aws

User Guide

AWS Entity Resolution

AWS Entity Resolution: User Guide

Table of Contents

What is AWS Entity Resolution?

AWS Entity Resolution is a service that helps you match, link, and enhance related records stored across multiple applications, channels, and data stores. You can get started using entity resolution workflows that are flexible, scalable, and can connect to your existing applications and data service providers.

AWS Entity Resolution offers advanced matching techniques, such as rule-based matching, machine learning-based matching (ML matching), and data service provider-led matching. These techniques can help you more accurately link and enhance related records of customer information, product codes, or business data codes.

You can use AWS Entity Resolution to create a unified view of customer interactions by linking recent events (such as ad clicks, cart abandonment, and purchases) with pseudonymized signals from your data service providers into a unique entity ID. You can also better track products that use different codes (for example, SKU, UPC) across your stores. You can use AWS Entity Resolution to control matching accuracy and better protect data security while minimizing data movement.

Topics

Are you a first-time AWS Entity Resolution user?

If you're a first-time user of AWS Entity Resolution, we recommend that you begin by reading the following sections:

- Features of AWS Entity Resolution
- Accessing AWS Entity Resolution
- Set up AWS Entity Resolution

Features of AWS Entity Resolution

AWS Entity Resolution includes the following features:

- **Flexible and customizable data preparation**

  AWS Entity Resolution reads your data from AWS Glue to use as inputs for match processing. You can specify a maximum of 20 data inputs. AWS Entity Resolution processes each row of the data input table as a record, with a unique entity serving as a primary key. AWS Entity Resolution can operate on encrypted datasets. First define the schema mapping for AWS Entity Resolution to understand what input fields you want to use in your matching workflow. You can bring your own data schema, or blueprint, from an existing AWS Glue data input. Or, you can build your custom schema using an interactive user interface or JSON editor. By default, AWS Entity Resolution also normalizes data inputs before matching to improve match processing, such as removing special characters and extra spaces, and formatting text to lowercase. If your data input is already normalized, then you can turn off normalization. We also provide a GitHub library, which you can use to further customize the data normalization process to suit your needs.

- **Configurable entity matching workflows**

  An entity matching workflow is a sequence of steps that you set up to tell AWS Entity Resolution how to match your data input and where to write the consolidated data output. You can set up one or more matching workflows to compare different data inputs and use different matching techniques, such as rule-based matching, machine learning matching, or data service provider-led matching without entity resolution or ML experience. You can also view the job status of existing matching workflows and metrics, such as resource number, number of records processed, and number of matches found.

  - **Ready-to-use rule-based matching**

    This matching technique includes a set of ready-to-use rules in the AWS Management Console or AWS Command Line Interface (AWS CLI). You can use these rules to find related records based on your input fields. You can also customize the rules by adding or removing input fields for each rule, deleting rules, rearranging rule priority, and creating new rules. You can also reset the rules to return them to their original configurations. The data output in your Amazon Simple Storage Service (Amazon S3) bucket has match groups that AWS Entity Resolution generates using the rule-based matching technique. Each match group has the rule number used to generate that match associated with it to help you understand the match. For

example, the rule number can demonstrate the precision of each match group such that rule one is more precise than rule two.

- **Pre-configured machine learning-based matching (ML matching)**

  This matching technique includes a pre-configured ML model to find matches across all of your data inputs, especially consumer-based records. The model uses all input fields associated with name, email address, phone number, address, and date of birth data types. The model generates match groups of related records with a confidence score in each group explaining the quality of the match relative to other match groups. The model considers missing input fields and analyzes the entire record together to represent an entity. The data output in your Amazon S3 bucket has match groups that AWS Entity Resolution generates using the ML matching. This is where each match group has an associated confidence score of 0.0–1.0, which indicates the precision of the match.

- **Matching records with data service providers**

  With AWS Entity Resolution you can match, link, and enhance your records with leading data service vendors and licensed datasets to expand your ability to understand, reach, and service your customers. For example, you can append attributes to your data to enhance your records, or you can improve the interoperability of systems and platforms you work with to meet your business goals. You can use this matching workflow with a few clicks, removing the need to build and maintain complex proprietary integrations. You must have a license agreement with these data service providers to take advantage of this matching technique.

- **Manual bulk processing and automatic incremental processing**

  You can use data processing to help convert your data input or inputs into a consolidated data output table with similar records that have a common match ID generated using entity matching workflow configurations. Using the API and AWS Management Console or the AWS CLI, you can run manual bulk processing on demand, based on your existing extract, transform, and load (ETL) data pipeline, which re-processes all data for any new matches and updates to existing matches. Also, for rule-based matching scenarios, you can initiate automatic incremental processing so that as soon as new data is available in your Amazon S3 bucket, the service reads those new records and compares them against existing records. This keeps your matches up to date with any changes in Amazon S3 data.

- **Near real-time lookup**

  Looking up any entity fields through the AWS Entity Resolution GetMatchId API operation helps you synchronously retrieve an existing match ID. You can call AWS Entity Resolution

with personally identifiable information (PII) attributes acquired through different sources and channels. AWS Entity Resolution hashes those attributes for data protection and retrieves the corresponding match ID to link and match the customer. For example, you can get a web sign-up with an associated name, email, and mailing address. Use the AWS Entity Resolution GetMatchId API operation to find out if this customer or entity already exists in your matched results stored in your S3 bucket, along with the corresponding entity match ID associated with it. After you get the entity match ID, you can find the transactional information associated with it in your source applications, such as your customer relationship management (CRM) or customer data platform (CDP) systems.

- **Data protection and Regionalization by design**

  AWS Entity Resolution offers a default encryption capability that can help you protect your data, and equips you with an encryption key for every data input into the service. For example, AWS Entity Resolution gives you the flexibility to bring server-side encrypted and hashed data to run rule-based matching workflows. AWS Entity Resolution supports Regionalization, which means that your matching workflows run to process your data in the same AWS Region from where you're using the service. You can also encrypt and hash the data output in Amazon S3 before using your resolved data in other applications.

- **Multi-party transcoding**

  AWS Entity Resolution helps you define your data sources and matching configurations between multiple parties who want to use a data collaboration, such as in AWS Clean Rooms.

Related services

The following AWS services are related to AWS Entity Resolution:

- **Amazon S3**

  Store data that you bring into AWS Entity Resolution in Amazon S3.

  For more information, see What Is Amazon S3? in the *Amazon Simple Storage Service User Guide*.

- **AWS Glue**

  Create AWS Glue tables from your data in Amazon S3 for use in AWS Entity Resolution.

  For more information, see What is AWS Glue? in the *AWS Glue Developer Guide*.

- **AWS CloudTrail**

Use AWS Entity Resolution with CloudTrail logs to enhance your analysis of AWS service activity.

For more information, see Logging AWS Entity Resolution API calls using AWS CloudTrail.

- **AWS CloudFormation**

  Create the following resources in AWS CloudFormation: AWS::EntityResolution::MatchingWorkflow, AWS::EntityResolution::SchemaMapping, AWS::EntityResolution:IdMappingWorkflow, AWS::EntityResolution::IdNamespace and AWS::EntityResolution::PolicyStatement

  For more information, see Create AWS Entity Resolution resources with AWS CloudFormation.

Accessing AWS Entity Resolution

You can access AWS Entity Resolution through the following options:

- Directly through the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
- Programmatically through the AWS Entity Resolution API. For more information, see the *AWS Entity Resolution API Reference*.
  - If you plan to call the AWS Entity Resolution API in AWS Lambda Runtime, create your own deployment package and include the desired version of the AWS SDK library. For more information, see the following examples in the *AWS Lambda Developer Guide*:
    - Deploy Java Lambda functions with .zip or JAR file archives
    - Working with .zip file archives for Python Lambda functions

Pricing for AWS Entity Resolution

For pricing information, see AWS Entity Resolution Pricing.

Set up AWS Entity Resolution

Before you use AWS Entity Resolution for the first time, sign up for AWS and create an administrator user to create roles.

Signing up for AWS

If you already have an AWS account, skip this step.

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 or text message 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 a user, and use only the root user to perform tasks that require root user access.

Creating an administrator user

To create an administrator user, choose one of the following options.

Choose one way to manage your administrator	To	By	You can also
In IAM Identity Center (Recommended)	Use short-term credentials to access AWS. This aligns with the security best practices. For information about best practices, see Security best practices in IAM in the *IAM User Guide*.	Following the instructions in Getting started in the *AWS IAM Identity Center User Guide*.	Configure programmatic access by Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*.
In IAM (Not recommended)	Use long-term credentials to access AWS.	Following the instructions in Create an IAM user for emergency access in the *IAM User Guide*.	Configure programmatic access by Manage access keys for IAM users in the *IAM User Guide*.

Creating an IAM role for a console user

Complete the following procedure if you are using the AWS Entity Resolution console.

To create an IAM role

1. Sign in to the IAM console (https://console.aws.amazon.com/iam/) with your administrator account.
2. Under **Access management**, choose **Roles**.

You can use **Roles** to create short-term credentials, which is recommended for increased security. You can also choose **Users** to create long-term credentials.

3. Choose **Create role**.
4. In the **Create role** wizard, for **Trusted entity type**, choose **AWS account**.
5. Keep the option **This account** selected, and then choose **Next**.
6. For **Add permissions**, choose **Create Policy**.

   A new tab opens.

   a. Select the **JSON** tab, and then add policies depending on the abilities granted to the console user. AWS Entity Resolution offers the following managed policies based on common use cases:

      - AWS managed policy: AWSEntityResolutionConsoleFullAccess
      - AWS managed policy: AWSEntityResolutionConsoleReadOnlyAccess

   b. Choose **Next: Tags**, add tags (optional), and then choose **Next: Review**.

   c. For **Review policy**, enter a **Name** and **Description**, and review the **Summary**.

   d. Choose **Create policy**.

      You have created a policy for a collaboration member.

   e. Go back to your original tab and under **Add permissions**, enter the name of the policy that you just created. (You might need to reload the page.)

   f. Select the check box next to the name of the policy that you created, and then choose **Next**.

7. For **Name, review, and create**, enter the **Role name** and **Description**.

   a. Review **Select trusted entities**, enter the AWS account for the person or persons who will assume the role (if necessary).

   b. Review the permissions in **Add permissions**, and edit if necessary.

   c. Review the **Tags**, and add tags if necessary.

   d. Choose **Create role**.

Creating a workflow job role for AWS Entity Resolution

AWS Entity Resolution uses a *workflow job role* to run a workflow. You can create this role using the console if you have the necessary IAM permissions. If you don't have `CreateRole` permissions, ask your administrator to create the role.

To create a workflow job role for AWS Entity Resolution

1. Sign in to the IAM console at https://console.aws.amazon.com/iam/ with your administrator account.
2. Under **Access management**, choose **Roles**.

   You can use **Roles** to create short-term credentials, which is recommended for increased security. You can also choose **Users** to create long-term credentials.
3. Choose **Create role**.
4. In the **Create role** wizard, for **Trusted entity type**, choose **Custom trust policy**.
5. Copy and paste the following custom trust policy into the JSON editor.

