Reference*.

Topics

- Policy best practices
- AWS managed policies
- Using the Amazon Personalize console
- Allow users to view their own permissions
- Allowing full access to Amazon Personalize resources
- Allowing read-only access to Amazon Personalize resources

Policy best practices

Identity-based policies 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 with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

AWS managed policies

AWS managed polices are policies that are created and managed by AWS. The following are examples of AWS managed policies you might use when working with Amazon Personalize.

AmazonPersonalizeFullAccess Policy

You can use the AWS managed `AmazonPersonalizeFullAccess` policy to give users the following permissions:

- 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

`AmazonPersonalizeFullAccess` provides more permissions than are necessary. We recommend creating a new IAM policy that only grants the necessary permissions (see Giving Amazon Personalize permission to access your resources).

CloudWatchFullAccess

To give your users permission to monitor Amazon Personalize with CloudWatch, attach the `CloudWatchFullAccess` policy to your role. For more information, see Monitoring Amazon Personalize.

The `CloudWatchFullAccess` policy is optional and grants permission for 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 (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

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:DescribeBatchSegmentJob",
                "personalize:DescribeCampaign",
                "personalize:DescribeDataset",
                "personalize:DescribeDatasetExportJob",
                "personalize:DescribeDatasetGroup",
                "personalize:DescribeDatasetImportJob",
                "personalize:DescribeEventTracker",
                "personalize:DescribeFeatureTransformation",
                "personalize:DescribeFilter",
                "personalize:DescribeRecipe",
                "personalize:DescribeRecommender",
                "personalize:DescribeSchema",
                "personalize:DescribeSolution",
                "personalize:DescribeSolutionVersion",
                "personalize:GetSolutionMetrics",
```

```
                "personalize:ListBatchInferenceJobs",
                "personalize:ListBatchSegmentJobs",
                "personalize:ListCampaigns",
                "personalize:ListDatasetExportJobs",
                "personalize:ListDatasetGroups",
                "personalize:ListDatasetImportJobs",
                "personalize:ListDatasets",
                "personalize:ListEventTrackers",
                "personalize:ListFilters",
                "personalize:ListRecipes",
                "personalize:ListRecommenders",
                "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
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my Amazon Personalize resources

I am not authorized to perform an action in Amazon Personalize

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `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, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `personalize:`*`GetWidget`* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to 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 that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to 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.
- 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

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

- Amazon Personalize serverless monitoring app example

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 `PutEventsRequests` metric.
How can I monitor the DatasetImportJob errors?	Use the `DatasetImportJobError` metric.
How can I monitor the latency of `GetRecommendations` calls?	Use the `GetRecommendationsLatency` 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.

The following table lists the Amazon Personalize metrics. All metrics except GetRecommendations and GetPersonalizedRanking support these statistics: `Average, Minimum, Maximum, Sum`. GetRecommendations and GetPersonalizedRanking support `Sum` only.

Metric	Description
DatasetImportJobRequests	The number of successful CreateDatasetImportJob API calls. Dimensions: DatasetGroupArn, DatasetArn, DatasetImportJobArn
DatasetImportJobError	The number of CreateDatasetImportJob API calls that resulted in an error. Dimensions: DatasetGroupArn, DatasetArn, DatasetImportJobArn
DatasetImportJobExecutionTime	The time between the CreateDatasetImportJob API call and the completion (or failure) of the operation. Dimensions: DatasetGroupArn, DatasetArn, DatasetImportJobArn Unit: Seconds
DatasetSize	The size of data imported by the dataset import job. Dimensions: DatasetGroupArn, DatasetArn, DatasetImportJobArn Unit: Bytes
SolutionTrainingJobRequests	The number of successful CreateSolutionVersion API calls. Dimensions: SolutionArn, SolutionVersionArn
SolutionTrainingJobError	The number of CreateSolutionVersion API calls that resulted in an error. Dimensions: SolutionArn, SolutionVersionArn

Metric	Description
SolutionTrainingJobExecutionTime	The time between the CreateSolutionVersion API call and the completion (or failure) of the operation. Dimensions: SolutionArn, SolutionVersionArn Unit: Seconds
GetPersonalizedRanking	Whether a GetPersonalizedRanking API call is successful. Use the sum statistic to view total count of successful GetPersonalizedRanking API calls. This metric doesn't support other statistics. 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

Metric	Description
GetRecommendations	Whether a GetRecommendations API calls is successful. Use the sum statistic to view total count of successful GetRecommendations API calls. This metric doesn't support other statistics. Dimension: CampaignArn
GetRecommendations 4xxErrors	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
PutEventsRequests	The number of successful PutEvents API calls. Dimension: DatasetGroupArn, DatasetArn, EventTrackerArn
PutEvents4xxErrors	The number of PutEvents API calls that returned a 4xx HTTP response code. Dimension: DatasetGroupArn, DatasetArn, EventTrackerArn

Metric	Description
`PutEvents5xxErrors`	The number of `PutEvents` API calls that returned a 5xx HTTP response code. Dimension: `DatasetGroupArn, DatasetArn, EventTrackerArn`
`PutEventLatency`	The time taken for the completion of the `PutEvents` API call (excludes 4xx and 5xx errors). Dimension: `DatasetGroupArn, DatasetArn, EventTrackerArn` Unit: Milliseconds
`PutItemsRequests`	The number of successful [PutItems](#) API calls. Dimension: `DatasetGroupArn, DatasetArn`
`PutItems4xxErrors`	The number of `PutItems` API calls that returned a 4xx HTTP response code. Dimension: `DatasetGroupArn, DatasetArn`
`PutItems5xxErrors`	The number of `PutItems` API calls that returned a 5xx HTTP response code. Dimension: `DatasetGroupArn, DatasetArn`
`PutItemsLatency`	The time taken for the completion of the `PutItems` API call (excludes 4xx and 5xx errors). Dimension: `DatasetGroupArn, DatasetArn` Unit: Milliseconds
`PutUsersRequests`	The number of successful [PutUsers](#) API calls. Dimension: `DatasetGroupArn, DatasetArn`

Metric	Description
`PutUsers4xxErrors`	The number of `PutUsers` API calls that returned a 4xx HTTP response code. Dimension: `DatasetGroupArn, DatasetArn`
`PutUsers5xxErrors`	The number of `PutUsers` API calls that returned a 5xx HTTP response code. Dimension: `DatasetGroupArn, DatasetArn`
`PutUsersLatency`	The time taken for the completion of the `PutUsers` API call (excludes 4xx and 5xx errors). Dimension: `DatasetGroupArn, DatasetArn` 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](https://docs.aws.amazon.com/awscloudtrail/latest/userguide/).

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.

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 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. Note that because the ListDatasetGroups API operation is an action that doesn't change state, the `responseElements` response is null. For more information about the body of CloudTrail records, see [CloudTrail record contents](#).


```
{
    "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": null,
    "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 Personalizeas 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 – Learn how you can use AWS to run sensitive workloads regulated under the U.S. Health Insurance Portability and Accountability Act (HIPAA).
- 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.

Amazon Personalize leverages the AWS global infrastructure for data resiliency. When you create an Amazon Personalize resource in an AWS Region, Amazon Personalize manages the resilience and data redundancy of the resource across multiple Availability Zones. For a list of AWS regions where you can create Amazon Personalize resources, see AWS regions and endpoints in the *Amazon Web Services General Reference*. 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 AWS global network security. For information about AWS security services and how AWS protects infrastructure, see AWS Cloud Security. To design your AWS environment using the best practices for infrastructure security, see Infrastructure Protection in *Security Pillar AWS Well-Architected Framework*.

You use AWS published API calls to access Amazon Personalize through the network. Clients must support the following:

- Transport Layer Security (TLS). We require TLS 1.2 and recommend TLS 1.3.
- Cipher suites with perfect forward secrecy (PFS) such as DHE (Ephemeral Diffie-Hellman) or ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Most modern systems such as Java 7 and later support these modes.

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 and interface VPC endpoints (AWS PrivateLink)

If you use Amazon Virtual Private Cloud (Amazon VPC) to host your AWS resources, you can establish a private connection between your VPC and Amazon Personalize. This connection allows Amazon Personalize to communicate with your resources on your VPC without going through the public internet.

Amazon VPC is an AWS service that you use to launch AWS resources in a virtual private cloud (VPC) or virtual network that you define. With a VPC, you have control over your network settings, such the IP address range, subnets, route tables, and network gateways. With VPC endpoints, the AWS network handles the routing between your VPC and AWS services.

To connect your VPC to Amazon Personalize, you define an interface VPC endpoint for Amazon Personalize. An interface endpoint is an elastic network interface with a private IP address that serves as an entry point for traffic destined to a supported AWS service. The endpoint provides reliable, scalable connectivity to Amazon Personalize. It doesn't require an internet gateway, a network address translation (NAT) instance, or a VPN connection. For more information, see What is Amazon VPC in the *Amazon VPC User Guide*.

Interface VPC endpoints are enabled by AWS PrivateLink. This AWS technology enables private communication between AWS services by using an elastic network interface with private IP addresses.

Note

All Amazon Personalize Federal Information Processing Standard (FIPS) endpoints are supported by AWS PrivateLink.

Considerations for Amazon Personalize VPC endpoints

Before you set up an interface VPC endpoint for Amazon Personalize, ensure that you review Interface endpoint properties and limitations in the *Amazon VPC User Guide*.

Amazon Personalize supports making calls to all of its API actions from your VPC.

Creating an interface VPC endpoint for Amazon Personalize

You can create a VPC endpoint for the Amazon Personalize service with either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Creating an interface endpoint in the *Amazon VPC User Guide*.

To create a VPC endpoint for Amazon Personalize, choose one of the following for the service:

- com.amazonaws.*region*.personalize
- com.amazonaws.*region*.personalize-events
- com.amazonaws.*region*.personalize-runtime

If you enable private DNS for the endpoint, you can make API requests to Amazon Personalize using its default DNS name for the Region, for example, `personalize.us-east-1.amazonaws.com`.

For more information, see Accessing a service through an interface endpoint in the *Amazon VPC User Guide*.

Creating a VPC endpoint policy for Amazon Personalize

You can attach an endpoint policy to your VPC endpoint that controls access to Amazon Personalize. The policy specifies the following information:

- The principal that can perform actions.

- The actions that can be performed.
- The resources on which actions can be performed.

For more information, see [Controlling access to services with VPC endpoints] in the *Amazon VPC User Guide*.

Example: VPC endpoint policy allowing all Amazon Personalize actions and passRole actions

When attached to an endpoint, this policy grants access to all Amazon Personalize actions and passRole actions.

```
{
    "Statement": [
        {
            "Principal": "*",
            "Effect": "Allow",
            "Action": [
                "personalize:*",
                "iam:PassRole"
            ],
            "Resource": "*"
        }
    ]
}
```

Example: VPC endpoint policy allowing Amazon Personalize ListDatasets actions

When attached to an endpoint, this policy grants access to the listed Amazon Personalize ListDatasets actions.

```
{
    "Statement": [
        {
            "Principal": "*",
            "Effect": "Allow",
            "Action": [
                "personalize:ListDatasets"
            ],
            "Resource": "*"
        }
    ]
```