   JSON

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "Service": [
                       "entityresolution.amazonaws.com"
                   ]
               },
               "Action": "sts:AssumeRole"
           }
       ]
   }
   ```
6. Choose **Next**.
7. For **Add permissions**, choose **Create Policy**.

   A new tab appears.

a. Copy and paste the following policy into the JSON editor.

> **Note**
>
> The following example policy supports the permissions needed to read corresponding data resources like Amazon S3 and AWS Glue. However, you might need to modify this policy depending on how you've set up your data sources. Your AWS Glue resources and underlying Amazon S3 resources must be in the same AWS Region as AWS Entity Resolution.
> You don't need to grant AWS KMS permissions if your data sources aren't encrypted or decrypted.

Replace each *{{user input placeholder}}* with your own information.

aws-region	AWS Region of your resources. Your AWS Glue resources, underlying Amazon S3 resources and AWS KMS resources must be in the same AWS Region as `AWS Entity Resolution .`
accountId	Your AWS account ID.
input-buckets	Amazon S3 buckets which contains the underlying data objects of AWS Glue where `AWS Entity Resolution` will read from.
output-buckets	Amazon S3 buckets where `AWS Entity Resolution` will generate the output data.
input-databases	AWS Glue databases where `AWS Entity Resolution` will read from.

b. (Optional) If the input Amazon S3 bucket is encrypted using the customer's KMS key, add the following:

```
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:{{aws-region}}:{{accountId}}:key/{{inputKeys}}"
            ]
        }
```

Replace each *{{user input placeholder}}* with your own information.

aws-region	AWS Region of your resources. Your AWS Glue resources, underlying Amazon S3 resources and AWS KMS resources must be in the same AWS Region as AWS Entity Resolution .
accountId	Your AWS account ID.
inputKeys	Managed keys in AWS Key Managemen t Service. If your input sources are encrypted, AWS Entity Resolution must decrypt your data using your key.

c. (Optional) If the data being written into the output Amazon S3 bucket needs to be encrypted, add the following:

```
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Encrypt"
            ],
            "Resource": [
                "arn:aws:kms:{{aws-region}}:{{accountId}}:key/{{outputKeys}}"
            ]
        }
```

Replace each *{{user input placeholder}}* with your own information.

aws-region	AWS Region of your resources. Your AWS Glue resources, underlying Amazon S3 resources and AWS KMS resources must be in the same AWS Region as AWS Entity Resolution .
accountId	Your AWS account ID.
outputKeys	Managed keys in AWS Key Managemen t Service. If you need your output sources to be encrypted, AWS Entity Resolution must encrypt the output data using your key.

d. (Optional) If you have a subscription with a provider service through AWS Data Exchange, and want to use an existing role for a provider service-based workflow, add the following:

```
        {
            "Effect": "Allow",
            "Sid": "DataExchangePermissions",
            "Action": "dataexchange:SendApiAsset",
            "Resource": [
                "arn:aws:dataexchange:{{aws-region}}::data-sets/{{datasetId}}/
revisions/{{revisionId}}/assets/{{assetId}}"
            ]
        }
```

Replace each *{{user input placeholder}}* with your own information.

aws-region	The AWS Region where the provider resource is granted. You can find this value in the asset ARN on the AWS Data Exchange console. For example: `arn:aws:dataexchange:us-east-2::data-sets/111122223333/revisions/339ffc64444examplef3bc15cf0b2346b/assets/546468b8dexamplea37bfc73b8f79fefa`
datasetId	The ID of the dataset, found on the AWS Data Exchange console.
revisionId	The revision of the dataset, found on the AWS Data Exchange console.
assetId	The ID of the asset, found on the AWS Data Exchange console.

8. Go back to your original tab and under **Add permissions**, enter the name of the policy that you just created. (You might need to reload the page.)
9. Select the check box next to the name of the policy that you created, and then choose **Next**.
10. For **Name, review, and create**, enter the **Role name** and **Description**.

> **Note**
>
> The **Role name** must match the pattern in the `passRole` permissions granted to the member who can pass the `workflow job role` to create a matching workflow. For example, if you're using the `AWSEntityResolutionConsoleFullAccess` managed policy, remember to include `entityresolution` into your role name.

    a. Review **Select trusted entities**, and edit if necessary.
    b. Review the permissions in **Add permissions**, and edit if necessary.
    c. Review the **Tags**, and add tags if necessary.

d. Choose **Create role**.

The workflow job role for AWS Entity Resolution has been created.

Prepare input data tables

In AWS Entity Resolution, each of your *input data tables* contain source records. These records contain consumer identifiers such as first name, last name, email address, or phone number. These source records can be matched with other source records that you provide within the same or other input data tables. Each record must have a unique Record ID (Unique ID) and you must define it as a primary key while creating a schema mapping within AWS Entity Resolution.

Every input data table is available as an AWS Glue table backed by Amazon S3. You can use your first-party data already within Amazon S3, or import data tables from other third-party SaaS providers into Amazon S3. After you upload the data to Amazon S3, you can use an AWS Glue crawler to create a data table in the AWS Glue Data Catalog. You can then use the data table as an input to AWS Entity Resolution.

The following sections describe how to prepare first-party data and third-party data.

Topics

Preparing first-party input data

The following steps describe how to prepare first-party data to use in a rule-based matching workflow, machine learning-based matching workflow, or an ID mapping workflow.

Step 1: Prepare first-party data tables

Each matching workflow type has a different set of recommendations and guidelines to help ensure a success.

To prepare first-party data tables, consult the following table:

First-party data tables guidelines

Workflow type	Required
Rule-based matching workflow with Advanced rule type	• A Unique ID is required. • The Unique ID doesn't exceed 38 characters.

Workflow type	Required

	• (Optional) A DELETE column that specifies which records to remove from AWS Entity Resolution after the workflow has finished processing. The default value is *`false`* if the column exists without any values. Records with the DELETE column set to *`true`* will be delete. Records with the DELETE column set to *`false`* or empty will processed by AWS Entity Resolution.
	
	The schema must have a DELETE column with type `String` and no `matchKey` and `groupName` .
	
	Note
	Look up match ID (`GetMatchID`) isn't supported because the **Advanced** rule type for the **Manual** processing cadence doesn't store any ingested data.
	
	In the following example, S1 will be ingested and S2 will be deleted.
	
	Example
	
	`sourceID, name, lastName, DELETE`
	`S1, name, lastname, false`
	`S2, name2, lastname2, true`

rule-based matching workflow with Simple rule type	• A Unique ID is required. • The Unique ID doesn't exceed 38 characters.

Workflow type	Required

machine learning-based matching workflow	• A Unique ID is required.
	• The dataset contains one of the following types:
	  • **`Full Name`**
	  • **`Full Address`**
	  • **`Full phone`**
	  • **`Email address`**
	  • **`Date`** – with a **Match key name** of **Date of birth**

ID mapping workflow	• A Unique ID is required. • The Unique ID doesn't exceed 257 characters.

Step 2: Save your input data table in a supported data format

If you already saved your first-party input data in a supported data format, you can skip this step.

To use AWS Entity Resolution, the input data must be in a format that AWS Entity Resolution supports.

AWS Entity Resolution supports the following data formats:

- comma-separated value (CSV)
- Parquet

Step 3: Upload your input data table to Amazon S3

If you already have your first-party data table in Amazon S3, you can skip this step.

> **Note**
>
> The input data must be stored in Amazon Simple Storage Service (Amazon S3) in the same AWS account and AWS Region in which you want to run the matching workflow.

To upload your input data table to Amazon S3

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Buckets**, and then choose a bucket to store your data table.
3. Choose **Upload**, and then follow the prompts.
4. Choose the **Objects** tab to view the prefix where your data is stored. Make a note of the name of the folder.

   You can select the folder to view the data table.

Step 4: Create an AWS Glue table

> **Note**
>
> If you need partitioned AWS Glue tables, skip to Step 4: Create a partitioned AWS Glue table.

The input data in Amazon S3 must be cataloged in AWS Glue and represented as an AWS Glue table. For more information about how to create an AWS Glue table with Amazon S3 as the input, see Working with crawlers on the AWS Glue console in the *AWS Glue Developer Guide.*

In this step, you set up a crawler in AWS Glue that crawls all the files in your S3 bucket and create an AWS Glue table.

> **Note**
>
> AWS Entity Resolution doesn't currently support Amazon S3 locations registered with AWS Lake Formation.

To create an AWS Glue table

1. Sign in to the AWS Management Console and open the AWS Glue console at https://console.aws.amazon.com/glue/.
2. From the navigation bar, select **Crawlers**.

3. Select your S3 bucket from the list, and then choose **Create crawler**.
4. On the **Set crawler properties** page, enter a crawler**Name**optional **Description**, and then choose **Next**.
5. Continue through the **Add crawler page**, specifying the details.
6. On the **Choose an IAM role** page, choose **Choose an existing IAM role** and then choose **Next**.

   You can also choose **Create an IAM role** or have your administrator create the IAM role if needed.
7. For **Create a schedule for this crawler**, keep the **Frequency** default (**Run on demand**) and then choose **Next**.
8. For **Configure the crawler's output**, enter the AWS Glue database and then choose **Next**.
9. Review all the details, and then choose **Finish**.
10. On the **Crawlers** page, select the check box next to your S3 bucket and then choose **Run crawler**.
11. After the crawler is finished running, on the AWS Glue navigation bar, choose **Databases**, and then choose your database name.
12. On the **Database** page, choose **Tables in {your database name}**.

    a. View the tables in the AWS Glue database.

    b. To view a table's schema, select a specific table.

    c. Make a note of the AWS Glue database name and AWS Glue table name.

You are now ready to create a schema mapping. For more information, see Creating a schema mapping.

Step 4: Create a partitioned AWS Glue table

> **Note**
>
> The AWS Glue partitioning feature in AWS Entity Resolution is only supported in ID mapping workflows. This AWS Glue partitioning feature enables you to choose specific partitions for processing with AWS Entity Resolution.
> If you don't need partitioned AWS Glue tables, you can skip this step.

A partitioned AWS Glue table automatically reflects new partitions in the AWS Glue table when you add new folders to the data structure (such as a new day folder under a month).

When you create a partitioned AWS Glue table in AWS Entity Resolution, you can specify which partitions you want to process in an ID mapping workflow. Then, every time you run the ID mapping workflow, only the data in those partitions are processed, rather than processing all of the data in the entire AWS Glue table. This feature allows for more precise, efficient, and cost-effective data processing in AWS Entity Resolution, giving you greater control and flexibility in managing your entity resolution tasks.

You can create a partitioned AWS Glue table for the source account in an ID mapping workflow.

You must first catalog the input data in Amazon S3 in AWS Glue and represented it as an AWS Glue table. For more information about how to create an AWS Glue table with Amazon S3 as the input, see Working with crawlers on the AWS Glue console in the *AWS Glue Developer Guide.*

In this step, you set up a crawler in AWS Glue that crawls all the files in your S3 bucket and then create a partitioned AWS Glue table.

> **Note**
>
> AWS Entity Resolution doesn't currently support Amazon S3 locations registered with AWS Lake Formation.

To create a partitioned AWS Glue table

1. Sign in to the AWS Management Console and open the AWS Glue console at https://console.aws.amazon.com/glue/.
2. From the navigation bar, select **Crawlers**.
3. Select your S3 bucket from the list, and then choose **Create crawler**.
4. On the **Set crawler properties** page, enter a crawler **Name**, optional **Description**, and then choose **Next**.
5. Continue through the **Add crawler page**, specifying the details.
6. On the **Choose an IAM role** page, choose **Choose an existing IAM role** and then choose **Next**.

   You can also choose **Create an IAM role** or have your administrator create the IAM role if needed.

7. For **Create a schedule for this crawler**, keep the **Frequency** default (**Run on demand**) and then choose **Next**.
8. For **Configure the crawler's output**, enter the AWS Glue database and then choose **Next**.
9. Review all the details, and then choose **Finish**.
10. On the **Crawlers** page, select the check box next to your S3 bucket and then choose **Run crawler**.
11. After the crawler is finished running, on the AWS Glue navigation bar, choose **Databases**, and then choose your database name.
12. On the **Database** page, under **Tables**, choose the table to be partitioned.
13. On the **Table overview**, select the **Actions** dropdown, and then choose **Edit table**.

    a. Under **Table properties**, choose **Add**.

    b. For the new **Key**, enter **`aerPushDownPredicateString`**.

    c. For the new **Value**, enter **`'<PartitionKey>=<PartitionValue'`**.

    d. Make a note of the AWS Glue database name and AWS Glue table name.

You are now ready to:

- Create a schema mapping and then create an ID mapping workflow for one AWS account.
- Create an ID namespace source, create an ID namespace target, and then create an ID mapping workflow across two AWS accounts.

Preparing third-party input data

Third-party data services provide identifiers that can be matched with your known identifiers.

AWS Entity Resolution currently supports the following third-party data provider services:

Data provider services

Company Name	Available AWS Regions	Identifier
LiveRamp	US East (N. Virginia) (us-east-1), US East (Ohio) (us-east-2), and US West (Oregon) (us-west-2)	Ramp ID

Company Name	Available AWS Regions	Identifier
TransUnion	US East (N. Virginia) (us-east-1), US East (Ohio) (us-east-2), and US West (Oregon) (us-west-2)	TransUnion Individual and Household IDs
Unified ID 2.0	US East (N. Virginia) (us-east-1), US East (Ohio) (us-east-2), and US West (Oregon) (us-west-2)	raw UID 2

The following steps describe how to prepare third-party data to use a provider service-based matching workflow or a provider service-based ID mapping workflow.

Topics

Step 1: Subscribe to a provider service on AWS Data Exchange

If you have a subscription with a provider service through AWS Data Exchange, you can run a matching workflow with one of the following provider services to match your known identifiers with your preferred provider. Your data will be matched with a set of inputs defined by your preferred provider.

To subscribe to a provider service on AWS Data Exchange

1. View the provider listing on AWS Data Exchange. The following provider listings are available:
    - LiveRamp
        - LiveRamp Identity Resolution
        - LiveRamp Transcoding

- TransUnion
  - TruAudience Identity Resolution & Enrichment
- Unified ID 2.0
  - Unified ID 2.0 Identity Resolution

2. Complete one of the following steps, depending on your offer type.

- **Private offer** – If you have an existing relationship with a provider, follow the Private products and offers procedure in the *AWS Data Exchange User Guide* to accept a private offer on AWS Data Exchange.
- **Bring your own subscription** – If you already have an existing data subscription with a provider, follow the Bring Your Own Subscription (BYOS) offers procedure in the *AWS Data Exchange User Guide* to accept a BYOS offer on AWS Data Exchange.

3. After you have subscribed to a provider service on AWS Data Exchange, you can then create a matching workflow or an ID mapping workflow with that provider service.

For more information about how to access a provider product that contains APIs, see Accessing an API product in the in the *AWS Data Exchange User Guide*.

Step 2: Prepare third-party data tables

Each third-party service has a different set of recommendations and guidelines to help ensure a successful matching workflow.

To prepare third-party data tables, consult the following table:

Data provider services guidelines

Provider service	Unique ID needed?	Actions
LiveRamp	Yes	Ensure the following: • The Unique ID can be either your own pseudonymous identifier or a row ID. • Your data input file format and normaliza tion is aligned with the LiveRamp guidelines.

Provider service	Unique ID needed?	Actions
		For more information about input file formatting guidelines for the matching workflow, see Perform Identity Resolution Through ADX in the LiveRamp documentation. For more information about input file formatting guidelines for the ID mapping workflow, see Perform Transcoding Through ADX in the LiveRamp documentation.

Provider service	Unique ID needed?	Actions

TransUnion	Yes	Ensure the following are a `string` type column in the input view:
		• Unique ID is required and can be a CRM ID, a contact ID, a user ID or any unique ID.
		• **`Name`**
		  • **`First Name`** can be lower or upper case, nicknames are supported, but titles and suffixes should be excluded.
		  • **`Last Name`** can be lower or upper case, middle initials to be excluded.
		• **`Address`**
		  • **`Street address1`** and **`Street address1`** is combined into a single **`Full address`** line, if present.
		  • **`City`** is separated from the **`Full address`**.
		  • **`Zip`** (or **`zip plus4`**), without any special characters such as spaces, hyphens, or blanks. Use nulls if no data.
		  • **`State`** is specified as a 2-letter code in upper case.
		• • **`Phone`**
		  • **`Phone number`** should be 10 digits, without any special characters such as spaces or hyphens.
		• **`Email addresses`** is either plaintext or SHA256-hashed lower case strings.
		• **`Date of Birth`** is in `yyyy-mm-dd` format.
		• **`Digital identifiers`** (Device IDs) can include IDs with hyphens (36-character

Provider service	Unique ID needed?	Actions