```
}
```

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 Service Quotas console. For more information see Requesting a quota increase.

Topics

- Amazon Personalize endpoints and regions
- Compliance
- Service quotas
- Requesting a quota increase

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 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
Item interactions	
Minimum number of unique item interacti ons required to create a solution version or recommender. For a custom solution, you must have this many records after any filtering by event type or event value before training.	1000

Resource	Quota
Maximum number of item interactions that are considered by a model during training.	500 million (adjustable)
Maximum number of distinct event types combined with total number of optional metadata columns in an Item interactions dataset.	10
Maximum number of metadata columns, excluding reserved fields, in an Item interacti ons dataset.	5
Maximum number of characters for categorical data and impression values.	1000
Maximum amount of bulk item interactions data per dataset import job with FULL import mode.	100 GB (increases to 1TB with any increase to *Item interactions considered by a model*)
Maximum amount of bulk item interactions data per dataset import job with INCREMENT AL import mode.	1 GB
Minimum number of item interactions records per dataset import job with FULL or INCREMENTAL import mode.	1000
Users	
Minimum number of unique users in item interactions data, with at minimum 2 item interactions each, required to create a domain recommender or custom solution version.	25

Resource	Quota
Minimum percentage of total users that must have at minimum 2 item interactions or more before you can create a domain recommender or custom solution version.	1 percent
Maximum number of metadata fields for a Users dataset.	25
Maximum number of characters for USER_ID data values.	256
Maximum number of characters for categorical data values.	1000 characters
Maximum amount of bulk user data per dataset import job with FULL import mode.	100 GB
Maximum amount of bulk user data per dataset import job with INCREMENTAL import mode.	1 GB
Items	
Maximum number of items that are considere d by a model during training and generating recommendations.	750,000
Maximum number of metadata fields for an Items dataset.	100
Maximum number of characters for ITEM_ID data values.	256
Maximum number of characters for categorical data values.	1000 characters
Maximum number of textual fields for an Items dataset.	1

Resource	Quota
Maximum number of characters for textual data values for Chinese and Japanese languages.	7,000 characters
Maximum number of characters for textual data values for all other languages.	20,000 characters
Maximum amount of bulk items data per dataset import job with BULK import mode.	100 GB
Maximum amount of bulk item data per dataset import job with INCREMENTAL import mode.	1 GB
Actions	
Maximum number of actions that are considered by a model during training and generating recommendations.	1000
Maximum number of metadata fields for an Actions dataset.	10
Maximum number of characters for ACTION_ID data values.	256
Maximum number of characters for categorical data values.	1000 characters
Maximum amount of bulk actions data per dataset import job with BULK import mode.	100 GB
Maximum amount of bulk actions data per dataset import job with INCREMENTAL import mode.	1 GB
Action interactions	

Resource	Quota
Maximum number of action interactions that are considered by a model during training.	500 million
Maximum number of metadata columns, excluding reserved fields, in a Action interactions dataset.	5
Maximum amount of bulk interactions data per dataset import job with FULL import mode.	100 GB (increases to 1TB with any increase to *Action item interactions considered by a model*)
Maximum amount of bulk interactions data per dataset import job with INCREMENTAL import mode.	1 GB
Individual record import APIs	
Maximum rate of `PutEvents` requests per dataset group.	1000/second
Maximum number of events in a `PutEvents` call.	10
Maximum size of an event.	10 KB
Maximum rate of `PutActionInteractions` requests per dataset group.	1000/second
Maximum number of action interaction events in a `PutActionInteractions` call.	10
Maximum size of an action interaction event.	10 KB
Maximum rate of `PutItems` requests per dataset group.	10/second
Maximum number of items in a `PutItems` call.	10

Resource	Quota
Maximum rate of `PutUsers` requests per dataset group.	10/second
Maximum number of users in a `PutUsers` call.	10
Maximum rate of `PutActions` requests per dataset group.	10/second
Maximum number of users in a `PutActions` call.	10
Legacy recipes	
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).	80000
Minimum number of cold start items the HRNN-Coldstart recipe requires to train a model (create a solution version).	100
Filters	
Total number of filters per dataset group.	10
Maximum number of distinct dataset fields for a filter.	5
Total number of distinct dataset fields across all filters in a dataset group.	10
Maximum number of item interactions per user per event type considered by a filter.	100 interactions (adjustable)

Resource	Quota
Maximum number of action interactions per user per event type considered by a filter.	300 action interactions (adjustable)
GetRecommendations / GetPersonalizedRanking / GetActionRecommendations requests	
Maximum transaction rate for `GetRecommendations`, `GetActionRecommendations` and `GetPersonalizedRanking` requests.	2500/sec
Maximum number of `GetRecommendations` requests per second per campaign.	500/sec
Maximum number of `GetActionRecommendations` requests per second per campaign.	500/sec
Maximum number of `GetPersonalizedRanking` requests per second per campaign.	500/sec.
Maximum number of metadata columns per `GetRecommendations` or `GetPersonalizedRanking` request.	10
Maximum number of recommendation results for a `GetRecommendation` request without metadata.	500
Maximum number of recommendation results for a `GetRecommendation` request with metadata.	50
Maximum number of items for ranking in a `GetPersonalizedRanking` request without metadata.	500

Resource	Quota
Maximum number of items for ranking in a `GetPersonalizedRanking` request with metadata.	50
Metric attribution quotas	
Maximum number of metrics for a metric attribution	10
Maximum number of unique event attribution sources	100
Batch inference jobs	
Maximum number of input files for a batch inference job.	1000
Maximum size of batch inference job input.	1 GB
Maximum number of records per input file for a batch inference job without themes.	50 million
Maximum number of records per input file for a batch inference job with themes.	100
Batch segment jobs	
Maximum number of queries per input file for Item-Affinity recipe.	500
Maximum number of queries per input file for Item-Attribute-Affinity recipe.	10
Maximum number of users per segment	5 million

Your AWS account has the following quotas for each region.

Resource	Quota
Total number of active schemas.	500
Total number of active dataset groups.	5 (adjustable)
Total number of pending or in progress dataset import jobs.	5
Total number of pending or in progress batch inference jobs.	5 (adjustable)
Total number of pending or in progress batch segment jobs.	5
Total number of pending or in progress solution versions.	20 (adjustable)

Each dataset group has the following quotas.

Resource	Quota
Total number of active solutions.	10 (adjustable)
Total number of active campaigns.	5 (adjustable)
Total number of recommenders.	5
Total number of filters.	10 (adjustable)
Total number of distinct dataset fields across all filters.	10

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:

- Maximum number of item interactions that are considered by a model during training.
- Active campaigns per dataset group
- Active dataset groups
- Active filters per dataset group

- Active solutions per dataset group
- Amount of data per incremental import
- Maximum number of item interactions per user per event type considered by a filter
- Total number of pending or in progress batch inference jobs
- Total number of pending or in progress solution versions
- Maximum rate of `PutEvents` or `PutActionInteraction` 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

The following actions are supported by Amazon Personalize:

- CreateBatchInferenceJob
- CreateBatchSegmentJob
- CreateCampaign
- CreateDataset
- CreateDatasetExportJob
- CreateDatasetGroup
- CreateDatasetImportJob
- CreateEventTracker
- CreateFilter
- CreateMetricAttribution
- CreateRecommender
- CreateSchema
- CreateSolution
- CreateSolutionVersion
- DeleteCampaign
- DeleteDataset

- DeleteDatasetGroup
- DeleteEventTracker
- DeleteFilter
- DeleteMetricAttribution
- DeleteRecommender
- DeleteSchema
- DeleteSolution
- DescribeAlgorithm
- DescribeBatchInferenceJob
- DescribeBatchSegmentJob
- DescribeCampaign
- DescribeDataset
- DescribeDatasetExportJob
- DescribeDatasetGroup
- DescribeDatasetImportJob
- DescribeEventTracker
- DescribeFeatureTransformation
- DescribeFilter
- DescribeMetricAttribution
- DescribeRecipe
- DescribeRecommender
- DescribeSchema
- DescribeSolution
- DescribeSolutionVersion
- GetSolutionMetrics
- ListBatchInferenceJobs
- ListBatchSegmentJobs
- ListCampaigns
- ListDatasetExportJobs
- ListDatasetGroups

- ListDatasetImportJobs
- ListDatasets
- ListEventTrackers
- ListFilters
- ListMetricAttributionMetrics
- ListMetricAttributions
- ListRecipes
- ListRecommenders
- ListSchemas
- ListSolutions
- ListSolutionVersions
- ListTagsForResource
- StartRecommender
- StopRecommender
- StopSolutionVersionCreation
- TagResource
- UntagResource
- UpdateCampaign
- UpdateDataset
- UpdateMetricAttribution
- UpdateRecommender

The following actions are supported by Amazon Personalize Events:

- PutActionInteractions
- PutActions
- PutEvents
- PutItems
- PutUsers

The following actions are supported by Amazon Personalize Runtime:

- GetActionRecommendations
- GetPersonalizedRanking
- GetRecommendations

Amazon Personalize

The following actions are supported by Amazon Personalize:

- CreateBatchInferenceJob
- CreateBatchSegmentJob
- CreateCampaign
- CreateDataset
- CreateDatasetExportJob
- CreateDatasetGroup
- CreateDatasetImportJob
- CreateEventTracker
- CreateFilter
- CreateMetricAttribution
- CreateRecommender
- CreateSchema
- CreateSolution
- CreateSolutionVersion
- DeleteCampaign
- DeleteDataset
- DeleteDatasetGroup
- DeleteEventTracker
- DeleteFilter
- DeleteMetricAttribution
- DeleteRecommender
- DeleteSchema
- DeleteSolution
- DescribeAlgorithm

- DescribeBatchInferenceJob
- DescribeBatchSegmentJob
- DescribeCampaign
- DescribeDataset
- DescribeDatasetExportJob
- DescribeDatasetGroup
- DescribeDatasetImportJob
- DescribeEventTracker
- DescribeFeatureTransformation
- DescribeFilter
- DescribeMetricAttribution
- DescribeRecipe
- DescribeRecommender
- DescribeSchema
- DescribeSolution
- DescribeSolutionVersion
- GetSolutionMetrics
- ListBatchInferenceJobs
- ListBatchSegmentJobs
- ListCampaigns
- ListDatasetExportJobs
- ListDatasetGroups
- ListDatasetImportJobs
- ListDatasets
- ListEventTrackers
- ListFilters
- ListMetricAttributionMetrics
- ListMetricAttributions
- ListRecipes
- ListRecommenders

- ListSchemas
- ListSolutions
- ListSolutionVersions
- ListTagsForResource
- StartRecommender
- StopRecommender
- StopSolutionVersionCreation
- TagResource
- UntagResource
- UpdateCampaign
- UpdateDataset
- UpdateMetricAttribution
- UpdateRecommender

CreateBatchInferenceJob

Service: Amazon Personalize

Generates batch recommendations based on a list of items or users stored in Amazon S3 and exports the recommendations to an Amazon S3 bucket.

To generate batch recommendations, specify the ARN of a solution version and an Amazon S3 URI for the input and output data. For user personalization, popular items, and personalized ranking solutions, the batch inference job generates a list of recommended items for each user ID in the input file. For related items solutions, the job generates a list of recommended items for each item ID in the input file.

For more information, see Creating a batch inference job .

If you use the Similar-Items recipe, Amazon Personalize can add descriptive themes to batch recommendations. To generate themes, set the job's mode to `THEME_GENERATION` and specify the name of the field that contains item names in the input data.

For more information about generating themes, see Batch recommendations with themes from Content Generator .

You can't get batch recommendations with the Trending-Now or Next-Best-Action recipes.

Request Syntax

```
{
   "batchInferenceJobConfig": {
      "itemExplorationConfig": {
         "string" : "string"
      }
   },
   "batchInferenceJobMode": "string",
   "filterArn": "string",
   "jobInput": {
      "s3DataSource": {
         "kmsKeyArn": "string",
         "path": "string"
      }
   },
   "jobName": "string",
   "jobOutput": {
      "s3DataDestination": {
```

```
            "kmsKeyArn": "string",
            "path": "string"
         }
      },
      "numResults": number,
      "roleArn": "string",
      "solutionVersionArn": "string",
      "tags": [
         {
            "tagKey": "string",
            "tagValue": "string"
         }
      ],
      "themeGenerationConfig": {
         "fieldsForThemeGeneration": {
            "itemName": "string"
         }
      }
}
```

Request Parameters

The request accepts the following data in JSON format.

batchInferenceJobConfig

The configuration details of a batch inference job.

Type: BatchInferenceJobConfig object

Required: No

batchInferenceJobMode

The mode of the batch inference job. To generate descriptive themes for groups of similar items, set the job mode to THEME_GENERATION. If you don't want to generate themes, use the default BATCH_INFERENCE.

When you get batch recommendations with themes, you will incur additional costs. For more information, see Amazon Personalize pricing.

Type: String

Valid Values: BATCH_INFERENCE | THEME_GENERATION

Required: No

filterArn

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

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 object

Required: Yes

jobName

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

The path to the Amazon S3 bucket where the job's output will be stored.

Type: BatchInferenceJobOutput object

Required: Yes

numResults

The number of recommendations to retrieve.

Type: Integer

Required: No

roleArn

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-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

Required: Yes

solutionVersionArn

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-z\d-]+):personalize:.*:.*:.+`

Required: Yes

tags

A list of tags to apply to the batch inference job.

Type: Array of Tag objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

Required: No

themeGenerationConfig

For theme generation jobs, specify the name of the column in your Items dataset that contains each item's name.

Type: ThemeGenerationConfig object

Required: No

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

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

TooManyTagsException

You have exceeded the maximum number of tags you can apply to this 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateBatchSegmentJob

Service: Amazon Personalize

Creates a batch segment 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 and user segments.

Request Syntax

```
{
   "filterArn": "string",
   "jobInput": {
      "s3DataSource": {
         "kmsKeyArn": "string",
         "path": "string"
      }
   },
   "jobName": "string",
   "jobOutput": {
      "s3DataDestination": {
         "kmsKeyArn": "string",
         "path": "string"
      }
   },
   "numResults": number,
   "roleArn": "string",
   "solutionVersionArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

Request Parameters

The request accepts the following data in JSON format.

filterArn

The ARN of the filter to apply to the batch segment 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

The Amazon S3 path for the input data used to generate the batch segment job.

Type: BatchSegmentJobInput object

Required: Yes

jobName

The name of the batch segment 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

The Amazon S3 path for the bucket where the job's output will be stored.

Type: BatchSegmentJobOutput object

Required: Yes

numResults

The number of predicted users generated by the batch segment job for each line of input data. The maximum number of users per segment is 5 million.

Type: Integer

Required: No

roleArn

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-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

Required: Yes

solutionVersionArn

The Amazon Resource Name (ARN) of the solution version you want the batch segment job to use to generate batch segments.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

tags

A list of tags to apply to the batch segment job.

Type: Array of Tag objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

Required: No

Response Syntax

```
{
   "batchSegmentJobArn": "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.

batchSegmentJobArn

The ARN of the batch segment 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

TooManyTagsException

You have exceeded the maximum number of tags you can apply to this 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateCampaign

Service: Amazon Personalize

> **Important**
>
> You incur campaign costs while it is active. To avoid unnecessary costs, make sure to delete the campaign when you are finished. For information about campaign costs, see Amazon Personalize pricing.

Creates a campaign that deploys 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

> **Important**
>
> A high `minProvisionedTPS` will increase your cost. We recommend starting with 1 for `minProvisionedTPS` (the default). Track your usage using Amazon CloudWatch metrics, and increase the `minProvisionedTPS` as necessary.

When you create an Amazon Personalize campaign, you can specify the minimum provisioned transactions per second (`minProvisionedTPS`) for the campaign. This is the baseline transaction throughput for the campaign provisioned by Amazon Personalize. It sets the minimum billing charge for the campaign while it is active. A transaction is a single `GetRecommendations` or `GetPersonalizedRanking` request. The default `minProvisionedTPS` is 1.

If your TPS increases beyond the `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. When your traffic reduces, capacity returns to the `minProvisionedTPS`.

You are charged for the the minimum provisioned TPS or, if your requests exceed the `minProvisionedTPS`, the actual TPS. The actual TPS is the total number of recommendation requests you make. We recommend starting with a low `minProvisionedTPS`, track your usage using Amazon CloudWatch metrics, and then increase the `minProvisionedTPS` as necessary.

For more information about campaign costs, see Amazon Personalize pricing.

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.

> **Note**
>
> Wait until the `status` of the campaign is `ACTIVE` before asking the campaign for recommendations.

Related APIs

- ListCampaigns
- DescribeCampaign
- UpdateCampaign
- DeleteCampaign

Request Syntax

```
{
   "campaignConfig": {
      "enableMetadataWithRecommendations": boolean,
      "itemExplorationConfig": {
         "string" : "string"
      },
      "syncWithLatestSolutionVersion": boolean
   },
   "minProvisionedTPS": number,
   "name": "string",
   "solutionVersionArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
```

```
        }
    ]
}
```

Request Parameters

The request accepts the following data in JSON format.

campaignConfig

The configuration details of a campaign.

Type: CampaignConfig object

Required: No

minProvisionedTPS

Specifies the requested minimum provisioned transactions (recommendations) per second that Amazon Personalize will support. A high `minProvisionedTPS` will increase your bill. We recommend starting with 1 for `minProvisionedTPS` (the default). Track your usage using Amazon CloudWatch metrics, and increase the `minProvisionedTPS` as necessary.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

name

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

The Amazon Resource Name (ARN) of the trained model to deploy with the campaign. To specify the latest solution version of your solution, specify the ARN of your *solution* in `SolutionArn/$LATEST` format. You must use this format if you set `syncWithLatestSolutionVersion` to `True` in the [CampaignConfig](#).

To deploy a model that isn't the latest solution version of your solution, specify the ARN of the solution version.

For more information about automatic campaign updates, see [Enabling automatic campaign updates](#).

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

tags

A list of [tags](#) to apply to the campaign.

Type: Array of [Tag](#) objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

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

The Amazon Resource Name (ARN) of the campaign.

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

TooManyTagsException

You have exceeded the maximum number of tags you can apply to this 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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 to import your training data to a dataset.

There are 5 types of datasets:

- Item interactions
- Items
- Users
- Action interactions
- Actions

Each dataset type has an associated schema with required field types. Only the `Item 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.

Related APIs

- CreateDatasetGroup
- ListDatasets
- DescribeDataset
- DeleteDataset

Request Syntax

```
{
   "datasetGroupArn": "string",
   "datasetType": "string",
```

```
   "name": "string",
   "schemaArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn

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

The type of dataset.

One of the following (case insensitive) values:

- Interactions
- Items
- Users
- Actions
- Action_Interactions

Type: String

Length Constraints: Maximum length of 256.

Required: Yes

name

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

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

tags

A list of tags to apply to the dataset.

Type: Array of Tag objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

Required: No

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

The ARN of the dataset.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

LimitExceededException

The limit on the number of requests per second has been exceeded.

HTTP Status Code: 400

ResourceAlreadyExistsException

The specified resource already exists.

HTTP Status Code: 400

ResourceInUseException

The specified resource is in use.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

TooManyTagsException

You have exceeded the maximum number of tags you can apply to this 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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, 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",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetArn

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

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

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

The path to the Amazon S3 bucket where the job's output is stored.

Type: DatasetExportJobOutput object

Required: Yes

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-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

Required: Yes

tags

A list of tags to apply to the dataset export job.

Type: Array of Tag objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

Required: No

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

The Amazon Resource Name (ARN) of the dataset export 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

TooManyTagsException

You have exceeded the maximum number of tags you can apply to this 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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 is a container for Amazon Personalize resources. A dataset group can contain at most three datasets, one for each type of dataset:

- Item interactions
- Items
- Users
- Actions
- Action interactions

A dataset group can be a Domain dataset group, where you specify a domain and use pre-configured resources like recommenders, or a Custom dataset group, where you use custom resources, such as a solution with a solution version, that you deploy with a campaign. If you start with a Domain dataset group, you can still add custom resources such as solutions and solution versions trained with recipes for custom use cases and deployed with campaigns.