		length raw Device IDs/MAIDs/IFAs) and without hyphens (32 & 40-character long hashed Device IDs/MAIDs/IFAs). • **`IPV4`** is a 32-bit IP address expressed in dotted decimal notation. For example: `192.0.2.1` • **`IPV6`** is a 128-bit IP address expressed in hexadecimal notation, separated by colons. For example: `2001:db8:0000:0000:0000:0000:0000:0001` • **`MAID`** (Mobile Advertising ID) is a unique, alphanumeric string assigned to a mobile device for advertising purposes. A MAID usually has 36 characters. For example: `a1b2c3d4-5678-90ab-cdef-EXAMPLE11111`

Provider service	Unique ID needed?	Actions
Unified ID 2.0	Yes	Ensure the following: • The Unique ID can't be a hash. • Either **`Phone number`** or **`Email addresses`** is used in the schema, not both. • UID2 supports both email and phone number for UID2 generation. However, if both values are present in the schema mapping, the workflow duplicates each record in the output. One record uses the email for UID2 generation and the second record uses phone number. If your data includes a mix of emails and phone numbers and you don't want this duplicati on of records in the output, the best approach is to create a separate workflow for each, with separate schema mappings. In this scenario, go through the steps twice —create one workflow for emails and a separate one for phone numbers. **Note** A specific email or phone number, at any specific time, results in the same raw UID2 value, no matter who made the request. Raw UID2s are created by adding salts from salt buckets which are rotated approximately once a year, causing the raw UID2 to also be rotated with it. Different salt buckets rotate at different times throughou

Provider service	Unique ID needed?	Actions
		t the year. AWS Entity Resolutio n currently doesn't keep track of rotating salt buckets and raw UID2s, so it is recommended that you regenerate the raw UID2s daily. For more information, see How often should UID2s be refreshed for incremental updates? in the UID 2.0 documentation.

Step 3: Save your input data table in a supported data format

If you already saved your third-party input data in a supported data format, you can skip this step.

To use AWS Entity Resolution, the input data must be in a format that AWS Entity Resolution supports.

AWS Entity Resolution supports the following data formats:

- comma-separated value (CSV)

> **Note**
>
> LiveRamp only supports CSV files.

- Parquet

Step 4: Upload your input data table to Amazon S3

If you already have your third-party data table in Amazon S3, you can skip this step.

> **Note**
>
> The input data must be stored in Amazon Simple Storage Service (Amazon S3) in the same AWS account and AWS Region in which you want to run the matching workflow.

To upload your input data table to Amazon S3

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Buckets**, and then choose a bucket to store your data table.
3. Choose **Upload**, and then follow the prompts.
4. Choose the **Objects** tab to view the prefix where your data is stored. Make a note of the name of the folder.

   You can select the folder to view the data table.

Step 5: Create an AWS Glue table

The input data in Amazon S3 must be cataloged in AWS Glue and represented as an AWS Glue table. For more information about how to create an AWS Glue table with Amazon S3 as the input, see Working with crawlers on the AWS Glue console in the *AWS Glue Developer Guide.*

> **Note**
>
> AWS Entity Resolution doesn't support partitioned tables.

In this step, you set up a crawler in AWS Glue that crawls all the files in your S3 bucket and create an AWS Glue table.

> **Note**
>
> AWS Entity Resolution doesn't currently support Amazon S3 locations registered with AWS Lake Formation.

To create an AWS Glue table

1. Sign in to the AWS Management Console and open the AWS Glue console at https://console.aws.amazon.com/glue/.
2. From the navigation bar, select **Crawlers**.
3. Select your S3 bucket from the list, and then choose **Add crawler**.

4. On the **Add crawler** page, enter a **Crawler name** and then choose **Next**.
5. Continue through the **Add crawler page**, specifying the details.
6. On the **Choose an IAM role** page, choose **Choose an existing IAM role** and then choose **Next**.

   You can also choose **Create an IAM role** or have your administrator create the IAM role if needed.
7. For **Create a schedule for this crawler**, keep the **Frequency** default (**Run on demand**) and then choose **Next**.
8. For **Configure the crawler's output**, enter the AWS Glue database and then choose **Next**.
9. Review all of the details, and then choose **Finish**.
10. On the **Crawlers** page, select the check box next to your S3 bucket and then choose **Run crawler**.
11. After the crawler is finished running, on the AWS Glue navigation bar, choose **Databases**, and then choose your database name.
12. On the **Database** page, choose **Tables in {your database name}**.

    a. View the tables in the AWS Glue database.

    b. To view a table's schema, select a specific table.

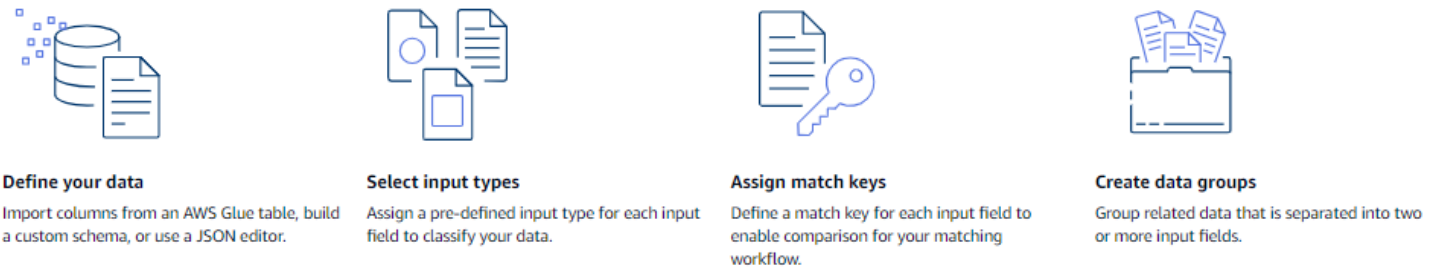
    c. Make a note of the AWS Glue database name and AWS Glue table name.

You are now ready to create a schema mapping. For more information, see Creating a schema mapping.

Define input data using schema mapping

A *schema mapping* defines the input data that you want to resolve. It also provides metadata about the input data, such as the attribute types of the columns (input fields) and which columns to match on.

When you create a schema mapping, you first define your input fields and attribute types, and then define your match keys and group related data. The following diagram summarizes how to create a schema mapping.



Before you create a schema mapping, you must first set up AWS Entity Resolution and prepare your data tables. For more information, see Set up AWS Entity Resolution and Prepare input data tables.

After you create a schema mapping, you can do one of the following:

- Create a matching workflow to find matches between different data inputs.
- Create an ID namespace source that you can use in an ID mapping workflow to translate data from a source to a target.
- Create an ID mapping workflow within the same AWS account using your schema mapping as the source.

Topics

- Creating a schema mapping
- Cloning a schema mapping
- Editing a schema mapping
- Deleting a schema mapping

Creating a schema mapping

This procedure describes the process of creating a schema mapping using the AWS Entity Resolution console.

There are three ways to create a schema mapping:

- Import existing input data using the **Import from AWS Glue** option – Use this creation method to define input fields starting with pre-populated columns from an AWS Glue table using a guided flow.
- Manually defining input data using the **Build custom schema** option – Use this creation method to manually define the input fields using a guided flow.
- Manually create using the **Use JSON editor** option – Use a JSON editor to manually create, use a sample, or import existing input data.

> **Note**
>
> The **Unique ID** and **Input fields** aren't available with this option.

Import from AWS Glue

To create schema mapping by importing existing input data from AWS Glue

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Data preparation**, choose **Schema mappings**.
3. On the **Schema mappings** page, in the upper right corner, choose **Create schema mapping**.
4. For **Step 1: Specify schema details**, do the following:
   a. For **Name and creation method**, enter a **Schema mapping name** and an optional **Description**.
   b. For **Creation method**, choose **Import from AWS Glue**.
   c. Choose the **AWS Glue database** from the dropdown, and then choose the **AWS Glue table** from the dropdown.

To create a new table, go to the AWS Glue console https://console.aws.amazon.com/glue/. For more information, see AWS Glue tables in the *AWS Glue User Guide*.

d. For **Unique ID**, specify the column that distinctly references each row of your data.

Example

For example: **`Primary_key`**, **`Row_ID`**, or **`Record_ID`**.

> **Note**
>
> The **Unique ID** column is required. The **Unique ID** must be a unique identifier within a single table. However, across different tables, the **Unique ID** can have duplicate values. If the **Unique ID** isn't specified, isn't unique within the same source, or overlaps in terms of attribute names across sources, then AWS Entity Resolution rejects the record when the matching workflow is run. If you are using this schema mapping in a rule-based matching workflow, the **Unique ID** must not exceed 38 characters.

e. For **Input fields**, choose the columns you want to use for matching and for optional pass through.

You can choose a maximum of 34 columns total for both matching and pass through.

i. Under **Matching**, choose the columns you to use as input fields for matching.

You can choose a maximum of 24 columns total for matching.

ii. Select **Add columns for pass through** if you want to specify the columns that aren't used for matching.

iii. (Optional) Under **Pass through**, choose the columns to include as pass through columns.

f. (Optional) If you want to enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.

g. Choose **Next**.

5. For **Step 2: Map input fields**, define the input fields you want to use for matching and for optional pass through.

a. For **Input fields for matching**, for each **Input field**,

- Specify the **Attribute type** to classify the data.
- Specify the **Match key name** to enable input field comparison to your matching workflow. Certain match key names are automatically associated with specific attribute types by default.
- Select the **Hashed** checkbox if the column value for that input field is hashed or leave the checkbox blank if the value is cleartext.

> **Note**
>
> If you're creating a schema mapping to use with the LiveRamp provider service-based matching technique, then you can:
>
> - Specify the **Attribute type** for the Provider ID as **LiveRamp ID**.
> - Specify the **Attribute type** for the **name** field as either multiple fields (such as **First name**, **Last name**) or in one field.
> - Specify the **Attribute type** for the **street address** field as either multiple fields (such as **Street address 1**, **Street address 2**,) or in one field (**Full address**).
>
> If matching against an address, a zip code (**Postal code**) is required.
> - If you include email (**Email address**) or phone (**Phone number**) with a name, those fields can match against the street address.

> **Note**
>
> If you're creating a schema mapping to use with the TransUnion provider service-based matching technique, then you can specify any of the following **Attribute types**:
>
> - **Full name**, **First name**, **Last name**
> - **Full address**, **Street address 1**, **City**, **State**, **Country**, **Postal code**
> - **Phone number**
> - **Email address**
> - **Date**

- **Digital Identifiers**: **IPV4**, **IPV6**, or **MAID**

> **Note**
>
> If you're creating a schema mapping to use with the machine learning-based matching workflow, your dataset must contain at least one of the following **Attribute types**:
>
> - **Full name**
> - **Full address**
> - **Full phone**
> - **Email address**
> - **Date** with a **Match key name** of **Date of birth**
>
> Don't specify the **Attribute type** for any of these attributes as a **Custom string**.

b. (Optional) For **Input fields for pass through**, add the input fields that won't be matched and their corresponding **Hashing status**.

The **Hashing status** indicates if the column value for that input field is hashed or cleartext.

c. Choose **Next**.

6. For **Step 3: Group data**, you can group the **Name**, **Address**, and **Phone number** input fields if they have been separated into multiple fields.

This step concatenates the related input fields into one field, which enables you to compare them as one field in a matching workflow.

If you don't have any data mapped to the **Name**, **Address**, or **Phone number** input fields, then this section will be blank.

You can also add more groups if you have more types of data.

a. If you want to group **Name** input data:

For **Full name**, choose two or more **Input fields** you want to group.

The **Group name** and **Match key** are automatically associated with the data type.

You can update the **Group name** and the **Match key** with a custom match key can contain up to 255 characters, including letters, numbers, underscores (_), or hyphens (-).

Choose **Add group** to add another group.

> **Note**
>
> Normalization is only supported for **Full name**.
> If you want to normalize the **Full name** subtypes, then assign the following subtypes to the **Full name** group: **First name**, **Middle name**, and **Last name**.

b. If you want to group **Address** input data:

For **Full address**, choose two or more **Input fields** fields you want to group.

The **Group name** and **Match key**. are automatically associated with the data type.

You can update the **Group name** and the **Match key** with a custom match key can contain up to 255 characters, including letters, numbers, underscores (_), or hyphens (-).

Choose **Add group** to add another group.

> **Note**
>
> Normalization is only supported for **Full address**.
> If you want to normalize the **Full address** subtypes, then assign the following subtypes to the **Full address** group: **Street address 1**, **Street address 2**: **Street address 3 name**, **City name**, **State**, **Country**, and **Postal code**.

c. If you want to group **Phone** input data:

For **Full phone**, choose two or more **Input fields** fields you want to group.

The **Group name** and **Match key**. are automatically associated with the data type.

You can update the **Group name** and the **Match key** with a custom match key can contain up to 255 characters, including letters, numbers, underscores (_), or hyphens (-).

Choose **Add group** to add another group.

> **Note**
>
> Normalization is only supported for **Full phone**.
> If you want to normalize the **Full phone** subtypes, then assign the following subtypes to the **Full phone** group: **Phone number**, and **Phone country code**.

d. Choose **Next**.

7. For **Step 4: Review and create**, do the following:

a. Review the selections that you made for the previous steps and edit if necessary.

b. Choose **Create schema mapping**.

> **Note**
>
> You can't modify a schema mapping after you associate it to a workflow. You can clone a schema mapping if you want to use an existing configuration to create a new schema mapping.

After you create the schema mapping, you're ready to create a matching workflow or create an ID namespace.

Build custom schema

To create a schema mapping using the Build custom schema option

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Data preparation**, choose **Schema mappings**.
3. On the **Schema mappings** page, in the upper right corner, choose **Create schema mapping**.
4. For **Step 1: Specify schema details**, do the following:

a. For name and creation method, enter a **Schema mapping name** and an optional **Description**.

b. For **Creation method**, choose **Build custom schema**.

c. For **Unique ID**, enter a unique ID to identify each row of your data.

Example

For example: **`Primary_key`**, **`Row_ID`**, or **`Record_ID`**.

> **Note**
>
> The **Unique ID** column is required. The **Unique ID** must be a unique identifier within a single table. However, across different tables, the **Unique ID** can have duplicate values. If the **Unique ID** isn't specified, isn't unique within the same source, or overlaps in terms of attribute names across sources, then AWS Entity Resolution rejects the record when the matching workflow is run. If you are using this schema mapping in a rule-based matching workflow, the **Unique ID** must not exceed 38 characters.

d. (Optional) If you want to enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.

e. Choose **Next**.

5. For **Step 2: Map input fields**, define the input fields you want to use for matching and for optional pass through.

   You can define a maximum of 34 columns total for both matching and pass through.

   a. For **Input fields for matching**, enter an **Input field**.

   b. Select the **Attribute type** to classify the data.

   > **Note**
   >
   > If you're creating a schema mapping to use with the LiveRamp provider service-based matching technique, then you can specify the providerID **Attribute type** as **LiveRamp ID**. If you want to include PII data in the output, then you must specify the **Attribute type** as **Custom string**.

> **Note**
>
> If you're creating a schema mapping to use with the TransUnion provider service-based matching technique, then you can specify any of the following **Attribute types**:
>
> - **Full name**, **First name**, **Last name**
> - **Full address**, **Street address 1**, **City**, **State**, **Country**, **Postal code**
> - **Phone number**
> - **Email address**
> - **Date**
> - **Digital Identifiers**: **IPV4**, **IPV6**, or **MAID**

> **Note**
>
> If you're creating a schema mapping to use with the machine learning-based matching workflow, your dataset must contain at least one of the following **Attribute types**:
>
> - **Full name**
> - **Full address**
> - **Full phone**
> - **Email address**
> - **Date** with a **Match key name** of **Date of birth**
>
> Don't specify the **Attribute type** for any of these attributes as a **Custom string**.

c. Select the **Match key name** to enable input field comparison to your matching workflow.

   Certain match key names are automatically associated with specific attribute types by default.

d. Select the **Hashed** checkbox if the column value for that input field is hashed or leave the checkbox blank if the value is cleartext.

e. Choose **Add input field** to add more input fields.

You can add a maximum of 24 input fields total for matching.

f. (Optional) For **Input fields for pass through**, add the input fields that won't be matched and their corresponding **Hashing status**.

g. Choose **Next**.

6. For **Step 3: Group data**, you can group the **Name**, **Address**, **Phone number** input fields if they have been separated into multiple fields.

This step concatenates the related input fields into one field, which enables you to compare them as one field in a matching workflow.

If you don't have any data mapped to **Name**, **Address**, **Phone number** input fields, then this section will be blank.

You can also add more groups if you have more types of data.

a. If you want to group **Name** input data:

For **Full name**, choose two or more **Input fields** you want to group.

The **Group name** and **Match key** are automatically associated with the data type.

You can update the **Group name** and the **Match key** with a custom match key can contain up to 255 characters, including letters, numbers, underscores (_), or hyphens (-).

Choose **Add group** to add another group.

> **Note**
>
> Normalization is only supported for **Full name**.
> If you want to normalize the **Full name** subtypes, then assign the following subtypes to the **Full name** group: **First name**, **Middle name**, and **Last name**.

b. If you want to group **Address** input data:

For **Full address**, choose two or more **Input fields** fields you want to group.

The **Group name** and **Match key**. are automatically associated with the data type.

You can update the **Group name** and the **Match key** with a custom match key can contain up to 255 characters, including letters, numbers, underscores (_), or hyphens (-).

Choose **Add group** to add another group.

> **Note**
>
> Normalization is only supported for **Full address**.
> If you want to normalize the **Full address** subtypes, then assign the following subtypes to the **Full address** group: **Street address 1**, **Street address 2**: **Street address 3 name**, **City name**, **State**, **Country**, and **Postal code**.

c. If you want to group **Phone** input data:

For **Full phone**, choose two or more **Input fields** fields you want to group.

The **Group name** and **Match key**. are automatically associated with the data type.

You can update the **Group name** and the **Match key** with a custom match key can contain up to 255 characters, including letters, numbers, underscores (_), or hyphens (-).

Choose **Add group** to add another group.

> **Note**
>
> Normalization is only supported for **Full phone**.
> If you want to normalize the **Full phone** subtypes, then assign the following subtypes to the **Full phone** group: **Phone number**, and **Phone country code**.

d. Choose **Next**.

7. For **Step 4: Review and create**, do the following:

a. Review the selections that you made for the previous steps and edit if necessary.

b. Choose **Create schema mapping**.

> **Note**
>
> You can't modify a schema mapping after you associate it with a workflow. You can clone a schema mapping if you want to use an existing configuration to create a new schema mapping.

After you create the schema mapping, you're ready to create a matching workflow or create an ID namespace.

Use JSON editor

To create a schema mapping by using the JSON editor

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Data preparation**, choose **Schema mappings**.
3. On the **Schema mappings** page, in the upper right corner, choose **Create schema mapping**.
4. For **Step 1: Specify schema details**, do the following:
   a. For name and creation method, enter a **Schema mapping name** and an optional **Description**.
   b. For **Creation method**, choose **Use JSON editor**.
   c. (Optional) If you want to enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.
   d. Choose **Next**.
5. For **Step 2: Specify mapping**:
   a. Start building the schema in the JSON editor or choose one of the following options based on your goal:

Your goal	Recommended option
Start building your schema mapping	**Insert sample JSON** and then edit the information as necessary.
Use an existing JSON file	**Import from file**

> **Note**
>
> Normalization is only supported for the following **types**: `NAME`, `ADDRESS`, `PHONE`, and `EMAIL_ADRESS`.
> If you want to normalize the `NAME` subtypes, then assign the following subtypes to the `NAME` **groupName**: `NAME_FIRST`, `NAME_MIDDLE`, and `NAME_LAST`
> If you want to normalize the `ADDRESS` subtypes, then assign the following subtypes to the `ADDRESS` **groupName**: `ADDRESS_STREET1`, `ADDRESS_STREET2`, `ADDRESS_STREET3`, `ADDRESS_CITY`, `ADDRESS_STATE`, `ADDRESS_COUNTRY`, and `ADDRESS_POSTALCODE`.
> If you want to normalize the `PHONE` subtypes, then assign the following subtypes to the `PHONE` **groupName**: `PHONE_NUMBER` and `PHONE_COUNTRYCODE`.

b. Choose **Next**.

6. For **Step 3: Review and create**:

   a. Review the selections that you made for the previous steps and edit if necessary.

   b. Choose **Create schema mapping**.

> **Note**
>
> You can't modify a schema mapping after you associate it with a workflow. You can clone a schema mapping if you want to use an existing configuration to create a new schema mapping.

After you create the schema mapping, you're ready to create a matching workflow or create an ID namespace.

Cloning a schema mapping

You can clone a schema mapping if you want to use an existing configuration to create a new schema mapping.

To clone a schema mapping:

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Data preparation**, choose **Schema mappings**.
3. Choose the schema mapping.
4. Choose **Clone**.
5. On the **Specify schema details** page, make any necessary changes and then choose **Next**.
6. On the **Choose matching technique** page, make any necessary changes and then choose **Next**.
7. On the **Map input fields** page, make any necessary changes and then choose **Next**.
8. On the **Group data** page, make any necessary changes and then choose **Next**.
9. On the **Review and save** page, make any necessary changes and then choose **Clone schema mapping**.

Editing a schema mapping

You can only edit a schema mapping before you associate it to a workflow. After you've associated a schema mapping to a workflow, you can't edit it. You can clone a schema mapping if you want to use an existing configuration to create a new schema mapping.

To edit a schema mapping:

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Data preparation**, choose **Schema mappings**.
3. Choose the schema mapping.
4. Choose **Edit**.

5. On the **Specify schema details** page, make any necessary changes and then choose **Next**.
6. On the **Choose matching technique** page, make any necessary changes and then choose **Next**.
7. On the **Map input fields** page, make any necessary changes and then choose **Next**.
8. On the **Group data** page, make any necessary changes and then choose **Next**.

> **Note**
>
> Normalization is only supported for the **Full name**, **Full address**, **Full phone**, and **Email address**.
> If you want to normalize the **Full name** sub-types, then assign the following subtypes to the **Full name** group: **First name**, **Middle name**, and **Last name**.
> If you want to normalize the **Full address** sub-types, then assign the following subtypes to the **Full address** group: **Street address 1**, **Street address 2**: **Street address 3 name**, **City name**, **State**, **Country**, and **Postal code**.
> If you want to normalize the **Full phone** sub-types, then assign the following subtypes to the **Full phone** group: **Phone number**, and **Phone country code**.

9. On the **Review and save** page, make any necessary changes and then choose **Edit schema mapping**.

Deleting a schema mapping

You can't delete a schema mapping when it's associated to a matching workflow. You must first remove the schema mapping from all associated matching workflows before you can delete it.

To delete a schema mapping:

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Data preparation**, choose **Schema mappings**.
3. Choose the schema mapping.
4. Choose **Delete**.
5. Confirm the deletion and then choose **Delete**.

Define input data using an ID namespace

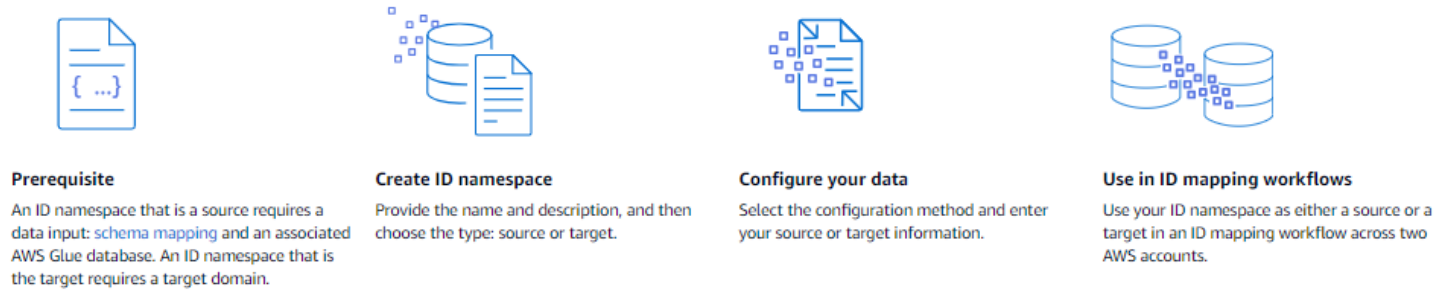
An *ID namespace* is a wrapper around your input data table. You use an ID namespace to provide metadata explaining your input data and matching techniques and how to use them in an ID mapping workflow.

There are two types of ID namespaces: **Source** and **Target**.

- The **Source** contains configurations for the source data that AWS Entity Resolution processes in an ID mapping workflow.
- The **Target** contains a configuration of the target data that all sources resolve to.

You can define the input data that you want to resolve across two AWS accounts in an ID mapping workflow. One participant creates an ID namespace source and another participant creates an ID namespace target. After the participants create the source and target, you can run an ID mapping workflow to translate the data from the source to the target.

The following diagram summarizes how to create an ID namespace to use in an ID mapping workflow.



The following sections describe how to create an ID namespace source and an ID namespace target.

Topics

- ID namespace source
- ID namespace target
- Editing an ID namespace
- Deleting an ID namespace
- Adding or updating a resource policy for an ID namespace

ID namespace source

The *ID namespace source* is the source of the data in an ID mapping workflow.

Before you create an ID namespace source you must first create a schema mapping or a matching workflow, depending on your use case. For more information, see Creating a schema mapping and Match input data using a matching workflow.

After you create an ID namespace source, you can use it along with an ID namespace target in an ID mapping workflow. For more information, see Map input data using an ID mapping workflow.

There are two ways to create an ID namespace source in the AWS Entity Resolution console: the rule-based method or the provider services method.

Topics

Creating an ID namespace source (rule-based)

This topic describes the process of creating an ID namespace source using the **rule-based** method. This method uses matching rules to translate first-party data from a source to a target in an ID mapping workflow.

> **Note**
>
> If the input data is the source, then it must have a schema mapping and an associated AWS Glue database.

To create an ID namespace source (rule-based)

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Data preparation**, choose **ID namespaces**.
3. On the **ID namespaces** page, in the upper right corner, choose **Create ID namespace**.
4. For **Details**, do the following:

a. For **ID namespace name**, enter a unique name.

b. (Optional) For **Description**, enter an optional description.

c. For **ID namespace type**, choose **Source**.

5. For the **ID namespace method**, choose **Rule-based**.

6. For **Data input**, choose the **Input type** that you want to use and then take the recommended actions.

Input type	Recommended actions
An existing schema mapping	1. Choose **Schema mapping**. 2. Choose the **AWS Glue database**, the **AWS Glue table**, and the **Schema mapping** from the dropdown list. You can add up to 20 data inputs.
An existing matching workflow	1. Choose the **Matching workflow**. 2. Choose the account that's associated with the ID namespace: either **Your AWS account** or **Another AWS account**. 3. Depending on the type of account, select the **Matching workflow name** or enter the **Matching workflow ARN**.

7. For **Rule parameters**, do the following.

a. Specify the **Rule controls** by choosing one of the following options based on your goal.

Your goal	Recommended option
Allow rules from both the source and the target	**No preference**
Choose whether a source, target, or both can provide rules in an ID mapping workflow	**Limited rules**

Rule controls must be compatible between the source and the target to be used in an ID mapping workflow. For example, if a source ID namespace limits rules to the target but the target ID namespace limits rules to the source, this results in an error.

b. Specify the **Matching rules** by choosing one of the following options based on your data input type.

Data input type	Recommended action
Schema mapping	Choose **Add another rule** to add a matching rule. You can apply up to 25 **Matching rules** to define your match criteria.
Matching workflow	Choose either **Use rules from matching workflow** or **Provide new rules** to define your **Matching rules**.

8. For **Comparison and matching parameters**, do the following.

a. Specify the **Comparison type** by choosing one of the following options based on your goal.

Your goal	Recommended option
Allow any comparison type to be used when you create the ID mapping workflow.	**No preference**
Find any combination of matches across data stored in multiple input fields, regardless of whether the data is in the same or different input field.	**Multiple input fields**
Limit comparison within a single input field, when similar data stored across	**Single input field**

Your goal	Recommended option
multiple input fields shouldn't be matched.	

b. Specify the **Record matching type** by choosing one of the following options based on your goal.

Your goal	Recommended option
Allow any comparison type to be used when you create the ID mapping workflow.	**No preference**
Limit the record matching type to store only one matching record in the source for each matched record in the target when you create the ID mapping workflow.	**Limited record matching** and **One source to one target**
Limit the record matching type to store all matching records in the source for each matched record in the target when you create the ID mapping workflow.	**Limited record matching** and **Many sources to one target**

> **Note**
>