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. 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
- CreateEventTracker
- CreateSolution

Related APIs

- ListDatasetGroups
- DescribeDatasetGroup
- DeleteDatasetGroup

Request Syntax

```
{
   "domain": "string",
   "kmsKeyArn": "string",
   "name": "string",
   "roleArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

Request Parameters

The request accepts the following data in JSON format.

domain

The domain of the dataset group. Specify a domain to create a Domain dataset group. The domain you specify determines the default schemas for datasets and the use cases available for recommenders. If you don't specify a domain, you create a Custom dataset group with solution versions that you deploy with a campaign.

Type: String

Valid Values: ECOMMERCE | VIDEO_ON_DEMAND

Required: No

kmsKeyArn

The Amazon Resource Name (ARN) of a AWS Key Management Service (KMS) key used to encrypt the datasets.

Type: String

Length Constraints: Maximum length of 2048.

Pattern: `arn:aws.*:kms:.*:[0-9]{12}:key/.*`

Required: No

name

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

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-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

Required: No

tags

A list of tags to apply to the dataset group.

Type: Array of Tag objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

Required: No

Response Syntax

```
{
   "datasetGroupArn": "string",
   "domain": "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

The Amazon Resource Name (ARN) of the new dataset group.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

domain

The domain for the new Domain dataset group.

Type: String

Valid Values: `ECOMMERCE | VIDEO_ON_DEMAND`

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

TooManyTagsException

You have exceeded the maximum number of tags you can apply to this 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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.

If you already created a recommender or deployed a custom solution version with a campaign, how new bulk records influence recommendations depends on the domain use case or recipe that you use. For more information, see How new data influences real-time recommendations.

> **Important**
>
> By default, a dataset import job replaces any existing data in the dataset that you imported in bulk. To add new records without replacing existing data, specify INCREMENTAL for the import mode in the CreateDatasetImportJob operation.

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, 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
- DescribeDatasetImportJob

Request Syntax

```
{
   "datasetArn": "string",
   "dataSource": {
      "dataLocation": "string"
   },
   "importMode": "string",
   "jobName": "string",
   "publishAttributionMetricsToS3": boolean,
   "roleArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetArn

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

The Amazon S3 bucket that contains the training data to import.

Type: DataSource object

Required: Yes

importMode

Specify how to add the new records to an existing dataset. The default import mode is FULL. If you haven't imported bulk records into the dataset previously, you can only specify FULL.

- Specify FULL to overwrite all existing bulk data in your dataset. Data you imported individually is not replaced.
- Specify INCREMENTAL to append the new records to the existing data in your dataset. Amazon Personalize replaces any record with the same ID with the new one.

Type: String

Valid Values: FULL | INCREMENTAL

Required: No

jobName

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

publishAttributionMetricsToS3

If you created a metric attribution, specify whether to publish metrics for this import job to Amazon S3

Type: Boolean

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-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

Required: Yes

tags

A list of tags to apply to the dataset import job.

Type: Array of Tag objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

Required: No

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

The ARN of the dataset import job.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

LimitExceededException

The limit on the number of requests per second has been exceeded.

HTTP Status Code: 400

ResourceAlreadyExistsException

The specified resource already exists.

HTTP Status Code: 400

ResourceInUseException

The specified resource is in use.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

TooManyTagsException

You have exceeded the maximum number of tags you can apply to this 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3

- [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 Item 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.

> **Note**
>
> The event tracker must be in the ACTIVE state before using the tracking ID.

Related APIs

- ListEventTrackers
- DescribeEventTracker
- DeleteEventTracker

Request Syntax

```
{
   "datasetGroupArn": "string",
```

```
    "name": "string",
    "tags": [
       {
          "tagKey": "string",
          "tagValue": "string"
       }
    ]
}
```

Request Parameters

The request accepts the following data in JSON format.

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: Yes

name

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

tags

A list of tags to apply to the event tracker.

Type: Array of Tag objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

Required: No

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

The ARN of the event tracker.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

trackingId

The ID of the event tracker. Include this ID in requests to the PutEvents API.

Type: String

Length Constraints: Maximum length of 256.

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

LimitExceededException

The limit on the number of requests per second has been exceeded.

HTTP Status Code: 400

ResourceAlreadyExistsException

The specified resource already exists.

HTTP Status Code: 400

ResourceInUseException

The specified resource is in use.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

TooManyTagsException

You have exceeded the maximum number of tags you can apply to this 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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 and user segments.

Request Syntax

```
{
   "datasetGroupArn": "string",
   "filterExpression": "string",
   "name": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn

The ARN of the dataset group that the filter will belong to.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

filterExpression

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.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 2500.

Required: Yes

name

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

tags

A list of tags to apply to the filter.

Type: Array of Tag objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

Required: No

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

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

TooManyTagsException

You have exceeded the maximum number of tags you can apply to this 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateMetricAttribution

Service: Amazon Personalize

Creates a metric attribution. A metric attribution creates reports on the data that you import into Amazon Personalize. Depending on how you imported the data, you can view reports in Amazon CloudWatch or Amazon S3. For more information, see Measuring impact of recommendations.

Request Syntax

```
{
   "datasetGroupArn": "string",
   "metrics": [
      {
         "eventType": "string",
         "expression": "string",
         "metricName": "string"
      }
   ],
   "metricsOutputConfig": {
      "roleArn": "string",
      "s3DataDestination": {
         "kmsKeyArn": "string",
         "path": "string"
      }
   },
   "name": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn

The Amazon Resource Name (ARN) of the destination dataset group for the metric attribution.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

metrics

A list of metric attributes for the metric attribution. Each metric attribute specifies an event type to track and a function. Available functions are `SUM()` or `SAMPLECOUNT()`. For SUM() functions, provide the dataset type (either Interactions or Items) and column to sum as a parameter. For example SUM(Items.PRICE).

Type: Array of MetricAttribute objects

Array Members: Maximum number of 10 items.

Required: Yes

metricsOutputConfig

The output configuration details for the metric attribution.

Type: MetricAttributionOutput object

Required: Yes

name

A name for the metric attribution.

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

```
{
   "metricAttributionArn": "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.

metricAttributionArn

The Amazon Resource Name (ARN) for the new metric attribution.

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateRecommender

Service: Amazon Personalize

Creates a recommender with the recipe (a Domain dataset group use case) you specify. You create recommenders for a Domain dataset group and specify the recommender's Amazon Resource Name (ARN) when you make a GetRecommendations request.

Minimum recommendation requests per second

> **Important**
>
> A high `minRecommendationRequestsPerSecond` will increase your bill. We recommend starting with 1 for `minRecommendationRequestsPerSecond` (the default). Track your usage using Amazon CloudWatch metrics, and increase the `minRecommendationRequestsPerSecond` as necessary.

When you create a recommender, you can configure the recommender's minimum recommendation requests per second. The minimum recommendation requests per second (`minRecommendationRequestsPerSecond`) specifies the baseline recommendation request throughput provisioned by Amazon Personalize. The default minRecommendationRequestsPerSecond is 1. A recommendation request is a single `GetRecommendations` operation. Request throughput is measured in requests per second and Amazon Personalize uses your requests per second to derive your requests per hour and the price of your recommender usage.

If your requests per second increases beyond `minRecommendationRequestsPerSecond`, Amazon Personalize auto-scales the provisioned capacity up and down, but never below `minRecommendationRequestsPerSecond`. There's a short time delay while the capacity is increased that might cause loss of requests.

Your bill is the greater of either the minimum requests per hour (based on minRecommendationRequestsPerSecond) or the actual number of requests. The actual request throughput used is calculated as the average requests/second within a one-hour window. We recommend starting with the default `minRecommendationRequestsPerSecond`, track your usage using Amazon CloudWatch metrics, and then increase the `minRecommendationRequestsPerSecond` as necessary.

Status

A recommender can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- STOP PENDING > STOP IN_PROGRESS > INACTIVE > START PENDING > START IN_PROGRESS > ACTIVE
- DELETE PENDING > DELETE IN_PROGRESS

To get the recommender status, call DescribeRecommender.

Note

Wait until the `status` of the recommender is `ACTIVE` before asking the recommender for recommendations.

Related APIs

- ListRecommenders
- DescribeRecommender
- UpdateRecommender
- DeleteRecommender

Request Syntax

```
{
   "datasetGroupArn": "string",
   "name": "string",
   "recipeArn": "string",
   "recommenderConfig": {
      "enableMetadataWithRecommendations": boolean,
      "itemExplorationConfig": {
         "string" : "string"
      },
      "minRecommendationRequestsPerSecond": number,
      "trainingDataConfig": {
         "excludedDatasetColumns": {
            "string" : [ "string" ]
         }
      }
```

```
   },
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn

The Amazon Resource Name (ARN) of the destination domain dataset group for the recommender.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

name

The name of the recommender.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

Required: Yes

recipeArn

The Amazon Resource Name (ARN) of the recipe that the recommender will use. For a recommender, a recipe is a Domain dataset group use case. Only Domain dataset group use cases can be used to create a recommender. For information about use cases see Choosing recommender use cases.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

recommenderConfig

The configuration details of the recommender.

Type: RecommenderConfig object

Required: No

tags

A list of tags to apply to the recommender.

Type: Array of Tag objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

Required: No

Response Syntax

```
{
   "recommenderArn": "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.

recommenderArn

The Amazon Resource Name (ARN) of the recommender.

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

TooManyTagsException

You have exceeded the maximum number of tags you can apply to this 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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. If you are creating a schema for a dataset in a Domain dataset group, you provide the domain of the Domain dataset group. You specify a schema when you call CreateDataset.

For more information on schemas, see Datasets and schemas.

Related APIs

- ListSchemas
- DescribeSchema
- DeleteSchema

Request Syntax

```
{
   "domain": "string",
   "name": "string",
   "schema": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

domain

The domain for the schema. If you are creating a schema for a dataset in a Domain dataset group, specify the domain you chose when you created the Domain dataset group.

Type: String

Valid Values: `ECOMMERCE | VIDEO_ON_DEMAND`

Required: No

name

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

A schema in Avro JSON format.

Type: String

Length Constraints: Maximum length of 20000.

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

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateSolution

Service: Amazon Personalize

> **Important**
>
> After you create a solution, you can't change its configuration. By default, all new solutions use automatic training. With automatic training, you incur training costs while your solution is active. You can't stop automatic training for a solution. To avoid unnecessary costs, make sure to delete the solution when you are finished. For information about training costs, see Amazon Personalize pricing.

Creates the configuration for training a model (creating a solution version). This configuration includes the recipe to use for model training and optional training configuration, such as columns to use in training and feature transformation parameters. For more information about configuring a solution, see Creating and configuring a solution.

By default, new solutions use automatic training to create solution versions every 7 days. You can change the training frequency. Automatic solution version creation starts one hour after the solution is ACTIVE. If you manually create a solution version within the hour, the solution skips the first automatic training. For more information, see Configuring automatic training.

To turn off automatic training, set `performAutoTraining` to false. If you turn off automatic training, you must manually create a solution version by calling the CreateSolutionVersion operation.

After training starts, you can get the solution version's Amazon Resource Name (ARN) with the ListSolutionVersions API operation. To get its status, use the DescribeSolutionVersion.

After training completes you can evaluate model accuracy by calling GetSolutionMetrics. When you are satisfied with the solution version, you deploy it using CreateCampaign. The campaign provides recommendations to a client through the GetRecommendations API.

> **Note**
>
> Amazon Personalize doesn't support configuring the `hpoObjective` 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. If you use manual training, the status must be ACTIVE before you call `CreateSolutionVersion`.

Related APIs

- ListSolutions
- CreateSolutionVersion
- DescribeSolution
- DeleteSolution

- ListSolutionVersions
- DescribeSolutionVersion

Request Syntax

```
{
   "datasetGroupArn": "string",
   "eventType": "string",
   "name": "string",
   "performAutoML": boolean,
   "performAutoTraining": boolean,
   "performHPO": boolean,
   "recipeArn": "string",
   "solutionConfig": {
      "algorithmHyperParameters": {
         "string" : "string"
      },
      "autoMLConfig": {
         "metricName": "string",
         "recipeList": [ "string" ]
      },
      "autoTrainingConfig": {
```

```
            "schedulingExpression": "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"
         },
         "trainingDataConfig": {
```

```
            "excludedDatasetColumns": {
               "string" : [ "string" ]
            }
         }
      },
      "tags": [
         {
            "tagKey": "string",
            "tagValue": "string"
         }
      ]
}
```

Request Parameters

The request accepts the following data in JSON format.

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: Yes

eventType

When your 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

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

> **Important**
>
> We don't recommend enabling automated machine learning. Instead, match your use case to the available Amazon Personalize recipes. For more information, see Choosing a recipe.

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

performAutoTraining

Whether the solution uses automatic training to create new solution versions (trained models). The default is `True` and the solution automatically creates new solution versions every 7 days. You can change the training frequency by specifying a `schedulingExpression` in the `AutoTrainingConfig` as part of solution configuration. For more information about automatic training, see Configuring automatic training.

Automatic solution version creation starts one hour after the solution is ACTIVE. If you manually create a solution version within the hour, the solution skips the first automatic training.

After training starts, you can get the solution version's Amazon Resource Name (ARN) with the ListSolutionVersions API operation. To get its status, use the DescribeSolutionVersion.

Type: Boolean

Required: No

performHPO

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

The Amazon Resource Name (ARN) of the recipe to use for model training. This is required when `performAutoML` is false. For information about different Amazon Personalize recipes and their ARNs, see Choosing a recipe.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

solutionConfig

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 object

Required: No

tags

A list of tags to apply to the solution.

Type: Array of Tag objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

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

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

TooManyTagsException

You have exceeded the maximum number of tags you can apply to this 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

CreateSolutionVersion

Service: Amazon Personalize

Trains or retrains an active solution in a Custom dataset group. A solution is created using the CreateSolution 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. 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
- DescribeSolutionVersion
- ListSolutions
- CreateSolution
- DescribeSolution
- DeleteSolution

Request Syntax

```
{
```

```
    "name": "string",
    "solutionArn": "string",
    "tags": [
       {
          "tagKey": "string",
          "tagValue": "string"
       }
    ],
    "trainingMode": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

name

The name of the solution version.

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 containing the training configuration information.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

tags

A list of tags to apply to the solution version.

Type: Array of Tag objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

Required: No

trainingMode

The scope of training to be performed when creating the solution version. The default is FULL. This creates a completely new model based on the entirety of the training data from the datasets in your dataset group.

If you use User-Personalization, you can specify a training mode of UPDATE. This updates the model to consider new items for recommendations. It is not a full retraining. You should still complete a full retraining weekly. If you specify UPDATE, Amazon Personalize will stop automatic updates for the solution version. To resume updates, create a new solution with training mode set to FULL and deploy it in a campaign. For more information about automatic updates, see Automatic updates.

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 legacy HRNN-Coldstart recipe.

Type: String

Valid Values: FULL | UPDATE | AUTOTRAIN

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

The ARN of the new solution version.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

LimitExceededException

The limit on the number of requests per second has been exceeded.

HTTP Status Code: 400

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

TooManyTagsException

You have exceeded the maximum number of tags you can apply to this 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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 information on creating campaigns, see CreateCampaign.

Request Syntax

```
{
   "campaignArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

campaignArn

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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.

Request Syntax

```
{
   "datasetArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetArn

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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 dataset from the dataset group. For more information on event trackers, see CreateEventTracker.

Request Syntax

```
{
   "eventTrackerArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

eventTrackerArn

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteMetricAttribution

Service: Amazon Personalize

Deletes a metric attribution.

Request Syntax

```
{
   "metricAttributionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

metricAttributionArn

The metric attribution's Amazon Resource Name (ARN).

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DeleteRecommender

Service: Amazon Personalize

Deactivates and removes a recommender. A deleted recommender can no longer be specified in a GetRecommendations request.

Request Syntax

```
{
   "recommenderArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

recommenderArn

The Amazon Resource Name (ARN) of the recommender 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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.

Request Syntax

```
{
   "schemaArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

schemaArn

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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 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.