> You must specify compatible limitations for the source and target ID namespaces. For example, if a source ID namespace limits rules to the target but the target ID namespace limits rules to the source, this results in an error.

9. Specify the **Service access permissions** by choosing an **Existing service role name** from the dropdown list.
10. (Optional) To enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.
11. Choose **Create ID namespace**.

The ID namespace source is created. You are now ready to create an ID namespace target.

Creating an ID namespace source (provider services)

This topic describes the process of creating an ID namespace source using the **Provider services** method. This method uses a provider service called LiveRamp. LiveRamp translates third-party encoded data from a source to a target during an ID mapping workflow.

> **Note**
>
> If the input data is the source, then it must have a schema mapping and an associated AWS Glue database.

To create an ID namespace source (provider services)

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Data preparation**, choose **ID namespaces**.
3. On the **ID namespaces** page, in the upper right corner, choose **Create ID namespace**.
4. For **Details**, do the following:
   a. For **ID namespace name**, enter a unique name.
   b. (Optional) For **Description**, enter an optional description.
   c. For **ID namespace type**, choose **Source**.
5. For the **ID namespace method**, choose **Provider services**.

   > **Note**
   >
   > AWS Entity Resolution currently offers the LiveRamp provider service as an ID namespace method. If you have a subscription to LiveRamp, then the status appears as **Subscribed**. For more information about how to subscribe to LiveRamp, see Step 1: Subscribe to a provider service on AWS Data Exchange.

6. For **Data input**, choose the **AWS Glue database**, the **AWS Glue table**, and the **Schema mapping** from the dropdown list.

   You can add up to 20 data inputs.

7. To specify the **Service access** permissions, choose an option and take the recommended action.

Option	Recommended action
Create and use a new service role	• AWS Entity Resolution creates a service role with the required policy for this table. • The default **Service role name** name is `entityresolution-id-mapping-workflow-<timestamp>` . • You must have permissions to create roles and attach policies. • If your input data is encrypted, choose the **This data is encrypted by a KMS key** option. Then, enter an **AWS KMS key** that is used to decrypt your data input.
Use an existing service role	1. Choose an **Existing service role name** from the dropdown list. The list of roles are displayed if you have permissions to list roles. If you don't have permissions to list roles, you can enter the Amazon Resource Name (ARN) of the role that you want to use. If there are no existing service roles, the option to **Use an existing service role** is unavailable. 2. View the service role by choosing the **View in IAM** external link. By default, AWS Entity Resolution doesn't attempt to update the existing role policy to add necessary permissions.

8. (Optional) To enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.
9. Choose **Create ID namespace**.

The ID namespace source is created. You are now ready to create an ID namespace target.

ID namespace target

The *ID namespace target* is the target of the data in an ID mapping workflow. All sources resolve to the target.

Before you create an ID namespace target you must first create a matching workflow or have a subscription to a provider service (LiveRamp), depending on your use case. For more information, see Match input data using a matching workflow and Step 1: Subscribe to a provider service on AWS Data Exchange.

After you create an ID namespace target, you can use it along with an ID namespace source in an ID mapping workflow. For more information, see Map input data using an ID mapping workflow.

There are two ways to create an ID namespace target in the AWS Entity Resolution console: the rule-based method or the provider services method.

Topics

- Creating an ID namespace target (rule-based method)
- Creating an ID namespace target (provider services method)

Creating an ID namespace target (rule-based method)

This topic describes the process of creating an ID namespace target using the **rule-based** method. This method uses matching rules to translate first-party data from a source to a target during an ID mapping workflow.

To create an ID namespace target (rule-based)

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Data preparation**, choose **ID namespaces**.

3. On the **ID namespaces** page, in the upper right corner, choose **Create ID namespace**.
4. For **Details**, do the following:
   a. For **ID namespace name**, enter a unique name.
   b. (Optional) For **Description**, enter an optional description.
   c. For **ID namespace type**, choose **Target**.
5. For the **ID namespace method**, choose **Rule-based**.
6. For **Data input**, under **Matching workflow**, do the following.
   a. Choose the account that's associated with the ID namespace: either **Your AWS account** or **Another AWS account**.
   b. Depending to the type of account, select the **Matching workflow name** or enter the **Matching workflow ARN**.
7. For **Rule parameters**, do the following.
   a. Specify the **Rule controls** by choosing one of the following options based on your goal.

Your goal	Recommended option
Allow rules from both the source and the target	**No preference**
Choose whether a source, target, or both can provide rules in an ID mapping workflow	**Limited rules**

   Rule controls must be compatible between the source and the target to be used in an ID mapping workflow. For example, if a source ID namespace limits rules to the target but the target ID namespace limits rules to the source, this results in an error.
   b. For **Matching rules**, AWS Entity Resolution automatically adds the rules from the matching workflow.
8. For **Comparison and matching parameters**, do the following.
   a. Specify the **Comparison type** by choosing one of the following options based on your goal.

Your goal	Recommended option
Allow any comparison type to be used when you create the ID mapping workflow.	**No preference**
Find any combination of matches across data stored in multiple input fields, regardless of whether the data is in the same or different input field.	**Multiple input fields**
Limit comparison within a single input field, when similar data stored across multiple input fields shouldn't be matched.	**Single input field**

b. Specify the **Record matching type** by choosing one of the following options based on your goal.

Your goal	Recommended option
Allow any comparison type to be used when you create the ID mapping workflow.	**No preference**
Limit the record matching type to store only one matching record in the source for each matched record in the target when you create the ID mapping workflow.	**Limited record matching** and **One source to one target**
Limit the record matching type to store all matching records in the source for each matched record in the target when you create the ID mapping workflow.	**Limited record matching** and **Many sources to one target**

> **Note**
>
> You must specify compatible limitations for the source and target ID namespaces. For example, if a source ID namespace limits rules to the target but the target ID namespace limits rules to the source, this results in an error.

9. Specify the **Service access permissions** by choosing an **Existing service role name** from the dropdown list.
10. (Optional) To enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.
11. Choose **Create ID namespace**.

The ID namespace target is created. After you create the ID namespaces (source and target) required for an ID mapping workflow, you're ready to create an ID mapping workflow.

Creating an ID namespace target (provider services method)

This topic describes the process of creating an ID namespace target using the **Provider services** method. This method uses a provider service called LiveRamp. LiveRamp translates third-party encoded data from a source to a target during an ID mapping workflow.

To create an ID namespace target (provider services)

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Data preparation**, choose **ID namespaces**.
3. On the **ID namespaces** page, in the upper right corner, choose **Create ID namespace**.
4. For **Details**, do the following:
    a. For **ID namespace name**, enter a unique name.
    b. (Optional) For **Description**, enter an optional description.
    c. For **ID namespace type**, choose **Target**.
5. For **ID namespace method**, choose **Provider services**.

> **Note**
>
> AWS Entity Resolution currently offers the LiveRamp provider service as an ID namespace method.
> If you have a subscription to LiveRamp, then the status appears as **Subscribed**.
> For more information about how to subscribe to LiveRamp, see Step 1: Subscribe to a provider service on AWS Data Exchange.

6. For **Target domain**, enter the LiveRamp client domain identifier targeted for transcoding that LiveRamp provides.
7. (Optional) To enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.
8. Choose **Create ID namespace**.

The ID namespace target is created. After you create the ID namespaces (source and target) required for an ID mapping workflow, you're ready to Create the ID mapping workflow.

Editing an ID namespace

You can only edit an ID namespace before you associate it to an ID mapping workflow. After you've associated an ID namespace to an ID mapping workflow, you can't edit it.

To edit an ID namespace:

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Data preparation**, choose **ID namespaces**.
3. Choose the ID namespace.
4. Choose **Edit**.
5. On the **Edit ID namespace** page, make any necessary changes and then choose **Save**.

Deleting an ID namespace

You can't delete an ID namespace when it's associated to an ID mapping workflow. You must first remove the schema mapping from all associated ID mapping workflows before you can delete it.

To delete an ID namespace:

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Data preparation**, choose **ID namespaces**.
3. Choose the ID namespace.
4. Choose **Delete**.
5. Confirm the deletion and then choose **Delete**.

Adding or updating a resource policy for an ID namespace

A resource policy allows the creator of the ID mapping resource to access your ID namespace resource.

To add or update a resource policy

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **ID namespaces**.
3. Choose the ID namespace.
4. On the ID namespace details page, choose the **Permissions** tab.
5. In the **Resource policy** section, choose **Edit**.
6. Add or update the policy in the JSON editor.
7. Choose **Save changes**.

Match input data using a matching workflow

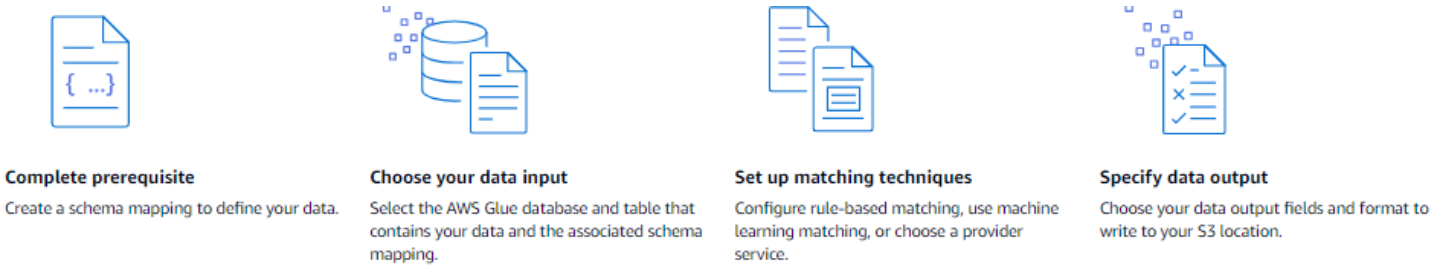
A *matching workflow* is a data processing job that combines and compares data from different input sources and determines which of it matches based on different matching techniques. It produces a data output table.

When you create a matching workflow, you first specify your data inputs, normalization steps, and then choose your desired matching techniques and data output. AWS Entity Resolution reads your data from your specified location or locations and finds a match between two or more records in your data. It then assigns a Match ID to the records in the matched set of data. AWS Entity Resolution then writes data output files to a location that you choose. You can use AWS Entity Resolution to hash output data if desired – helping you maintain control over your data.

A matching workflow can have multiple runs and the results (successes or errors) are written to a folder with the `jobId` as the name.

The data output contains both a file for successful matches and a file for errors. The data output can contain multiple fields. The successful results are written to a `success` folder that contains multiple files, and each file contains a subset of the successful records. Similarly, errors are written to an `error` folder with multiple fields, with each containing a subset of the error records. For more information about troubleshooting errors, see Troubleshooting matching workflows.

The following diagram summarizes how to create a matching workflow.



Before you create a matching workflow, you must first create a schema mapping. For more information, see Creating a schema mapping.

There are three ways to create a matching workflow, based on matching techniques: rule-based, machine learning-based, or provider service-based.

After you create and run a matching workflow, you can do the following:

- View the results in the S3 location you specified. Matching workflows generate IDs after the data is indexed.
- Use the output of rule-based matching or machine learning (ML) matching as an input to provider service-based matching or the other way around to meet your business needs.

For example, to save provider subscription costs, you can first run rule-based matching to find matches on your data. Then, you can send a subset of unmatched records to provider service-based matching.

Topics

Creating a rule-based matching workflow

Rule-based matching is a hierarchical set of waterfall matching rules, suggested by AWS Entity Resolution, based upon the data that you input and is completely configurable by you. The rule-based matching workflow enables you to compare cleartext or hashed data to find exact matches based on criteria that you customize.

When AWS Entity Resolution finds a match between two or more records in your data, it assigns:

- A Match ID to the records in the matched set of data
- The Match rule that generated the match.

When you create a rule-based matching workflow in AWS Entity Resolution, you must choose either a **Simple** or **Advanced** rule type. The rule type determines the complexity of rule conditions you can create. You can't change the rule type after creating the workflow.

You can use the following chart to compare the two **Rule types** and determine which one suits your use case.

Rule type comparison chart

Use case	Advanced rule type	Simple rule type
Schema mappings mapped one-to-one with input types	Yes	No
Schema mapping with multiple data columns mapped to the same input types	No	Yes
Supports Exact and Fuzzy matching	Yes	No (Exact matching only)
Supports AND, OR, and parentheses operators	Yes	No (AND operator only)
Supports batch workflows	Yes	Yes
Supports incremental workflows	Yes	Yes
Supports real-time workflows	NoYes	
Supports ID mapping workflows	No	Yes

After you have determined which rule type you want to use, use the following topics to create a rule-based matching workflow with either the **Advanced** or **Simple** rule type.

Topics

Creating a rule-based matching workflow with the Advanced rule type

The following procedure demonstrates how to create a rule-based matching workflow with the **Advanced** rule type using either the AWS Entity Resolution console or the `CreateMatchingWorkflow` API.

Console

To create a rule-based matching workflow with the Advanced rule type using the console

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **Matching**.
3. On the **Matching workflows** page, in the upper right corner, choose **Create matching workflow**.
4. For **Step 1: Specify matching workflow details**, do the following:
   a. Enter a **Matching workflow name** and an optional **Description**.
   b. For **Data input**, choose an **AWS Glue database** from the dropdown, select the **AWS Glue table**, and then the corresponding **Schema mapping**.

      You can add up to 19 data inputs.

      > **Note**
      >
      > To use **Advanced** rules, your schema mappings must meet the following requirements:
      >
      > 1. Each input field must be mapped to a unique match key, unless the fields are grouped together.
      > 2. If input fields are grouped together, they can share the same match key.
      >
      >   For example, the following schema mapping would be valid for **Advanced** rules:

```
firstName: { matchKey: 'name', groupName: 'name' }

lastName: { matchKey: 'name', groupName: 'name' }
```

In this case, the `firstName` and `lastName` fields are grouped together and share the same name match key, which is allowed.

Review your schema mappings and update them to follow this one-to-one matching rule, unless the fields are properly grouped, in order to use **Advanced** rules.

3. If your data table has a DELETE column, the schema mapping's type must be `String` and you can't have a `matchKey` and `groupName`.

c. The **Normalize data** option is selected by default, so that data inputs are normalized before matching. If you don't want to normalize data, deselect the **Normalize data** option.

> **Note**
>
> Normalization is only supported for the following scenarios in **Create schema mapping**:
>
> - If the following **Name** sub-types are grouped: **First name**, **Middle name**, **Last name**.
> - If the following **Address** sub-types are grouped: **Street address 1**, **Street address 2**, **Street address 3**, **City**, **State**, **Country**, **Postal code**.
> - If the following **Phone** sub-types are grouped: **Phone number**, **Phone country code**.

d. To specify the **Service access** permissions, choose an option and take the recommended action.

Option	Recommended action
Create and use a new service role	• AWS Entity Resolution creates a service role with the required policy for this table. • The default **Service role name** is `entityresolution-matching-workflow-<timestamp>` . • You must have permissions to create roles and attach policies. • If your input data is encrypted, you can choose the **This data is encrypted with a KMS key** option and then enter an **AWS KMS key** that will be used to decrypt your data input.

Option	Recommended action
Use an existing service role	1. Choose an **Existing service role name** from the dropdown list. The list of roles are displayed if you have permissions to list roles. If you don't have permissions to list roles, you can enter the Amazon Resource Name (ARN) of the role that you want to use. If there are no existing service roles, the option to **Use an existing service role** is unavailable. 2. View the service role by choosing the **View in IAM** external link. By default, AWS Entity Resolutio n doesn't attempt to update the existing role policy to add necessary permissions.

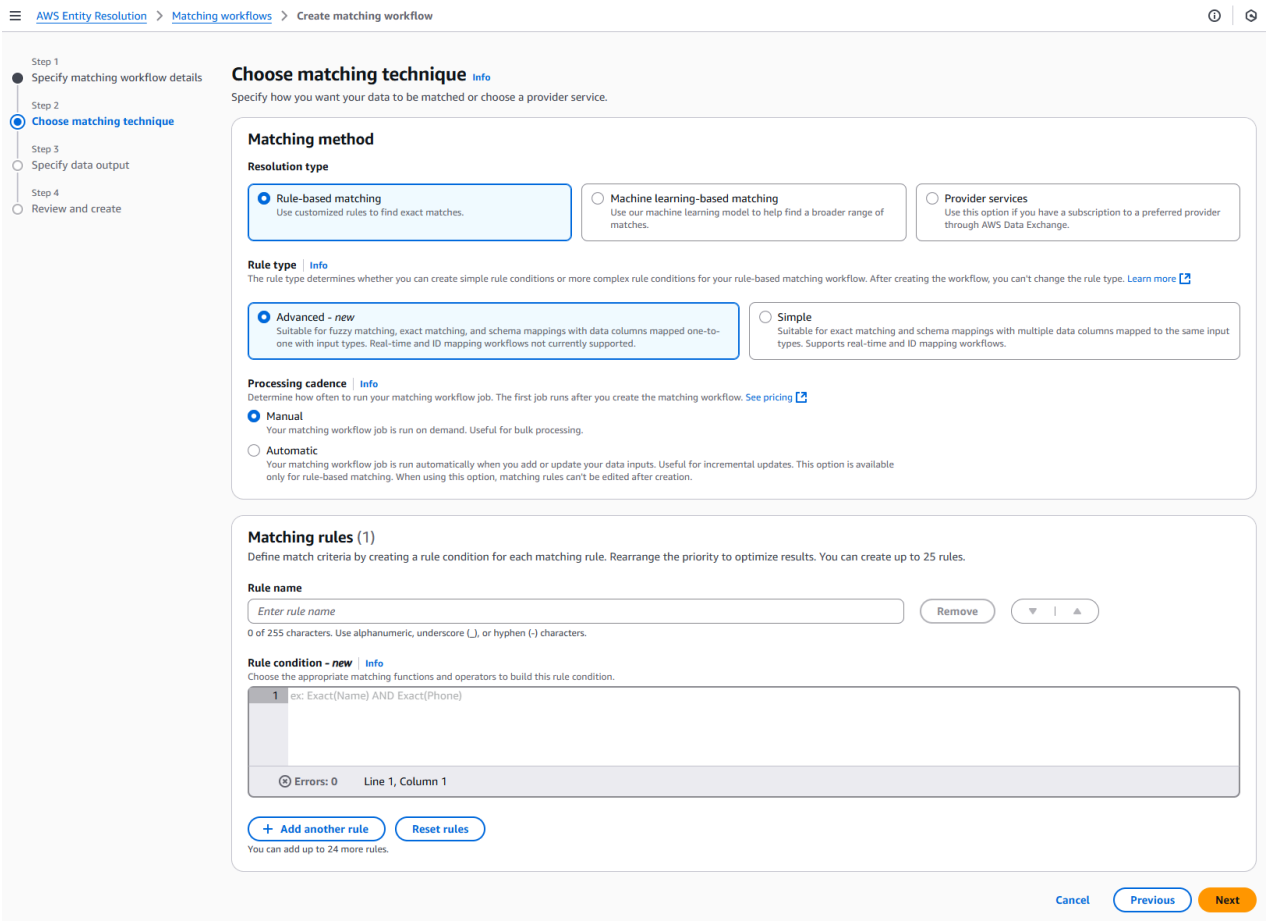
e. (Optional) To enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.

f. Choose **Next**.

5. For **Step 2: Choose matching technique**:

   a. For **Matching method**, choose **Rule-based matching**.

   b. For **Rule type**, choose **Advanced**.



c. For **Processing cadence**, select one of the following options.

- Choose **Manual** to run a workflow on demand for a bulk update
- Choose **Automatic** to run a workflow as soon as new data is in your S3 bucket

> **Note**
>
> If you choose **Automatic**, ensure that you have Amazon EventBridge notifications turned on for your S3 bucket. For instructions on enabling Amazon EventBridge using the S3 console, see Enabling Amazon EventBridge in the *Amazon S3 User Guide*.

d. For **Matching rules**, enter a **Rule name** and then build the **Rule condition** by choosing the appropriate matching functions and operators from the dropdown list based on your goal.

You can create up to 25 rules.

You must combine a fuzzy matching function (**Cosine**, **Levenshtein**, or **Soundex**) with an exact matching function (**Exact**, **ExactManyToMany**) using the **AND** operator.

You can use the following table to help decide what type of function or operator you want to use, depending on your goal.

Your goal	**Recommended function or operator**	**Recommended optional modifier**	**Pros**
Match identical strings on accurate data but don't match on empty values.	**Exact**	**EmptyValues=Process**	
Match identical strings on accurate data and ignore empty values.	**Exact(*matchKey*)**	**EmptyValues=Ignore**	
Match multiple records across match keys. Suitable for flexible pairings. Limit: 15 match keys	**ExactManyToMany**(*matchKey*, *matchKey*, ...)	n/a	

Your goal	Recommended function or operator	Recommended optional modifier	Pros
Measure similarity between numerical representations of data but don't match on empty values. Suitable for text, numbers, or a mix of both.	**Cosine**	**EmptyValues=Process**	Simple, efficient. Works well with long text when combined with TF-IDF weighting. Good for exact word-based matching.
Measure similarity between numerical representations of data and ignore empty values.	**Cosine(*matchKey*, *threshold* , ...)**	**EmptyValues=Ignore**	Handles typos, spelling errors, and transpositions well. Effective on a wide range of PII types. Good for short strings (for example, names or phone numbers).
Count the minimum number of changes needed to change one word into another but don't match on empty values. Suitable for text with slight differences in spelling.	**Levenshtein**	**EmptyValues=Process**	

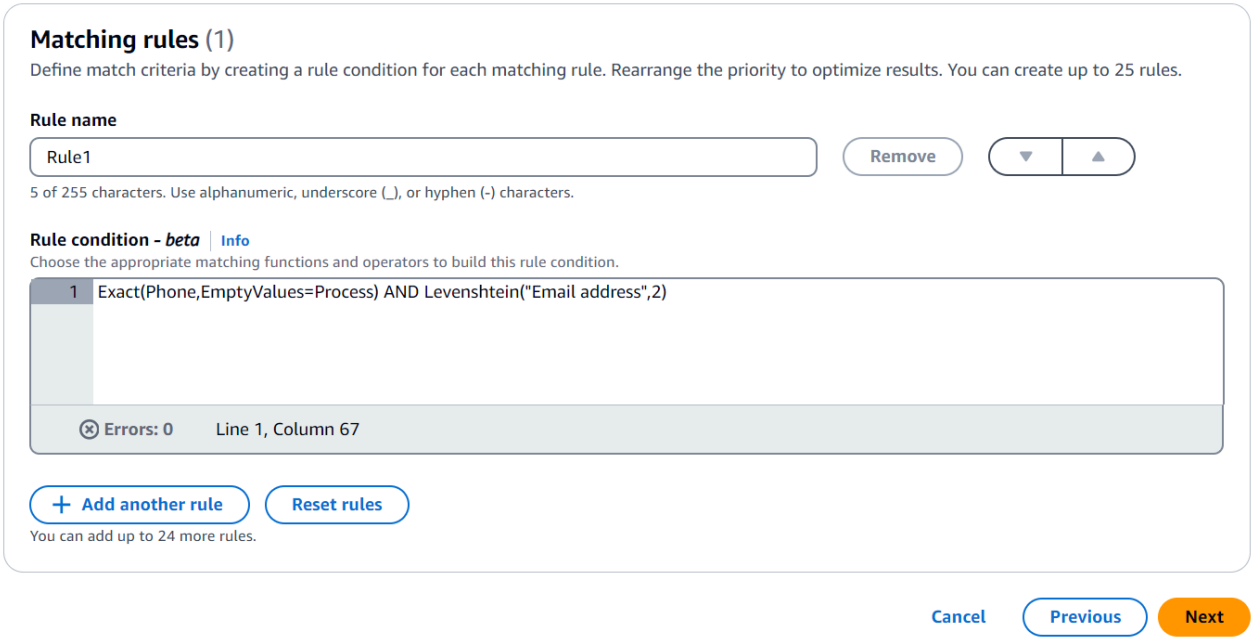
Your goal	Recommended function or operator	Recommended optional modifier	Pros
Count the minimum number of changes needed to change one word into another and ignore empty values.	**Levenshtein(*matchKey*, *threshold* , ...)**	**EmptyValues=Ignore**	
Compare and match text strings based on how similar they sound but don't match on empty values. Suitable for text with variations in spelling or pronunciation.	**Soundex**	**EmptyValues=Process**	Effective for phonetic matching, identifying similar-sounding words. Fast and computationally inexpensive. Good for matching names with similar pronunciations but different spellings.
Compare and match text strings based on how similar they sound and ignore empty values.	**Soundex(*matchKey*)**	**EmptyValues=Ignore**	
Combine functions.	**AND**	n/a	
Separate functions.	**OR**	n/a	

Your goal	Recommended function or operator	Recommended optional modifier	Pros
Group conditions to create nested conditions.	**(...)**	n/a	

Example Rule condition that matches on phone numbers and email

The following is an example of a rule condition that matches records on phone numbers (**Phone** match key) and email addresses (**Email address** match key):