Request Syntax

```
{
   "solutionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

solutionArn

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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

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

A listing of the properties of the algorithm.

Type: Algorithm 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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

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"
         }
      },
      "batchInferenceJobMode": "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",
        "themeGenerationConfig": {
           "fieldsForThemeGeneration": {
              "itemName": "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

Information on the specified batch inference job.

Type: BatchInferenceJob 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeBatchSegmentJob

Service: Amazon Personalize

Gets the properties of a batch segment job including name, Amazon Resource Name (ARN), status, input and output configurations, and the ARN of the solution version used to generate segments.

Request Syntax

```
{
   "batchSegmentJobArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

batchSegmentJobArn

The ARN of the batch segment job to describe.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

Response Syntax

```
{
   "batchSegmentJob": {
      "batchSegmentJobArn": "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.

batchSegmentJob

Information on the specified batch segment job.

Type: BatchSegmentJob 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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.

Request Syntax

```
{
   "campaignArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

campaignArn

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": {
         "enableMetadataWithRecommendations": boolean,
         "itemExplorationConfig": {
            "string" : "string"
         },
         "syncWithLatestSolutionVersion": boolean
      },
      "creationDateTime": number,
      "failureReason": "string",
      "lastUpdatedDateTime": number,
      "latestCampaignUpdate": {
         "campaignConfig": {
            "enableMetadataWithRecommendations": boolean,
            "itemExplorationConfig": {
               "string" : "string"
            },
            "syncWithLatestSolutionVersion": boolean
         },
         "creationDateTime": number,
         "failureReason": "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

The properties of the campaign.

Type: Campaign 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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.

Request Syntax

```
{
   "datasetArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetArn

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,
      "latestDatasetUpdate": {
         "creationDateTime": number,
         "failureReason": "string",
         "lastUpdatedDateTime": number,
         "schemaArn": "string",
         "status": "string"
```

```
      },
      "name": "string",
      "schemaArn": "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.

dataset

A listing of the dataset's properties.

Type: Dataset 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 v2]
- [AWS SDK for Java V2]
- [AWS SDK for JavaScript V3]
- [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, including the export job status.

Request Syntax

```
{
   "datasetExportJobArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetExportJobArn

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

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 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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.

Request Syntax

```
{
   "datasetGroupArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn

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",
      "domain": "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

A listing of the dataset group's properties.

Type: DatasetGroup 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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, including the import job status.

Request Syntax

```
{
   "datasetImportJobArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetImportJobArn

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",
      "importMode": "string",
      "jobName": "string",
      "lastUpdatedDateTime": number,
```

```
        "publishAttributionMetricsToS3": boolean,
        "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

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 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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.

Request Syntax

```
{
   "eventTrackerArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

eventTrackerArn

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

An object that describes the event tracker.

Type: EventTracker 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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

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

A listing of the FeatureTransformation properties.

Type: FeatureTransformation 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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

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

The filter's details.

Type: Filter 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python

- AWS SDK for Ruby V3

DescribeMetricAttribution

Service: Amazon Personalize

Describes a metric attribution.

Request Syntax

```
{
   "metricAttributionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

metricAttributionArn

The metric attribution's Amazon Resource Name (ARN).

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

Response Syntax

```
{
   "metricAttribution": {
      "creationDateTime": number,
      "datasetGroupArn": "string",
      "failureReason": "string",
      "lastUpdatedDateTime": number,
      "metricAttributionArn": "string",
      "metricsOutputConfig": {
         "roleArn": "string",
         "s3DataDestination": {
            "kmsKeyArn": "string",
            "path": "string"
         }
```

```
      },
      "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.

metricAttribution

The details of the metric attribution.

Type: MetricAttribution 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 v2

- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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 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

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

An object that describes the recipe.

Type: Recipe 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

DescribeRecommender

Service: Amazon Personalize

Describes the given recommender, including its status.

A recommender can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- STOP PENDING > STOP IN_PROGRESS > INACTIVE > START PENDING > START IN_PROGRESS > ACTIVE
- DELETE PENDING > DELETE IN_PROGRESS

When the `status` is `CREATE FAILED`, the response includes the `failureReason` key, which describes why.

The `modelMetrics` key is null when the recommender is being created or deleted.

For more information on recommenders, see CreateRecommender.

Request Syntax

```
{
   "recommenderArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

recommenderArn

The Amazon Resource Name (ARN) of the recommender to describe.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

Response Syntax

```
{
   "recommender": {
      "creationDateTime": number,
      "datasetGroupArn": "string",
      "failureReason": "string",
      "lastUpdatedDateTime": number,
      "latestRecommenderUpdate": {
         "creationDateTime": number,
         "failureReason": "string",
         "lastUpdatedDateTime": number,
         "recommenderConfig": {
            "enableMetadataWithRecommendations": boolean,
            "itemExplorationConfig": {
               "string" : "string"
            },
            "minRecommendationRequestsPerSecond": number,
            "trainingDataConfig": {
               "excludedDatasetColumns": {
                  "string" : [ "string" ]
               }
            }
         },
         "status": "string"
      },
      "modelMetrics": {
         "string" : number
      },
      "name": "string",
      "recipeArn": "string",
      "recommenderArn": "string",
      "recommenderConfig": {
         "enableMetadataWithRecommendations": boolean,
         "itemExplorationConfig": {
            "string" : "string"
         },
         "minRecommendationRequestsPerSecond": number,
         "trainingDataConfig": {
            "excludedDatasetColumns": {
               "string" : [ "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.

recommender

The properties of the recommender.

Type: Recommender 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 v2
- AWS SDK for Java V2

- AWS SDK for JavaScript V3
- 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.

Request Syntax

```
{
   "schemaArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

schemaArn

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,
      "domain": "string",
      "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

The requested schema.

Type: DatasetSchema 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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.

Request Syntax

```
{
   "solutionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

solutionArn

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",
         "trainingMode": "string",
         "trainingType": "string"
      },
      "name": "string",
      "performAutoML": boolean,
      "performAutoTraining": boolean,
      "performHPO": boolean,
      "recipeArn": "string",
      "solutionArn": "string",
      "solutionConfig": {
         "algorithmHyperParameters": {
            "string" : "string"
         },
         "autoMLConfig": {
            "metricName": "string",
            "recipeList": [ "string" ]
         },
         "autoTrainingConfig": {
            "schedulingExpression": "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"
            },
            "trainingDataConfig": {
               "excludedDatasetColumns": {
                  "string" : [ "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

An object that describes the solution.

Type: Solution 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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

Request Syntax

```
{
   "solutionVersionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

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: Yes

Response Syntax

```
{
   "solutionVersion": {
      "creationDateTime": number,
      "datasetGroupArn": "string",
      "eventType": "string",
      "failureReason": "string",
      "lastUpdatedDateTime": number,
      "name": "string",
      "performAutoML": boolean,
      "performHPO": boolean,
      "recipeArn": "string",
      "solutionArn": "string",
      "solutionConfig": {
         "algorithmHyperParameters": {
```

```
            "string" : "string"
         },
         "autoMLConfig": {
            "metricName": "string",
            "recipeList": [ "string" ]
         },
         "autoTrainingConfig": {
            "schedulingExpression": "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"
         },
         "trainingDataConfig": {
            "excludedDatasetColumns": {
               "string" : [ "string" ]
            }
         }
      },
      "solutionVersionArn": "string",
      "status": "string",
      "trainingHours": number,
      "trainingMode": "string",
      "trainingType": "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

The solution version.

Type: SolutionVersion 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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

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

The metrics for the solution version. For more information, see Evaluating a solution version with metrics.

Type: String to double map

Map Entries: Maximum number of 100 items.

Key Length Constraints: Maximum length of 256.

solutionVersionArn

The same solution version ARN as specified in the request.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

ResourceInUseException

The specified resource is in use.

HTTP Status Code: 400

ResourceNotFoundException

Could not find the specified resource.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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

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

The token to request the next page of results.

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

solutionVersionArn

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",
         "batchInferenceJobMode": "string",
         "creationDateTime": number,
         "failureReason": "string",
         "jobName": "string",
         "lastUpdatedDateTime": number,
         "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

A list containing information on each job that is returned.

Type: Array of BatchInferenceJobSummary objects

Array Members: Maximum number of 100 items.

nextToken

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 1500.

Pattern: `\p{ASCII}{0,1500}`

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListBatchSegmentJobs

Service: Amazon Personalize

Gets a list of the batch segment jobs that have been performed off of a solution version that you specify.

Request Syntax

```
{
   "maxResults": number,
   "nextToken": "string",
   "solutionVersionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

maxResults

The maximum number of batch segment 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

The token to request the next page of results.

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

solutionVersionArn

The Amazon Resource Name (ARN) of the solution version that the batch segment jobs used to generate batch segments.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

Response Syntax

```
{
   "batchSegmentJobs": [
      {
         "batchSegmentJobArn": "string",
         "creationDateTime": number,
         "failureReason": "string",
         "jobName": "string",
         "lastUpdatedDateTime": number,
         "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.

batchSegmentJobs

A list containing information on each job that is returned.

Type: Array of BatchSegmentJobSummary objects

Array Members: Maximum number of 100 items.

nextToken

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 1500.

Pattern: `\p{ASCII}{0,1500}`

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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.

Request Syntax

```
{
   "maxResults": number,
   "nextToken": "string",
   "solutionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

maxResults

The maximum number of campaigns to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken

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 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

solutionArn

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",
         "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

A list of the campaigns.

Type: Array of CampaignSummary objects

Array Members: Maximum number of 100 items.

nextToken

A token for getting the next set of campaigns (if they exist).

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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. For more information on datasets, see CreateDataset.

Request Syntax

```
{
   "datasetArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetArn

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

The maximum number of dataset export jobs to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken

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 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

Response Syntax

```
{
   "datasetExportJobs": [
      {
         "creationDateTime": number,
         "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

The list of dataset export jobs.

Type: Array of DatasetExportJobSummary objects

Array Members: Maximum number of 100 items.

nextToken

A token for getting the next set of dataset export jobs (if they exist).

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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.

Request Syntax

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

Request Parameters

The request accepts the following data in JSON format.

maxResults

The maximum number of dataset groups to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken

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 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

Response Syntax

```
{
```

```
    "datasetGroups": [
       {
          "creationDateTime": number,
          "datasetGroupArn": "string",
          "domain": "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

The list of your dataset groups.

Type: Array of DatasetGroupSummary objects

Array Members: Maximum number of 100 items.

nextToken

A token for getting the next set of dataset groups (if they exist).

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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. For more information on datasets, see CreateDataset.

Request Syntax

```
{
   "datasetArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetArn

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

The maximum number of dataset import jobs to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken

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 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

Response Syntax

```
{
   "datasetImportJobs": [
      {
         "creationDateTime": number,
         "datasetImportJobArn": "string",
         "failureReason": "string",
         "importMode": "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

The list of dataset import jobs.

Type: Array of DatasetImportJobSummary objects

Array Members: Maximum number of 100 items.

nextToken

A token for getting the next set of dataset import jobs (if they exist).

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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.

Request Syntax

```
{
   "datasetGroupArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn

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

The maximum number of datasets to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken

A token returned from the previous call to `ListDatasets` for getting the next set of dataset import jobs (if they exist).

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

Response Syntax

```
{
   "datasets": [
      {
         "creationDateTime": number,
         "datasetArn": "string",
         "datasetType": "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

An array of `Dataset` objects. Each object provides metadata information.

Type: Array of DatasetSummary objects

Array Members: Maximum number of 100 items.

nextToken

A token for getting the next set of datasets (if they exist).

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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.

Request Syntax

```
{
   "datasetGroupArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn

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

The maximum number of event trackers to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken

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 1500.

Pattern: \p{ASCII}{0,1500}

Required: No

Response Syntax

```
{
   "eventTrackers": [
      {
         "creationDateTime": number,
         "eventTrackerArn": "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.

eventTrackers

A list of event trackers.

Type: Array of EventTrackerSummary objects

Array Members: Maximum number of 100 items.

nextToken

A token for getting the next set of event trackers (if they exist).

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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

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

The maximum number of filters to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken

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 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

Response Syntax

```
{
   "Filters": [
      {
         "creationDateTime": number,
         "datasetGroupArn": "string",
         "failureReason": "string",
         "filterArn": "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.

Filters

A list of returned filters.

Type: Array of FilterSummary objects

Array Members: Maximum number of 100 items.

nextToken

A token for getting the next set of filters (if they exist).

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListMetricAttributionMetrics

Service: Amazon Personalize

Lists the metrics for the metric attribution.

Request Syntax

```
{
   "maxResults": number,
   "metricAttributionArn": "string",
   "nextToken": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

maxResults

The maximum number of metrics to return in one page of results.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

metricAttributionArn

The Amazon Resource Name (ARN) of the metric attribution to retrieve attributes for.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

nextToken

Specify the pagination token from a previous request to retrieve the next page of results.

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

Response Syntax

```
{
   "metrics": [
      {
         "eventType": "string",
         "expression": "string",
         "metricName": "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.

metrics

The metrics for the specified metric attribution.

Type: Array of MetricAttribute objects

Array Members: Maximum number of 10 items.

nextToken

Specify the pagination token from a previous `ListMetricAttributionMetricsResponse` request to retrieve the next page of results.

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListMetricAttributions

Service: Amazon Personalize

Lists metric attributions.

Request Syntax

```
{
   "datasetGroupArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn

The metric attributions' dataset group Amazon Resource Name (ARN).

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

maxResults

The maximum number of metric attributions to return in one page of results.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken

Specify the pagination token from a previous request to retrieve the next page of results.

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

Response Syntax

```
{
   "metricAttributions": [
      {
         "creationDateTime": number,
         "failureReason": "string",
         "lastUpdatedDateTime": number,
         "metricAttributionArn": "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.

metricAttributions

The list of metric attributions.

Type: Array of MetricAttributionSummary objects

Array Members: Maximum number of 100 items.

nextToken

Specify the pagination token from a previous request to retrieve the next page of results.

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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

```
{
   "domain": "string",
   "maxResults": number,
   "nextToken": "string",
   "recipeProvider": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

domain

Filters returned recipes by domain for a Domain dataset group. Only recipes (Domain dataset group use cases) for this domain are included in the response. If you don't specify a domain, all recipes are returned.

Type: String

Valid Values: `ECOMMERCE | VIDEO_ON_DEMAND`

Required: No

maxResults

The maximum number of recipes to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken

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 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

recipeProvider

The default is `SERVICE`.

Type: String

Valid Values: `SERVICE`

Required: No

Response Syntax

```
{
   "nextToken": "string",
   "recipes": [
      {
         "creationDateTime": number,
         "domain": "string",
         "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

A token for getting the next set of recipes.

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

recipes

The list of available recipes.

Type: Array of RecipeSummary 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3

- AWS SDK for Python
- AWS SDK for Ruby V3

ListRecommenders

Service: Amazon Personalize

Returns a list of recommenders in a given Domain dataset group. When a Domain dataset group is not specified, all the recommenders associated with the account are listed. The response provides the properties for each recommender, including the Amazon Resource Name (ARN). For more information on recommenders, see CreateRecommender.

Request Syntax

```
{
   "datasetGroupArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

datasetGroupArn

The Amazon Resource Name (ARN) of the Domain dataset group to list the recommenders for. When a Domain dataset group is not specified, all the recommenders associated with the account are listed.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

maxResults

The maximum number of recommenders to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken

A token returned from the previous call to `ListRecommenders` for getting the next set of recommenders (if they exist).

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

Response Syntax

```
{
   "nextToken": "string",
   "recommenders": [
      {
         "creationDateTime": number,
         "datasetGroupArn": "string",
         "lastUpdatedDateTime": number,
         "name": "string",
         "recipeArn": "string",
         "recommenderArn": "string",
         "recommenderConfig": {
            "enableMetadataWithRecommendations": boolean,
            "itemExplorationConfig": {
               "string" : "string"
            },
            "minRecommendationRequestsPerSecond": number,
            "trainingDataConfig": {
               "excludedDatasetColumns": {
                  "string" : [ "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

A token for getting the next set of recommenders (if they exist).

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

recommenders

A list of the recommenders.

Type: Array of RecommenderSummary 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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.

Request Syntax

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

Request Parameters

The request accepts the following data in JSON format.

maxResults

The maximum number of schemas to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken

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 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

Response Syntax

```
{
```

```
    "nextToken": "string",
    "schemas": [
       {
          "creationDateTime": number,
          "domain": "string",
          "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

A token used to get the next set of schemas (if they exist).

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

schemas

A list of schemas.

Type: Array of DatasetSchemaSummary 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListSolutions

Service: Amazon Personalize

Returns a list of solutions in a 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.

Request Syntax

```
{
   "datasetGroupArn": "string",
   "maxResults": number,
   "nextToken": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

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

maxResults

The maximum number of solutions to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken

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 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

Response Syntax

```
{
   "nextToken": "string",
   "solutions": [
      {
         "creationDateTime": number,
         "lastUpdatedDateTime": number,
         "name": "string",
         "recipeArn": "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

A token for getting the next set of solutions (if they exist).

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

solutions

A list of the current solutions.

Type: Array of SolutionSummary 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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).

Request Syntax

```
{
   "maxResults": number,
   "nextToken": "string",
   "solutionArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

maxResults

The maximum number of solution versions to return.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

Required: No

nextToken

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 1500.

Pattern: `\p{ASCII}{0,1500}`

Required: No

solutionArn

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,
         "failureReason": "string",
         "lastUpdatedDateTime": number,
         "solutionVersionArn": "string",
         "status": "string",
         "trainingMode": "string",
         "trainingType": "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

A token for getting the next set of solution versions (if they exist).

Type: String

Length Constraints: Maximum length of 1500.

Pattern: `\p{ASCII}{0,1500}`

solutionVersions

A list of solution versions describing the version properties.

Type: Array of SolutionVersionSummary 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

ListTagsForResource

Service: Amazon Personalize

Get a list of tags attached to a resource.

Request Syntax

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

Request Parameters

The request accepts the following data in JSON format.

resourceArn

The resource's Amazon Resource Name (ARN).

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

Response Syntax

```
{
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "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.

tags

The resource's tags.

Type: Array of Tag objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3

- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

StartRecommender

Service: Amazon Personalize

Starts a recommender that is INACTIVE. Starting a recommender does not create any new models, but resumes billing and automatic retraining for the recommender.

Request Syntax

```
{
   "recommenderArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

recommenderArn

The Amazon Resource Name (ARN) of the recommender to start.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

Response Syntax

```
{
   "recommenderArn": "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.

recommenderArn

The Amazon Resource Name (ARN) of the recommender you started.

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

StopRecommender

Service: Amazon Personalize

Stops a recommender that is ACTIVE. Stopping a recommender halts billing and automatic retraining for the recommender.

Request Syntax

```
{
   "recommenderArn": "string"
}
```

Request Parameters

The request accepts the following data in JSON format.

recommenderArn

The Amazon Resource Name (ARN) of the recommender to stop.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

Response Syntax

```
{
   "recommenderArn": "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.

recommenderArn

The Amazon Resource Name (ARN) of the recommender you stopped.

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

TagResource

Service: Amazon Personalize

Add a list of tags to a resource.

Request Syntax

```
{
   "resourceArn": "string",
   "tags": [
      {
         "tagKey": "string",
         "tagValue": "string"
      }
   ]
}
```

Request Parameters

The request accepts the following data in JSON format.

resourceArn

The resource's Amazon Resource Name (ARN).

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

tags

Tags to apply to the resource. For more information see Tagging Amazon Personalize resources.

Type: Array of Tag objects

Array Members: Minimum number of 0 items. Maximum number of 200 items.

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

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

TooManyTagsException

You have exceeded the maximum number of tags you can apply to this 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UntagResource

Service: Amazon Personalize

Removes the specified tags that are attached to a resource. For more information, see Removing tags from Amazon Personalize resources.

Request Syntax

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

Request Parameters

The request accepts the following data in JSON format.

resourceArn

The resource's Amazon Resource Name (ARN).

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

tagKeys

The keys of the tags to be removed.

Type: Array of strings

Array Members: Minimum number of 0 items. Maximum number of 200 items.

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

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

TooManyTagKeysException

The request contains more tag keys than can be associated with a resource (50 tag keys per 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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3

- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateCampaign

Service: Amazon Personalize

Updates a campaign to deploy a retrained solution version with an existing campaign, change your campaign's `minProvisionedTPS`, or modify your campaign's configuration. For example, you can set `enableMetadataWithRecommendations` to true for an existing campaign.

To update a campaign to start automatically using the latest solution version, specify the following:

- For the `SolutionVersionArn` parameter, specify the Amazon Resource Name (ARN) of your solution in `SolutionArn/$LATEST` format.
- In the `campaignConfig`, set `syncWithLatestSolutionVersion` to `true`.

To update a campaign, the campaign status must be ACTIVE or CREATE FAILED. Check the campaign status using the DescribeCampaign operation.

> **Note**
>
> You can still get recommendations from a campaign while an update is in progress. The campaign will use the previous solution version and campaign configuration to generate recommendations until the latest campaign update status is `Active`.

For more information about updating a campaign, including code samples, see Updating a campaign. For more information about campaigns, see Creating a campaign.

Request Syntax

```
{
   "campaignArn": "string",
   "campaignConfig": {
      "enableMetadataWithRecommendations": boolean,
      "itemExplorationConfig": {
         "string" : "string"
      },
      "syncWithLatestSolutionVersion": boolean
   },
   "minProvisionedTPS": number,
   "solutionVersionArn": "string"
```

```
}
```

Request Parameters

The request accepts the following data in JSON format.

campaignArn

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

The configuration details of a campaign.

Type: CampaignConfig object

Required: No

minProvisionedTPS

Specifies the requested minimum provisioned transactions (recommendations) per second that Amazon Personalize will support. A high `minProvisionedTPS` will increase your bill. We recommend starting with 1 for `minProvisionedTPS` (the default). Track your usage using Amazon CloudWatch metrics, and increase the `minProvisionedTPS` as necessary.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

solutionVersionArn

The Amazon Resource Name (ARN) of a new model to deploy. To specify the latest solution version of your solution, specify the ARN of your *solution* in `SolutionArn/$LATEST` format. You must use this format if you set `syncWithLatestSolutionVersion` to `True` in the CampaignConfig.

To deploy a model that isn't the latest solution version of your solution, specify the ARN of the solution version.

For more information about automatic campaign updates, see Enabling automatic campaign updates.

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

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateDataset

Service: Amazon Personalize

Update a dataset to replace its schema with a new or existing one. For more information, see [Replacing a dataset's schema](#).

Request Syntax

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

Request Parameters

The request accepts the following data in JSON format.

datasetArn

The Amazon Resource Name (ARN) of the dataset that you want to update.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

schemaArn

The Amazon Resource Name (ARN) of the new schema you want use.

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

The Amazon Resource Name (ARN) of the dataset you updated.

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateMetricAttribution

Service: Amazon Personalize

Updates a metric attribution.

Request Syntax

```
{
   "addMetrics": [
      {
         "eventType": "string",
         "expression": "string",
         "metricName": "string"
      }
   ],
   "metricAttributionArn": "string",
   "metricsOutputConfig": {
      "roleArn": "string",
      "s3DataDestination": {
         "kmsKeyArn": "string",
         "path": "string"
      }
   },
   "removeMetrics": [ "string" ]
}
```

Request Parameters

The request accepts the following data in JSON format.

addMetrics

Add new metric attributes to the metric attribution.

Type: Array of MetricAttribute objects

Array Members: Maximum number of 10 items.

Required: No

metricAttributionArn

The Amazon Resource Name (ARN) for the metric attribution to update.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

metricsOutputConfig

An output config for the metric attribution.

Type: MetricAttributionOutput object

Required: No

removeMetrics

Remove metric attributes from the metric attribution.

Type: Array of strings

Array Members: Maximum number of 10 items.

Length Constraints: Maximum length of 256.

Required: No

Response Syntax

```
{
   "metricAttributionArn": "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.

metricAttributionArn

The Amazon Resource Name (ARN) for the metric attribution that you updated.

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

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 v2
- AWS SDK for Java V2

- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

UpdateRecommender

Service: Amazon Personalize

Updates the recommender to modify the recommender configuration. If you update the recommender to modify the columns used in training, Amazon Personalize automatically starts a full retraining of the models backing your recommender. While the update completes, you can still get recommendations from the recommender. The recommender uses the previous configuration until the update completes. To track the status of this update, use the `latestRecommenderUpdate` returned in the DescribeRecommender operation.

Request Syntax

```
{
   "recommenderArn": "string",
   "recommenderConfig": {
      "enableMetadataWithRecommendations": boolean,
      "itemExplorationConfig": {
         "string" : "string"
      },
      "minRecommendationRequestsPerSecond": number,
      "trainingDataConfig": {
         "excludedDatasetColumns": {
            "string" : [ "string" ]
         }
      }
   }
}
```

Request Parameters

The request accepts the following data in JSON format.

recommenderArn

The Amazon Resource Name (ARN) of the recommender to modify.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

recommenderConfig

The configuration details of the recommender.

Type: RecommenderConfig object

Required: Yes

Response Syntax

```
{
   "recommenderArn": "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.

recommenderArn

The same recommender Amazon Resource Name (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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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:

- PutActionInteractions
- PutActions
- PutEvents
- PutItems
- PutUsers

PutActionInteractions

Service: Amazon Personalize Events

Records action interaction event data. An *action interaction* event is an interaction between a user and an *action*. For example, a user taking an action, such a enrolling in a membership program or downloading your app.

For more information about recording action interactions, see Recording action interaction events. For more information about actions in an Actions dataset, see Actions dataset.

Request Syntax

```
POST /action-interactions HTTP/1.1
Content-type: application/json

{
   "actionInteractions": [
      {
         "actionId": "string",
         "eventId": "string",
         "eventType": "string",
         "impression": [ "string" ],
         "properties": "string",
         "recommendationId": "string",
         "sessionId": "string",
         "timestamp": number,
         "userId": "string"
      }
   ],
   "trackingId": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

actionInteractions

A list of action interaction events from the session.

Type: Array of ActionInteraction objects

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Required: Yes

trackingId

The ID of your action interaction event tracker. When you create an Action interactions dataset, Amazon Personalize creates an action interaction event tracker for you. For more information, see Action interaction event tracker ID.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

PutActions

Service: Amazon Personalize Events

Adds one or more actions to an Actions dataset. For more information see Importing actions individually.

Request Syntax

```
POST /actions HTTP/1.1
Content-type: application/json

{
   "actions": [
      {
         "actionId": "string",
         "properties": "string"
      }
   ],
   "datasetArn": "string"
}
```

URI Request Parameters

The request does not use any URI parameters.

Request Body

The request accepts the following data in JSON format.

actions

A list of action data.

Type: Array of Action objects

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Required: Yes

datasetArn

The Amazon Resource Name (ARN) of the Actions dataset you are adding the action or actions to.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

PutEvents

Service: Amazon Personalize Events

Records item interaction event data. For more information see Recording item interaction events.

Request Syntax

```
POST /events HTTP/1.1
Content-type: application/json

{
   "eventList": [
      {
         "eventId": "string",
         "eventType": "string",
         "eventValue": number,
         "impression": [ "string" ],
         "itemId": "string",
         "metricAttribution": {
            "eventAttributionSource": "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

A list of event data from the session.

Type: Array of Event objects

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Required: Yes

sessionId

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 item interaction events.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: Yes

trackingId

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

The user associated with the event.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: No

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS Command Line Interface
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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 individually.

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

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

A list of item data.

Type: Array of Item objects

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

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 v2]
- [AWS SDK for Java V2]
- [AWS SDK for JavaScript V3]
- [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 individually.

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

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

A list of user data.

Type: Array of User objects

Array Members: Minimum number of 1 item. Maximum number of 10 items.

Required: Yes

Response Syntax

```
HTTP/1.1 200
```

Response Elements

If the action is successful, the service sends back an HTTP 200 response with an empty HTTP body.

Errors

InvalidInputException

Provide a valid value for the field or parameter.

HTTP Status Code: 400

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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:

- GetActionRecommendations
- GetPersonalizedRanking
- GetRecommendations

GetActionRecommendations

Service: Amazon Personalize Runtime

Returns a list of recommended actions in sorted in descending order by prediction score. Use the `GetActionRecommendations` API if you have a custom campaign that deploys a solution version trained with a PERSONALIZED_ACTIONS recipe.

For more information about PERSONALIZED_ACTIONS recipes, see PERSONALIZED_ACTIONS recipes. For more information about getting action recommendations, see Getting action recommendations.

Request Syntax

```
POST /action-recommendations HTTP/1.1
Content-type: application/json

{
   "campaignArn": "string",
   "filterArn": "string",
   "filterValues": {
      "string" : "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

The Amazon Resource Name (ARN) of the campaign to use for getting action recommendations. This campaign must deploy a solution version trained with a PERSONALIZED_ACTIONS recipe.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

filterArn

The ARN of the filter to apply to the returned recommendations. For more information, see Filtering Recommendations.

When using this parameter, be sure the filter resource is `ACTIVE`.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

filterValues

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 actions, you must provide values for all parameters that are defined in the expression. For filters with expressions that use an `EXCLUDE` element to exclude actions, 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 and user segments.

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

numResults

The number of results to return. The default is 5. The maximum is 100.

Type: Integer

Valid Range: Minimum value of 0.

Required: No

userId

The user ID of the user to provide action recommendations for.

Type: String

Length Constraints: Maximum length of 256.

Required: No

Response Syntax

```
HTTP/1.1 200
Content-type: application/json

{
   "actionList": [
      {
         "actionId": "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.

actionList

A list of action recommendations sorted in descending order by prediction score. There can be a maximum of 100 actions in the list. For information about action scores, see How action recommendation scoring works.

Type: Array of [PredictedAction](#) objects

recommendationId

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

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" ],
   "metadataColumns": {
      "string" : [ "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

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

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

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

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

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. If you are including metadata in recommendations, the maximum is 50. Otherwise, the maximum is 500.

Type: Array of strings

Length Constraints: Maximum length of 256.

Required: Yes

metadataColumns

If you enabled metadata in recommendations when you created or updated the campaign, specify metadata columns from your Items dataset to include in the personalized ranking. The map key is `ITEMS` and the value is a list of column names from your Items dataset. The maximum number of columns you can provide is 10.

For information about enabling metadata for a campaign, see [Enabling metadata in recommendations for a campaign](#).

Type: String to array of strings map

Map Entries: Maximum number of 1 item.

Key Length Constraints: Maximum length of 256.

Array Members: Maximum number of 99 items.

Length Constraints: Maximum length of 150.

Required: No

userId

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",
         "metadata": {
            "string" : "string"
         },
         "promotionName": "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

A list of items in order of most likely interest to the user. The maximum is 500.

Type: Array of PredictedItem objects

recommendationId

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 v2
- AWS SDK for Java V2

- [AWS SDK for JavaScript V3]
- [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. For campaigns, the campaign's Amazon Resource Name (ARN) is required and the required user and item 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 API.

For recommenders, the recommender's ARN is required and the required item and user input depends on the use case (domain-based recipe) backing the recommender. For information on use case requirements see Choosing recommender use cases.

Request Syntax

```
POST /recommendations HTTP/1.1
Content-type: application/json

{
   "campaignArn": "string",
   "context": {
      "string" : "string"
   },
   "filterArn": "string",
   "filterValues": {
      "string" : "string"
   },
   "itemId": "string",
   "metadataColumns": {
      "string" : [ "string" ]
   },
   "numResults": number,
   "promotions": [
```

```
      {
         "filterArn": "string",
         "filterValues": {
            "string" : "string"
         },
         "name": "string",
         "percentPromotedItems": number
      }
   ],
   "recommenderArn": "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

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: No

context

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

The ARN of the filter to apply to the returned recommendations. For more information, see Filtering Recommendations.

When using this parameter, be sure the filter resource is `ACTIVE`.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

filterValues

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 and user segments.

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

The item ID to provide recommendations for.

Required for `RELATED_ITEMS` recipe type.

Type: String

Length Constraints: Maximum length of 256.

Required: No

metadataColumns

If you enabled metadata in recommendations when you created or updated the campaign or recommender, specify the metadata columns from your Items dataset to include in item recommendations. The map key is `ITEMS` and the value is a list of column names from your Items dataset. The maximum number of columns you can provide is 10.

For information about enabling metadata for a campaign, see Enabling metadata in recommendations for a campaign. For information about enabling metadata for a recommender, see Enabling metadata in recommendations for a recommender.

Type: String to array of strings map

Map Entries: Maximum number of 1 item.

Key Length Constraints: Maximum length of 256.

Array Members: Maximum number of 99 items.

Length Constraints: Maximum length of 150.

Required: No

numResults

The number of results to return. The default is 25. If you are including metadata in recommendations, the maximum is 50. Otherwise, the maximum is 500.

Type: Integer

Valid Range: Minimum value of 0.

Required: No

promotions

The promotions to apply to the recommendation request. A promotion defines additional business rules that apply to a configurable subset of recommended items.

Type: Array of Promotion objects

Array Members: Maximum number of 1 item.

Required: No

recommenderArn

The Amazon Resource Name (ARN) of the recommender to use to get recommendations. Provide a recommender ARN if you created a Domain dataset group with a recommender for a domain use case.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

userId

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",
         "metadata": {
            "string" : "string"
         },
         "promotionName": "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

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 objects

recommendationId

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 v2
- AWS SDK for Java V2
- AWS SDK for JavaScript V3
- 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
- AlgorithmImage
- AutoMLConfig
- AutoMLResult
- AutoTrainingConfig
- BatchInferenceJob
- BatchInferenceJobConfig
- BatchInferenceJobInput
- BatchInferenceJobOutput
- BatchInferenceJobSummary
- BatchSegmentJob
- BatchSegmentJobInput

- BatchSegmentJobOutput
- BatchSegmentJobSummary
- Campaign
- CampaignConfig
- CampaignSummary
- CampaignUpdateSummary
- CategoricalHyperParameterRange
- ContinuousHyperParameterRange
- Dataset
- DatasetExportJob
- DatasetExportJobOutput
- DatasetExportJobSummary
- DatasetGroup
- DatasetGroupSummary
- DatasetImportJob
- DatasetImportJobSummary
- DatasetSchema
- DatasetSchemaSummary
- DatasetSummary
- DatasetUpdateSummary
- DataSource
- DefaultCategoricalHyperParameterRange
- DefaultContinuousHyperParameterRange
- DefaultHyperParameterRanges
- DefaultIntegerHyperParameterRange
- EventTracker
- EventTrackerSummary
- FeatureTransformation
- FieldsForThemeGeneration
- Filter

- FilterSummary
- HPOConfig
- HPOObjective
- HPOResourceConfig
- HyperParameterRanges
- IntegerHyperParameterRange
- MetricAttribute
- MetricAttribution
- MetricAttributionOutput
- MetricAttributionSummary
- OptimizationObjective
- Recipe
- RecipeSummary
- Recommender
- RecommenderConfig
- RecommenderSummary
- RecommenderUpdateSummary
- S3DataConfig
- Solution
- SolutionConfig
- SolutionSummary
- SolutionVersion
- SolutionVersionSummary
- Tag
- ThemeGenerationConfig
- TrainingDataConfig
- TunedHPOParams

The following data types are supported by Amazon Personalize Events:

- Action

- ActionInteraction
- Event
- Item
- MetricAttribution
- User

The following data types are supported by Amazon Personalize Runtime:

- PredictedAction
- PredictedItem
- Promotion

Amazon Personalize

The following data types are supported by Amazon Personalize:

- Algorithm
- AlgorithmImage
- AutoMLConfig
- AutoMLResult
- AutoTrainingConfig
- BatchInferenceJob
- BatchInferenceJobConfig
- BatchInferenceJobInput
- BatchInferenceJobOutput
- BatchInferenceJobSummary
- BatchSegmentJob
- BatchSegmentJobInput
- BatchSegmentJobOutput
- BatchSegmentJobSummary
- Campaign
- CampaignConfig

- CampaignSummary
- CampaignUpdateSummary
- CategoricalHyperParameterRange
- ContinuousHyperParameterRange
- Dataset
- DatasetExportJob
- DatasetExportJobOutput
- DatasetExportJobSummary
- DatasetGroup
- DatasetGroupSummary
- DatasetImportJob
- DatasetImportJobSummary
- DatasetSchema
- DatasetSchemaSummary
- DatasetSummary
- DatasetUpdateSummary
- DataSource
- DefaultCategoricalHyperParameterRange
- DefaultContinuousHyperParameterRange
- DefaultHyperParameterRanges
- DefaultIntegerHyperParameterRange
- EventTracker
- EventTrackerSummary
- FeatureTransformation
- FieldsForThemeGeneration
- Filter
- FilterSummary
- HPOConfig
- HPOObjective
- HPOResourceConfig

- [HyperParameterRanges]
- [IntegerHyperParameterRange]
- [MetricAttribute]
- [MetricAttribution]
- [MetricAttributionOutput]
- [MetricAttributionSummary]
- [OptimizationObjective]
- [Recipe]
- [RecipeSummary]
- [Recommender]
- [RecommenderConfig]
- [RecommenderSummary]
- [RecommenderUpdateSummary]
- [S3DataConfig]
- [Solution]
- [SolutionConfig]
- [SolutionSummary]
- [SolutionVersion]
- [SolutionVersionSummary]
- [Tag]
- [ThemeGenerationConfig]
- [TrainingDataConfig]
- [TunedHPOParams]

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 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 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:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

trainingInputMode

The training input mode.

Type: String

Length Constraints: Maximum length of 256.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for 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 Java V2
- AWS SDK for Ruby V3

AutoMLConfig

Service: Amazon Personalize

When the solution performs AutoML (`performAutoML` is true in [CreateSolution]), 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 Java V2
- AWS SDK for Ruby V3

AutoMLResult

Service: Amazon Personalize

When the solution performs AutoML (`performAutoML` is true in CreateSolution), 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 Java V2
- AWS SDK for Ruby V3

AutoTrainingConfig

Service: Amazon Personalize

The automatic training configuration to use when `performAutoTraining` is true.

Contents

schedulingExpression

Specifies how often to automatically train new solution versions. Specify a rate expression in rate(*value unit*) format. For value, specify a number between 1 and 30. For unit, specify `day` or `days`. For example, to automatically create a new solution version every 5 days, specify `rate(5 days)`. The default is every 7 days.

For more information about auto training, see Creating and configuring a solution.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 16.

Pattern: `rate\(\d+ days?\)`

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 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](#) object

Required: No

batchInferenceJobMode

The job's mode.

Type: String

Valid Values: `BATCH_INFERENCE | THEME_GENERATION`

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 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 object

Required: No

lastUpdatedDateTime

The time at which the batch inference job was last updated.

Type: Timestamp

Required: No

numResults

The number of recommendations generated by the batch inference job. This number includes the error messages generated for failed input records.

Type: Integer

Required: No

roleArn

The ARN of the Amazon Identity and Access Management (IAM) role that requested the batch inference job.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-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-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

themeGenerationConfig

The job's theme generation settings.

Type: ThemeGenerationConfig 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 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 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 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 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 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 Java V2
- AWS SDK for Ruby V3

BatchInferenceJobSummary

Service: Amazon Personalize

A truncated version of the BatchInferenceJob. The ListBatchInferenceJobs operation returns a list of batch inference job summaries.

Contents

batchInferenceJobArn

The Amazon Resource Name (ARN) of the batch inference job.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

batchInferenceJobMode

The job's mode.

Type: String

Valid Values: `BATCH_INFERENCE | THEME_GENERATION`

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-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 Java V2
- AWS SDK for Ruby V3

BatchSegmentJob

Service: Amazon Personalize

Contains information on a batch segment job.

Contents

batchSegmentJobArn

The Amazon Resource Name (ARN) of the batch segment job.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

creationDateTime

The time at which the batch segment job was created.

Type: Timestamp

Required: No

failureReason

If the batch segment job failed, the reason for the failure.

Type: String

Required: No

filterArn

The ARN of the filter used on the batch segment 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 segment job.

Type: [BatchSegmentJobInput](#) object

Required: No

jobName

The name of the batch segment 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 segment job.

Type: [BatchSegmentJobOutput](#) object

Required: No

lastUpdatedDateTime

The time at which the batch segment job last updated.

Type: Timestamp

Required: No

numResults

The number of predicted users generated by the batch segment job for each line of input data. The maximum number of users per segment is 5 million.

Type: Integer

Required: No

roleArn

The ARN of the Amazon Identity and Access Management (IAM) role that requested the batch segment job.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

Required: No

solutionVersionArn

The Amazon Resource Name (ARN) of the solution version used by the batch segment job to generate batch segments.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

status

The status of the batch segment 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 Java V2]
- [AWS SDK for Ruby V3]

BatchSegmentJobInput

Service: Amazon Personalize

The input configuration of a batch segment job.

Contents

s3DataSource

The configuration details of an Amazon S3 input or output bucket.

Type: S3DataConfig 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 Java V2
- AWS SDK for Ruby V3

BatchSegmentJobOutput

Service: Amazon Personalize

The output configuration parameters of a batch segment job.

Contents

s3DataDestination

The configuration details of an Amazon S3 input or output bucket.

Type: S3DataConfig 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 Java V2
- AWS SDK for Ruby V3

BatchSegmentJobSummary

Service: Amazon Personalize

A truncated version of the BatchSegmentJob datatype. ListBatchSegmentJobs operation returns a list of batch segment job summaries.

Contents

batchSegmentJobArn

The Amazon Resource Name (ARN) of the batch segment job.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

creationDateTime

The time at which the batch segment job was created.

Type: Timestamp

Required: No

failureReason

If the batch segment job failed, the reason for the failure.

Type: String

Required: No

jobName

The name of the batch segment 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 segment job was last updated.

Type: Timestamp

Required: No

solutionVersionArn

The Amazon Resource Name (ARN) of the solution version used by the batch segment job to generate batch segments.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

status

The status of the batch segment 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 Java V2]
- [AWS SDK for Ruby V3]

Campaign

Service: Amazon Personalize

An object that describes the deployment of a solution version. For more information on campaigns, see CreateCampaign.

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 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 API.

Type: CampaignUpdateSummary object

Required: No

minProvisionedTPS

Specifies the requested minimum provisioned transactions (recommendations) per second. A high `minProvisionedTPS` will increase your bill. We recommend starting with 1 for `minProvisionedTPS` (the default). Track your usage using Amazon CloudWatch metrics, and increase the `minProvisionedTPS` as necessary.

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 the solution version the campaign uses.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

status

The status of the campaign.

A campaign can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

Type: String

Length Constraints: Maximum length of 256.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

CampaignConfig

Service: Amazon Personalize

The configuration details of a campaign.

Contents

enableMetadataWithRecommendations

Whether metadata with recommendations is enabled for the campaign. If enabled, you can specify the columns from your Items dataset in your request for recommendations. Amazon Personalize returns this data for each item in the recommendation response. For information about enabling metadata for a campaign, see Enabling metadata in recommendations for a campaign.

If you enable metadata in recommendations, you will incur additional costs. For more information, see Amazon Personalize pricing.

Type: Boolean

Required: No

itemExplorationConfig

Specifies 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

syncWithLatestSolutionVersion

Whether the campaign automatically updates to use the latest solution version (trained model) of a solution. If you specify `True`, you must specify the ARN of your *solution* for the

`SolutionVersionArn` parameter. It must be in `SolutionArn/$LATEST` format. The default is `False` and you must manually update the campaign to deploy the latest solution version.

For more information about automatic campaign updates, see Enabling automatic campaign updates.

Type: Boolean

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 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 API.

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

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: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

Required: No

status

The status of the campaign.

A campaign can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

Type: String

Length Constraints: Maximum length of 256.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for 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 API.

Contents

campaignConfig

The configuration details of a campaign.

Type: CampaignConfig object

Required: No

creationDateTime

The date and time (in Unix time) that the campaign update was created.

Type: Timestamp

Required: No

failureReason

If a campaign update fails, the reason behind the failure.

Type: String

Required: No

lastUpdatedDateTime

The date and time (in Unix time) that the campaign update was last updated.

Type: Timestamp

Required: No

minProvisionedTPS

Specifies the requested minimum provisioned transactions (recommendations) per second that Amazon Personalize will support.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

solutionVersionArn

The Amazon Resource Name (ARN) of the deployed solution version.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

status

The status of the campaign update.

A campaign update can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

Type: String

Length Constraints: Maximum length of 256.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for 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 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 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
- Actions
- Action_Interactions

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

latestDatasetUpdate

Describes the latest update to the dataset.

Type: DatasetUpdateSummary object

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-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

trackingId

The ID of the event tracker for an Action interactions dataset. You specify the tracker's ID in the `PutActionInteractions` API operation. Amazon Personalize uses it to direct new data to the Action interactions dataset in your dataset group.

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 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.

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 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-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 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 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 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 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-z\d-]+):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 Java V2
- AWS SDK for Ruby V3

DatasetGroup

Service: Amazon Personalize

A dataset group is a collection of related datasets (Item interactions, Users, Items, Actions, Action interactions). You create a dataset group by calling CreateDatasetGroup. You then create a dataset and add it to a dataset group by calling CreateDataset. The dataset group is used to create and train a solution by calling CreateSolution. 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

domain

The domain of a Domain dataset group.

Type: String

Valid Values: `ECOMMERCE | VIDEO_ON_DEMAND`

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

Length Constraints: Maximum length of 2048.

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 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-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 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 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

domain

The domain of a Domain dataset group.

Type: String

Valid Values: `ECOMMERCE | VIDEO_ON_DEMAND`

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 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.

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 object

Required: No

failureReason

If a dataset import job fails, provides the reason why.

Type: String

Required: No

importMode

The import mode used by the dataset import job to import new records.

Type: String

Valid Values: `FULL | INCREMENTAL`

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

publishAttributionMetricsToS3

Whether the job publishes metrics to Amazon S3 for a metric attribution.

Type: Boolean

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-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 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](#) API.

Contents

creationDateTime

The date and time (in Unix time) that the dataset import job was created.

Type: Timestamp

Required: No

datasetImportJobArn

The Amazon Resource Name (ARN) of the dataset import job.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

failureReason

If a dataset import job fails, the reason behind the failure.

Type: String

Required: No

importMode

The import mode the dataset import job used to update the data in the dataset. For more information see [Updating existing bulk data](#).

Type: String

Valid Values: `FULL | INCREMENTAL`

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 Java V2
- AWS SDK for Ruby V3

DatasetSchema

Service: Amazon Personalize

Describes the schema for a dataset. For more information on schemas, see CreateSchema.

Contents

creationDateTime

The date and time (in Unix time) that the schema was created.

Type: Timestamp

Required: No

domain

The domain of a schema that you created for a dataset in a Domain dataset group.

Type: String

Valid Values: `ECOMMERCE | VIDEO_ON_DEMAND`

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 20000.

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 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 API.

Contents

creationDateTime

The date and time (in Unix time) that the schema was created.

Type: Timestamp

Required: No

domain

The domain of a schema that you created for a dataset in a Domain dataset group.

Type: String

Valid Values: `ECOMMERCE | VIDEO_ON_DEMAND`

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 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 API.

Contents

creationDateTime

The date and time (in Unix time) that the dataset was created.

Type: Timestamp

Required: No

datasetArn

The Amazon Resource Name (ARN) of the dataset.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

datasetType

The dataset type. One of the following values:

- Interactions
- Items
- Users
- Event-Interactions

Type: String

Length Constraints: Maximum length of 256.

Required: No

lastUpdatedDateTime

The date and time (in Unix time) that the dataset was last updated.

Type: Timestamp

Required: No

name

The name of the dataset.

Type: String

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 Java V2
- AWS SDK for Ruby V3

DatasetUpdateSummary

Service: Amazon Personalize

Describes an update to a dataset.

Contents

creationDateTime

The creation date and time (in Unix time) of the dataset update.

Type: Timestamp

Required: No

failureReason

If updating a dataset fails, provides the reason why.

Type: String

Required: No

lastUpdatedDateTime

The last update date and time (in Unix time) of the dataset.

Type: Timestamp

Required: No

schemaArn

The Amazon Resource Name (ARN) of the schema that replaced the previous schema of the dataset.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

status

The status of the dataset update.

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 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 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 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 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 objects

Array Members: Maximum number of 100 items.

Required: No

continuousHyperParameterRanges

The continuous hyperparameters and their default ranges.

Type: Array of DefaultContinuousHyperParameterRange objects

Array Members: Maximum number of 100 items.

Required: No

integerHyperParameterRanges

The integer-valued hyperparameters and their default ranges.

Type: Array of DefaultIntegerHyperParameterRange 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 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 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 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 API.

Contents

creationDateTime

The date and time (in Unix time) that the event tracker was created.

Type: Timestamp

Required: No

eventTrackerArn

The Amazon Resource Name (ARN) of the event tracker.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

lastUpdatedDateTime

The date and time (in Unix time) that the event tracker was last updated.

Type: Timestamp

Required: No

name

The name of the event tracker.

Type: String

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 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-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 Java V2
- AWS SDK for Ruby V3

FieldsForThemeGeneration

Service: Amazon Personalize

A string to string map of the configuration details for theme generation.

Contents

itemName

The name of the Items dataset column that stores the name of each item in the dataset.

Type: String

Length Constraints: Maximum length of 150.

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 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.

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 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-z\d-]+):personalize:.*:.*:.+`

Required: No

failureReason

If the filter failed, the reason for the failure.

Type: String

Required: No

filterArn

The ARN of the filter.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

lastUpdatedDateTime

The time at which the filter was last updated.

Type: Timestamp

Required: No

name

The name of the filter.

Type: String

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 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 object

Required: No

hpoObjective

The metric to optimize during HPO.

> **Note**
> Amazon Personalize doesn't support configuring the `hpoObjective` at this time.

Type: HPOObjective object

Required: No

hpoResourceConfig

Describes the resource configuration for HPO.

Type: HPOResourceConfig 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 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 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 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 objects

Array Members: Maximum number of 100 items.

Required: No

continuousHyperParameterRanges

The continuous hyperparameters and their ranges.

Type: Array of ContinuousHyperParameterRange objects

Array Members: Maximum number of 100 items.

Required: No

integerHyperParameterRanges

The integer-valued hyperparameters and their ranges.

Type: Array of IntegerHyperParameterRange 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 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 Java V2

- AWS SDK for Ruby V3

MetricAttribute

Service: Amazon Personalize

Contains information on a metric that a metric attribution reports on. For more information, see [Measuring impact of recommendations](#).

Contents

eventType

The metric's event type.

Type: String

Length Constraints: Maximum length of 256.

Required: Yes

expression

The attribute's expression. Available functions are `SUM()` or `SAMPLECOUNT()`. For SUM() functions, provide the dataset type (either Interactions or Items) and column to sum as a parameter. For example SUM(Items.PRICE).

Type: String

Length Constraints: Maximum length of 256.

Required: Yes

metricName

The metric's name. The name helps you identify the metric in Amazon CloudWatch or Amazon S3.

Type: String

Length Constraints: Maximum length of 256.

Required: Yes

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

MetricAttribution

Service: Amazon Personalize

Contains information on a metric attribution. A metric attribution creates reports on the data that you import into Amazon Personalize. Depending on how you import the data, you can view reports in Amazon CloudWatch or Amazon S3. For more information, see Measuring impact of recommendations.

Contents

creationDateTime

The metric attribution's creation date time.

Type: Timestamp

Required: No

datasetGroupArn

The metric attribution's dataset group Amazon Resource Name (ARN).

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

failureReason

The metric attribution's failure reason.

Type: String

Required: No

lastUpdatedDateTime

The metric attribution's last updated date time.

Type: Timestamp

Required: No

metricAttributionArn

The metric attribution's Amazon Resource Name (ARN).

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

metricsOutputConfig

The metric attribution's output configuration.

Type: MetricAttributionOutput object

Required: No

name

The metric attribution's name.

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 metric attribution's status.

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 Java V2
- AWS SDK for Ruby V3

MetricAttributionOutput

Service: Amazon Personalize

The output configuration details for a metric attribution.

Contents

roleArn

The Amazon Resource Name (ARN) of the IAM service role that has permissions to add data to your output Amazon S3 bucket and add metrics to Amazon CloudWatch. For more information, see Measuring impact of recommendations.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):iam::\d{12}:role/?[a-zA-Z_0-9+=,.@\-_/]+`

Required: Yes

s3DataDestination

The configuration details of an Amazon S3 input or output bucket.

Type: S3DataConfig 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 Java V2
- AWS SDK for Ruby V3

MetricAttributionSummary

Service: Amazon Personalize

Provides a summary of the properties of a metric attribution. For a complete listing, call the DescribeMetricAttribution.

Contents

creationDateTime

The metric attribution's creation date time.

Type: Timestamp

Required: No

failureReason

The metric attribution's failure reason.

Type: String

Required: No

lastUpdatedDateTime

The metric attribution's last updated date time.

Type: Timestamp

Required: No

metricAttributionArn

The metric attribution's Amazon Resource Name (ARN).

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

name

The name of the metric attribution.

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 metric attribution's status.

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 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 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 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-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 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 API.

Contents

creationDateTime

The date and time (in Unix time) that the recipe was created.

Type: Timestamp

Required: No

domain

The domain of the recipe (if the recipe is a Domain dataset group use case).

Type: String

Valid Values: `ECOMMERCE | VIDEO_ON_DEMAND`

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-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 Java V2
- AWS SDK for Ruby V3

Recommender

Service: Amazon Personalize

Describes a recommendation generator for a Domain dataset group. You create a recommender in a Domain dataset group for a specific domain use case (domain recipe), and specify the recommender in a GetRecommendations request.

Contents

creationDateTime

The date and time (in Unix format) that the recommender was created.

Type: Timestamp

Required: No

datasetGroupArn

The Amazon Resource Name (ARN) of the Domain dataset group that contains the recommender.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

failureReason

If a recommender fails, the reason behind the failure.

Type: String

Required: No

lastUpdatedDateTime

The date and time (in Unix format) that the recommender was last updated.

Type: Timestamp

Required: No

latestRecommenderUpdate

Provides a summary of the latest updates to the recommender.

Type: RecommenderUpdateSummary object

Required: No

modelMetrics

Provides evaluation metrics that help you determine the performance of a recommender. For more information, see Evaluating a recommender.

Type: String to double map

Map Entries: Maximum number of 100 items.

Key Length Constraints: Maximum length of 256.

Required: No

name

The name of the recommender.

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 (Domain dataset group use case) that the recommender was created for.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

recommenderArn

The Amazon Resource Name (ARN) of the recommender.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

recommenderConfig

The configuration details of the recommender.

Type: RecommenderConfig object

Required: No

status

The status of the recommender.

A recommender can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- STOP PENDING > STOP IN_PROGRESS > INACTIVE > START PENDING > START IN_PROGRESS > ACTIVE
- 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 Java V2
- AWS SDK for Ruby V3

RecommenderConfig

Service: Amazon Personalize

The configuration details of the recommender.

Contents

enableMetadataWithRecommendations

Whether metadata with recommendations is enabled for the recommender. If enabled, you can specify the columns from your Items dataset in your request for recommendations. Amazon Personalize returns this data for each item in the recommendation response. For information about enabling metadata for a recommender, see Enabling metadata in recommendations for a recommender.

If you enable metadata in recommendations, you will incur additional costs. For more information, see Amazon Personalize pricing.

Type: Boolean

Required: No

itemExplorationConfig

Specifies 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 recommenders generate personalized recommendations for a user (not popular items or similar items).

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

minRecommendationRequestsPerSecond

Specifies the requested minimum provisioned recommendation requests per second that Amazon Personalize will support. A high `minRecommendationRequestsPerSecond` will increase your bill. We recommend starting with 1 for `minRecommendationRequestsPerSecond` (the default). Track your usage using Amazon CloudWatch metrics, and increase the `minRecommendationRequestsPerSecond` as necessary.

Type: Integer

Valid Range: Minimum value of 1.

Required: No

trainingDataConfig

Specifies the training data configuration to use when creating a domain recommender.

Type: TrainingDataConfig 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 Java V2
- AWS SDK for Ruby V3

RecommenderSummary

Service: Amazon Personalize

Provides a summary of the properties of the recommender.

Contents

creationDateTime

The date and time (in Unix format) that the recommender was created.

Type: Timestamp

Required: No

datasetGroupArn

The Amazon Resource Name (ARN) of the Domain dataset group that contains the recommender.

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 recommender was last updated.

Type: Timestamp

Required: No

name

The name of the recommender.

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 (Domain dataset group use case) that the recommender was created for.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

recommenderArn

The Amazon Resource Name (ARN) of the recommender.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

recommenderConfig

The configuration details of the recommender.

Type: RecommenderConfig object

Required: No

status

The status of the recommender. A recommender can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- STOP PENDING > STOP IN_PROGRESS > INACTIVE > START PENDING > START IN_PROGRESS > ACTIVE
- 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 Java V2
- AWS SDK for Ruby V3

RecommenderUpdateSummary

Service: Amazon Personalize

Provides a summary of the properties of a recommender update. For a complete listing, call the DescribeRecommender API.

Contents

creationDateTime

The date and time (in Unix format) that the recommender update was created.

Type: Timestamp

Required: No

failureReason

If a recommender update fails, the reason behind the failure.

Type: String

Required: No

lastUpdatedDateTime

The date and time (in Unix time) that the recommender update was last updated.

Type: Timestamp

Required: No

recommenderConfig

The configuration details of the recommender update.

Type: RecommenderConfig object

Required: No

status

The status of the recommender update.

A recommender can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- STOP PENDING > STOP IN_PROGRESS > INACTIVE > START PENDING > START IN_PROGRESS > ACTIVE
- 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 Java V2
- AWS SDK for Ruby V3

S3DataConfig

Service: Amazon Personalize

The configuration details of an Amazon S3 input or output bucket.

Contents

path

The file path of the Amazon S3 bucket.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `(s3|http|https)://.+`

Required: Yes

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.

Type: String

Length Constraints: Maximum length of 2048.

Pattern: `arn:aws.*:kms:.*:[0-9]{12}:key/.*`

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 Java V2
- AWS SDK for Ruby V3

Solution

Service: Amazon Personalize

> **Important**
>
> After you create a solution, you can’t change its configuration. By default, all new solutions use automatic training. With automatic training, you incur training costs while your solution is active. You can't stop automatic training for a solution. To avoid unnecessary costs, make sure to delete the solution when you are finished. For information about training costs, see Amazon Personalize pricing.

An object that provides information about a solution. A solution includes the custom recipe, customized parameters, and trained models (Solution Versions) that Amazon Personalize uses to generate recommendations.

After you create a solution, you can’t change its configuration. If you need to make changes, you can clone the solution with the Amazon Personalize console or create a new one.

Contents

autoMLResult

When `performAutoML` is true, specifies the best recipe found.

Type: AutoMLResult 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 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

> **Important**
>
> We don't recommend enabling automated machine learning. Instead, match your use case to the available Amazon Personalize recipes. For more information, see [Determining your use case.]

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

performAutoTraining

Specifies whether the solution automatically creates solution versions. The default is `True` and the solution automatically creates new solution versions every 7 days.

For more information about auto training, see [Creating and configuring a solution].

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. This is required when `performAutoML` is false.

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 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 Java V2
- AWS SDK for Ruby V3

SolutionConfig

Service: Amazon Personalize

Describes the configuration properties for the solution.

Contents

algorithmHyperParameters

Lists the algorithm hyperparameters and their values.

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 object containing a list of recipes to search when AutoML is performed.

Type: AutoMLConfig object

Required: No

autoTrainingConfig

Specifies the automatic training configuration to use.

Type: AutoTrainingConfig 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 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 object

Required: No

trainingDataConfig

Specifies the training data configuration to use when creating a custom solution version (trained model).

Type: TrainingDataConfig 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 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 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

recipeArn

The Amazon Resource Name (ARN) of the recipe used by the solution.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

solutionArn

The Amazon Resource Name (ARN) of the solution.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

status

The status of the solution.

A solution can be in one of the following states:

- CREATE PENDING > CREATE IN_PROGRESS > ACTIVE -or- CREATE FAILED
- DELETE PENDING > DELETE IN_PROGRESS

Type: String

Length Constraints: Maximum length of 256.

Required: No

See Also

For more information about using this API in one of the language-specific AWS SDKs, see the following:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK for Ruby V3

SolutionVersion

Service: Amazon Personalize

An object that provides information about a specific version of a Solution in a Custom dataset group.

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

name

The name of the solution version.

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 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]() 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. A `FULL` training considers all of the data in your dataset group. An `UPDATE` processes only the data that has changed since the latest training. Only solution versions created with the User-Personalization recipe can use `UPDATE`.

Type: String

Valid Values: `FULL | UPDATE | AUTOTRAIN`

Required: No

trainingType

Whether the solution version was created automatically or manually.

Type: String

Valid Values: `AUTOMATIC | MANUAL`

Required: No

tunedHPOParams

If hyperparameter optimization was performed, contains the hyperparameter values of the best performing model.

Type: TunedHPOParams 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 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 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

trainingMode

The scope of training to be performed when creating the solution version. A `FULL` training considers all of the data in your dataset group. An `UPDATE` processes only the data that has changed since the latest training. Only solution versions created with the User-Personalization recipe can use `UPDATE`.

Type: String

Valid Values: `FULL | UPDATE | AUTOTRAIN`

Required: No

trainingType

Whether the solution version was created automatically or manually.

Type: String

Valid Values: `AUTOMATIC | MANUAL`

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 Java V2
- AWS SDK for Ruby V3

Tag

Service: Amazon Personalize

The optional metadata that you apply to resources to help you categorize and organize them. Each tag consists of a key and an optional value, both of which you define. For more information see Tagging Amazon Personalize resources.

Contents

tagKey

One part of a key-value pair that makes up a tag. A key is a general label that acts like a category for more specific tag values.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 128.

Pattern: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

Required: Yes

tagValue

The optional part of a key-value pair that makes up a tag. A value acts as a descriptor within a tag category (key).

Type: String

Length Constraints: Minimum length of 0. Maximum length of 256.

Pattern: `^([\p{L}\p{Z}\p{N}_.:/=+\-@]*)$`

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 Java V2

- [AWS SDK for Ruby V3]

ThemeGenerationConfig

Service: Amazon Personalize

The configuration details for generating themes with a batch inference job.

Contents

fieldsForThemeGeneration

Fields used to generate descriptive themes for a batch inference job.

Type: FieldsForThemeGeneration 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 Java V2
- AWS SDK for Ruby V3

TrainingDataConfig

Service: Amazon Personalize

The training data configuration to use when creating a domain recommender or custom solution version (trained model).

Contents

excludedDatasetColumns

Specifies the columns to exclude from training. Each key is a dataset type, and each value is a list of columns. Exclude columns to control what data Amazon Personalize uses to generate recommendations.

For example, you might have a column that you want to use only to filter recommendations. You can exclude this column from training and Amazon Personalize considers it only when filtering.

Type: String to array of strings map

Map Entries: Maximum number of 3 items.

Key Length Constraints: Maximum length of 256.

Array Members: Maximum number of 50 items.

Length Constraints: Maximum length of 150.

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 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 Java V2
- AWS SDK for Ruby V3

Amazon Personalize Events

The following data types are supported by Amazon Personalize Events:

- Action
- ActionInteraction
- Event
- Item

- MetricAttribution
- User

Action

Service: Amazon Personalize Events

Represents action metadata added to an Action dataset using the `PutActions` API. For more information see Importing actions individually.

Contents

actionId

The ID associated with the action.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: Yes

properties

A string map of action-specific metadata. Each element in the map consists of a key-value pair. For example, `{"value": "100"}`.

The keys use camel case names that match the fields in the schema for the Actions dataset. In the previous example, the `value` matches the 'VALUE' field defined in the Actions schema. For categorical string data, to include multiple categories for a single action, separate each category with a pipe separator (|). For example, `\"Deluxe|Premium\"`.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 32000.

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 Java V2
- AWS SDK for Ruby V3

ActionInteraction

Service: Amazon Personalize Events

Represents an action interaction event sent using the `PutActionInteractions` API.

Contents

actionId

The ID of the action the user interacted with. This corresponds to the `ACTION_ID` field of the Action interaction schema.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: Yes

eventType

The type of action interaction event. You can specify `Viewed`, `Taken`, and `Not Taken` event types. For more information about action interaction event type data, see Event type data.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: Yes

sessionId

The ID associated with the user's visit. Your application generates a unique `sessionId` when a user first visits your website or uses your application.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: Yes

timestamp

The timestamp for when the action interaction event occurred. Timestamps must be in Unix epoch time format, in seconds.

Type: Timestamp

Required: Yes

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

impression

A list of action IDs that represents the sequence of actions you have shown the user. For example, `["actionId1", "actionId2", "actionId3"]`. Amazon Personalize doesn't use impressions data from action interaction events. Instead, record multiple events for each action and use the `Viewed` event type.

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

properties

A string map of event-specific data that you might choose to record. For example, if a user takes an action, other than the action ID, you might also send the number of actions taken by the user.

Each item in the map consists of a key-value pair. For example,

`{"numberOfActions": "12"}`

The keys use camel case names that match the fields in the Action interactions schema. In the above example, the `numberOfActions` would match the 'NUMBER_OF_ACTIONS' field defined in the Action interactions schema.

The following can't be included as a keyword for properties (case insensitive).

- userId
- sessionId
- eventType
- timestamp
- recommendationId
- impression

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Required: No

recommendationId

The ID of the list of recommendations that contains the action the user interacted with.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 40.

Required: No

userId

The ID of the user who interacted with the action. This corresponds to the `USER_ID` field of the Action interaction schema.

Type: String

Length Constraints: Minimum length of 1. 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 Java V2
- AWS SDK for Ruby V3

Event

Service: Amazon Personalize Events

Represents item interaction event information sent using the `PutEvents` API.

Contents

eventType

The type of event, such as click or download. This property corresponds to the `EVENT_TYPE` field of your Item interactions dataset's 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

sentAt

The timestamp (in Unix time) on the client side when the event occurred.

Type: Timestamp

Required: Yes

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

eventValue

The event value that corresponds to the `EVENT_VALUE` field of the Item 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"]`. Provide a list of items to manually record impressions data for an event. For more information on recording impressions data, see [Recording impressions data](#).

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 Item interactions dataset's schema.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: No

metricAttribution

Contains information about the metric attribution associated with an event. For more information about metric attributions, see [Measuring impact of recommendations](#).

Type: [MetricAttribution](#) object

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 Item interactions dataset's schema. In the above example, the `numberOfRatings` would match the 'NUMBER_OF_RATINGS' field defined in the Item interactions dataset's schema.

The following can't be included as a keyword for properties (case insensitive).

- userId
- sessionId
- eventType
- timestamp
- recommendationId
- impression

Type: String

Length Constraints: Minimum length of 1. Maximum length of 1024.

Required: No

recommendationId

The ID of the list of recommendations that contains the item the user interacted with. Provide a `recommendationId` to have Amazon Personalize implicitly record the recommendations you show your user as impressions data. Or provide a `recommendationId` if you use a metric attribution to measure the impact of recommendations.

For more information on recording impressions data, see Recording impressions data. For more information on creating a metric attribution see Measuring impact of recommendations.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 40.

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 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 individually.

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 32000.

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 Java V2
- AWS SDK for Ruby V3

MetricAttribution

Service: Amazon Personalize Events

Contains information about a metric attribution associated with an event. For more information about metric attributions, see Measuring impact of recommendations.

Contents

eventAttributionSource

The source of the event, such as a third party.

Type: String

Length Constraints: Maximum length of 1024.

Pattern: `^[\x20-\x7E]*[\x21-\x7E]+[\x20-\x7E]*$`

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 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 individually.

Contents

userId

The ID associated with the user.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 256.

Required: Yes

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 24000.

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 Java V2

- AWS SDK for Ruby V3

Amazon Personalize Runtime

The following data types are supported by Amazon Personalize Runtime:

- PredictedAction
- PredictedItem
- Promotion

PredictedAction

Service: Amazon Personalize Runtime

An object that identifies an action.

The GetActionRecommendations API returns a list of `PredictedActions`.

Contents

actionId

The ID of the recommended action.

Type: String

Length Constraints: Maximum length of 256.

Required: No

score

The score of the recommended action. For information about action scores, see How action recommendation scoring works.

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 Java V2
- AWS SDK for Ruby V3

PredictedItem

Service: Amazon Personalize Runtime

An object that identifies an item.

The GetRecommendations and GetPersonalizedRanking APIs return a list of `PredictedItems`.

Contents

itemId

The recommended item ID.

Type: String

Length Constraints: Maximum length of 256.

Required: No

metadata

Metadata about the item from your Items dataset.

Type: String to string map

Key Length Constraints: Maximum length of 150.

Value Length Constraints: Maximum length of 20000.

Required: No

promotionName

The name of the promotion that included the predicted item.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

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.

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 Java V2
- AWS SDK for Ruby V3

Promotion

Service: Amazon Personalize Runtime

Contains information on a promotion. A promotion defines additional business rules that apply to a configurable subset of recommended items.

Contents

filterArn

The Amazon Resource Name (ARN) of the filter used by the promotion. This filter defines the criteria for promoted items. For more information, see Promotion filters.

Type: String

Length Constraints: Maximum length of 256.

Pattern: `arn:([a-z\d-]+):personalize:.*:.*:.+`

Required: No

filterValues

The values to use when promoting items. For each placeholder parameter in your promotion's 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 on creating filters, see Filtering recommendations and user segments.

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

name

The name of the promotion.

Type: String

Length Constraints: Minimum length of 1. Maximum length of 63.

Pattern: `^[a-zA-Z0-9][a-zA-Z0-9\-_]*`

Required: No

percentPromotedItems

The percentage of recommended items to apply the promotion to.

Type: Integer

Valid Range: Minimum value of 1. Maximum value of 100.

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 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

NotAuthorized

You do not have permission to perform this action.

HTTP Status Code: 400

OptInRequired

The AWS access key ID needs a subscription for the service.

HTTP Status Code: 403

RequestExpired

The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.

HTTP Status Code: 400

ServiceUnavailable

The request has failed due to a temporary failure of the server.

HTTP Status Code: 503

ThrottlingException

The request was denied due to request throttling.

HTTP Status Code: 400

ValidationError

The input fails to satisfy the constraints specified by an AWS service.

HTTP Status Code: 400

Common Parameters

The following list contains the parameters that all actions use for signing Signature Version 4 requests with a query string. Any action-specific parameters are listed in the topic for that action. For more information about Signature Version 4, see Signing AWS API requests in the *IAM User Guide*.

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 Create a signed AWS API request in the *IAM User Guide*.

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 Elements of an AWS API request signature in the *IAM User Guide*.

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 STS, see AWS services that work with IAM in the *IAM User Guide*.

Condition: If you're using temporary security credentials from AWS STS, 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 Create a signed AWS API request in the *IAM User Guide*.

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.

Change	Description	Date
New feature	All new Amazon Personalize custom solutions now use automatic training. For more information about configuring automatic training, see Configuring automatic training.	April 19, 2024
New feature	Amazon Personalize now supports the ability to recommend the next best action for a user based on their behavior. For more information, see Next-Best-Action recipe.	November 26, 2023
New feature	Amazon Personalize now supports the ability to include descriptive themes in batch recommendations with the help of generative AI. For information about generating batch recommendations with themes, see Batch recommendations with themes. For information about using Amazon Personalize with generative AI, see	November 26, 2023

	Amazon Personalize and generative AI.	
New feature	Amazon Personalize now supports the ability to include metadata from an Items dataset in recommend ations. For information about enabling metadata for a campaign, see Enabling metadata in recommend ations for a campaign. For information about enabling metadata for a recommend er, see Enabling metadata in recommendations for a recommender.	November 26, 2023
New feature	Amazon Personalize now supports the ability to personalize search results from OpenSearch Service. For more information, see Personalizing search results from OpenSearch.	October 16, 2023
New feature	Amazon Personalize now supports the ability to import up to 100 metadata columns for Items datasets (up from 50) and 25 metadata columns for Users datasets (up from 5). For more information about Amazon Personalize limits, see Amazon Personali ze endpoints and quotas.	September 5, 2023

New feature	Amazon Personalize now supports the ability to personalize search results from OpenSearch. For more information, see Personalizing search results from OpenSearch (self-managed).	July 25, 2023
New feature	Amazon Personalize now supports the ability to replace a dataset's schema with a new or existing one. For more information, see Replacing a dataset's schema.	July 13, 2023
New feature	If you use the User-Personalization or Personalized-Ranking recipes, Amazon Personalize batch inference jobs can now use data that you import incrementally without retraining. For information, see Getting batch recommendations.	June 30, 2023
New feature	Amazon Personalize now supports the ability to filter items based on the item you specify in your request for related items recommendations. For information about filters, see Filtering recommendations and user segments.	June 21, 2023

New feature	Amazon Personalize now supports the ability to filter items based on the item you specify in your request for related items recommend ations. For information about filters, see Filtering recommendations and user segments.	June 21, 2023
New feature	Amazon Personalize now supports private connectio ns between a virtual private cloud (VPC) and Amazon Personalize with an interface Amazon VPC endpoint. For more information, see Amazon Personalize and interface VPC endpoints (AWS PrivateLink).	June 12, 2023
New feature	Amazon Personalize now supports configuring the columns used when training when you create a recommender or custom solution. For information about configuring columns when creating a recommend er, see Creating recommend ers. For information about configuring columns when creating a solution, see Configuring columns used when training.	May 30, 2023

New documentation feature	The Amazon Personalize developer guide now includes information about performin g A/B testing with Amazon Personalize recommend ations. For more information, see Measuring recommend ation impact with A/B testing.	May 5, 2023
New feature	Amazon Personalize now supports configuring how popularity influences recommendations generated by the Similar-Items recipe. For more information, see Similar Items recipe.	April 21, 2023
New feature	Amazon Personalize now supports using Amazon SageMaker Data Wrangler to import data from 40+ sources into Amazon Personalize datasets. For more informati on, see Importing data using Amazon SageMaker Data Wrangler.	April 14, 2023
New documentation feature	The Amazon Personalize developer guide now includes a new readiness checklist. This checklist helps you prepare to use Amazon Personalize with your own data. For more information, see Readiness checklist.	February 9, 2023

New feature	Amazon Personalize now supports generating insights and statistics for data that you import into datasets. For more information, see Analyzing data in datasets.	January 25, 2023
New feature	Amazon Personalize now supports a new Trending-Now recipe for Custom dataset groups. For more information, see Trending-Now recipe.	January 6, 2023
New feature	Amazon Personalize now supports using tags in IAM policies. For more informati on, see Using tags in IAM policies.	December 28, 2022
New feature	Amazon Personalize now supports making adjustmen ts to the *Maximum number of interactions that are considere d by a model during training* quota. Additionally, the *Maximum number of users that are considered by a model during training limit* quota no longer applies. For more information, see Amazon Personalize endpoints and quotas.	December 15, 2022

New feature	Amazon Personalize now supports creating a metric attribution to measure the business impact of recommendations. For more information, see Measuring impact of recommendations.	November 17, 2022
New feature	Amazon Personalize quotas for total number of active solutions, active campaigns , recommenders, and filters have now increased. Each of these quotas now apply per dataset group rather than per account. For information about quotas, see Amazon Personalize endpoints and quotas.	September 7, 2022
New feature	Amazon Personalize filters now consider up to 100 interactions per user per event type. For information about filters, see Filtering recommendations and user segments.	August 29, 2022
New feature	Amazon Personalize now supports a new Trending now use case for the VIDEO_ON_ DEMAND domain. For more information, see VIDEO_ON_ DEMAND use cases.	August 17, 2022

New feature	Amazon Personalize now supports promoting items in recommendations with a separate promotion filter. For information about promoting items see Promoting items in recommendations.	August 12, 2022
New feature	Amazon Personalize now supports using comparison operators in filter expressions with placeholder parameter s. For information about filter expressions, see Filter expressions.	August 12, 2022
New feature	Amazon Personalize now supports incremental bulk updates to datasets. You can now use a dataset import job to update a dataset without replacing the existing data. For more information, see Updating existing bulk data.	August 2, 2022
New feature	Amazon Personalize can now use unstructured text metadata in different languages. For more informati on, see Unstructured text metadata.	June 6, 2022

New feature	Amazon Personalize recommenders now generate offline metrics. You can use these metrics to evaluate the performance of your recommender. For more information see Evaluating a recommender.	May 24, 2022
New feature	Amazon Personalize now supports the ability stop a recommender and restart it later. This way, you can pause recommender billing and pay for it only when you use it. For more information, see Stopping and starting a recommender.	April 20, 2022
New feature	Amazon Personalize now supports using tags to categorize and manage Amazon Personalize resources . For more information, see Tagging Amazon Personalize resources.	April 7, 2022
New feature	Amazon Personalize now supports specifying resources with AWS CloudFormation. For more information, see Specifying resources with AWS CloudFormation.	March 11, 2022

New documentation feature	The Amazon Personalize developer guide now includes a new Troubleshooting topic that provides answers to common questions and troubleshooting advice for error messages that you might encounter with Amazon Personalize. For more information, see Troublesh ooting.	February 15, 2022
New feature	Amazon Personalize now supports creating a Domain dataset group with use case optimized resources for video on demand or e-commerce domains. For more informati on, see Domain dataset groups.	November 29, 2021
New feature	Amazon Personalize now supports creating user segments with new USER_SEGMENTATION recipes. USER_SEGM ENTATION recipes generate segments of users based on item input data. For more information, see USER_SEGM ENTATION recipes.	November 29, 2021

New feature	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	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	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	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)	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	Amazon Personalize can now export the records in an Amazon Personalize dataset to an Amazon S3 bucket for analysis and tracking. For more information, see Exporting a dataset.	April 26, 2021
New feature	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	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	Amazon Personalize now supports the ability to incrementally import users and items. For more information, see Importing records incrementally.	October 2, 2020
New feature	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	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	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	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	Amazon Personalize adds support for the Asia Pacific (Seoul) Region. For a complete list of the AWS Regions supported by Amazon Personalize, see the AWS Region table or AWS Regions and endpoints in the *Amazon Web Services General Reference*.	January 21, 2020
New feature	Amazon Personalize can now get recommendations based on contextual metadata. For more information, see Getting recommendations.	December 19, 2019

New regions	Amazon Personalize adds support for the Asia Pacific (Mumbai), Asia Pacific (Sydney), and Canada (Central) Regions. For a complete list of the AWS Regions supported by Amazon Personalize, see the AWS Region table or AWS Regions and endpoints in the *Amazon Web Services General Reference*.	December 18, 2019
New feature	Amazon Personalize now supports batch recommend ation workflows. For more information, see Get batch recommendations.	November 14, 2019
Amazon Personalize general availability	Amazon Personalize is now available for general use.	June 10, 2019
Amazon Personalize preview release	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 Glossary Reference*.