```
Exact(Phone,EmptyValues=Process) AND Levenshtein("Email address",2)
```



The **Phone** match key uses the **Exact** matching function to match identical strings. The **Phone** match key processes empty values in matching using the **EmptyValues=Process** modifier.

The **Email address** match key uses the **Levenshtein** matching function to match data with misspellings using the default Levenshtein Distance algorithm threshold of 2. The **Email** match key doesn't use any optional modifiers.

The **AND** operator combines the **Exact** matching function and the **Levenshtein** matching function.

Example Rule condition that uses ExactManyToMany to perform matchkey matching

The following is an example of a rule condition that matches records on three address fields (**HomeAddress** match key, **BillingAddress** match key, and **ShippingAddress** match key to find potential matches by checking if any if any of them have identical values.

The `ExactManyToMany` operator evaluates all possible combinations of the specified address fields to identify exact matches between any two or more addresses. For example, it would detect if the `HomeAddress` matches either the `BillingAddress` or `ShippingAddress`, or if all three addresses match exactly.

```
ExactManyToMany(HomeAddress, BillingAddress, ShippingAddress)
```

Example Rule condition that uses clustering

In Advanced Rule Based Matching with fuzzy conditions, the system first groups records into clusters based on exact matches. Once these initial clusters are formed, the system applies fuzzy matching filters to identify additional matches within each cluster. For optimal performance, you should select exact match conditions based on your data patterns to create well-defined initial clusters.

The following is an example of a rule condition that combines multiple exact matches with a fuzzy match requirement. It uses `AND` operators to check that three fields — `FullName`, Date of Birth (`DOB`), and `Address` — match exactly between records. It also allows for minor variations in the `InternalID` field using a Levenshtein distance of `1`. The Levenshtein distance measures the minimum number of single-character edits required to change one string into another. A distance of 1 means it will match `InternalIDs` that differ by only one character (like a single typo, deletion, or insertion). This combination of conditions helps identify records that are very likely to represent the same entity, even if there are small discrepancies in the identifier.

```
Exact(FullName) AND Exact(DOB) AND Exact(Address) and
 Levenshtein(InternalID, 1)
```

e. Choose **Next**.

6. For **Step 3: Specify data output and format**:

   a. For **Data output destination and format**, choose the **Amazon S3 location** for the data output and whether the **Data format** will be **Normalized data** or **Original data**.

   b. For **Encryption**, if you choose to **Customize encryption settings**, enter the **AWS KMS key** ARN.

   c. View the **System generated output**.

   d. For **Data output**, decide which fields you want to include, hide, or mask, and then take the recommended actions based on your goals.

Your goal	Recommended action
Include fields	Keep the output state as **Included**.
Hide fields (exclude from output)	Choose the **Output field**, and then choose **Hide**.
Mask fields	Choose the **Output field**, and then choose **Hash output**.
Reset the previous settings	Choose **Reset**.

   e. Choose **Next**.

7. For **Step 4: Review and create**:

   a. Review the selections that you made for the previous steps and edit if necessary.

   b. Choose **Create and run**.

   A message appears, indicating that the matching workflow has been created and that the job has started.

8. On the matching workflow details page, on the **Metrics** tab, view the following under **Last job metrics**:

- The **Job ID**.
- The **Status** of the matching workflow job: **Queued**, **In progress**, **Completed**, **Failed**
- The **Time completed** for the workflow job.
- The number of **Records processed**.
- The number of **Records not processed**.
- The **Unique match IDs generated**.
- The number of **Input records**.

You can also view the job metrics for matching workflow jobs that have been previously run under the **Job history**.

9. After the matching workflow job completes (**Status** is **Completed**), you can go to the **Data output** tab and then select your **Amazon S3 location** to view the results.
10. (**Manual** processing type only) If you have created a **Rule-based matching** workflow with the **Manual** processing type, you can run the matching workflow anytime by choosing **Run workflow** on the matching workflow details page.
11. (**Automatic** processing type only) If your data table has a DELETE column, then:

    - Records set to *`true`* in the DELETE column are deleted.
    - Records set to *`false`* in the DELETE column are ingested into S3.

    For more information, see Step 1: Prepare first-party data tables.

API

To create a rule-based matching workflow with the Advanced rule type using the API

> **Note**
>
> By default, the workflow uses standard (batch) processing. To use incremental (automatic processing, you must explicitly configure it.

1. Open a terminal or command prompt to make the API request.
2. Create a POST request to the following endpoint:

```
/matchingworkflows
```

3. In the request header, set the Content-type to application/json.

> **Note**
>
> For a complete list of supported programming languages, see the *AWS Entity Resolution API Reference*.

4. For the request body, provide the following required JSON parameters:

```
{
   "description": "string",
   "incrementalRunConfig": {
      "incrementalRunType": "string"
   },
   "inputSourceConfig": [
      {
         "applyNormalization": boolean,
         "inputSourceARN": "string",
         "schemaName": "string"
      }
   ],
   "outputSourceConfig": [
      {
         "applyNormalization": boolean,
         "KMSArn": "string",
         "output": [
            {
               "hashed": boolean,
               "name": "string"
            }
         ],
         "outputS3Path": "string"
      }
   ],
   "resolutionTechniques": {
      "providerProperties": {
         "intermediateSourceConfiguration": {
            "intermediateS3Path": "string"
         },
         "providerConfiguration": JSON value,
```

```
            "providerServiceArn": "string"
         },
         "resolutionType": "RULE_MATCHING",
         "ruleBasedProperties": {
            "attributeMatchingModel": "string",
            "matchPurpose": "string",
            "rules": [
               {
                  "matchingKeys": [ "string" ],
                  "ruleName": "string"
               }
            ]
         },
         "ruleConditionProperties": {
            "rules": [
               {
                  "condition": "string",
                  "ruleName": "string"
               }
            ]
         }
      },
      "roleArn": "string",
      "tags": {
         "string" : "string"
      },
      "workflowName": "string"
}
```

Where:

- `workflowName` (required) – Must be unique and between 1–255 characters matching pattern [a-zA-Z_0-9-]*
- `inputSourceConfig` (required) – List of 1–20 input source configurations
- `outputSourceConfig` (required) – Exactly one output source configuration
- `resolutionTechniques` (required) – Set to "RULE_MATCHING" as the resolutionType for rule-based matching
- `roleArn` (required) – IAM role ARN for workflow execution
- `ruleConditionProperties` (required) – List of rule conditions and the name of the matching rule.

Optional parameters include:

- `description` – Up to 255 characters
- `incrementalRunConfig` – Incremental run type configuration
- `tags` – Up to 200 key-value pairs

5. (Optional) To use incremental processing instead of the default standard (batch) processing, add the following parameter to the request body:

```
"incrementalRunConfig": {
   "incrementalRunType": "AUTOMATIC"
}
```

6. Send the request.
7. If successful, you'll receive a response with status code 200 and a JSON body containing:

```
{
   "workflowArn": "string",
   "workflowName": "string",
   // Plus all configured workflow details
}
```

8. If the call is unsuccessful, you might receive one of these errors:

- 400 – ConflictException if the workflow name already exists
- 400 – ValidationException if the input fails validation
- 402 – ExceedsLimitException if account limits are exceeded
- 403 – AccessDeniedException if you don't have sufficient access
- 429 – ThrottlingException if the request was throttled
- 500 – InternalServerException if there's an internal service failure

Creating a rule-based matching workflow with the Simple rule type

The following procedure demonstrates how to create a rule-based matching workflow with the **Simple** rule type using either the AWS Entity Resolution Console or the `CreateMatchingWorkflow` API.

Console

To create a rule-based matching workflow with the Simple rule type using the console

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **Matching**.
3. On the **Matching workflows** page, in the upper right corner, choose **Create matching workflow**.
4. For **Step 1: Specify matching workflow details**, do the following:

   a. Enter a **Matching workflow name** and an optional **Description**.

   b. For **Data input**, choose an **AWS Glue database** from the dropdown, select the **AWS Glue table**, and then the corresponding **Schema mapping**.

   You can add up to 19 data inputs.

   c. The **Normalize data** option is selected by default, so that data inputs are normalized before matching. If you don't want to normalize data, deselect the **Normalize data** option.

   > **Note**
   >
   > Normalization is only supported for the following scenarios in **Create schema mapping**:
   >
   > - If the following **Name** sub-types are grouped: **First name**, **Middle name**, **Last name**.
   > - If the following **Address** sub-types are grouped: **Street address 1**, **Street address 2**, **Street address 3**, **City**, **State**, **Country**, **Postal code**.
   > - If the following **Phone** sub-types are grouped: **Phone number**, **Phone country code**.

   d. To specify the **Service access** permissions, choose an option and take the recommended action.

Option	Recommended action
Create and use a new service role	• AWS Entity Resolution creates a service role with the required policy for this table. • The default **Service role name** is `entityresolution-matching-workflow-<timestamp>` . • You must have permissions to create roles and attach policies. • If your input data is encrypted, you can choose the **This data is encrypted with a KMS key** option and then enter an **AWS KMS key** that will be used to decrypt your data input.

Option	Recommended action
Use an existing service role	1. Choose an **Existing service role name** from the dropdown list. The list of roles are displayed if you have permissions to list roles. If you don't have permissions to list roles, you can enter the Amazon Resource Name (ARN) of the role that you want to use. If there are no existing service roles, the option to **Use an existing service role** is unavailable. 2. View the service role by choosing the **View in IAM** external link. By default, AWS Entity Resolutio n doesn't attempt to update the existing role policy to add necessary permissions.

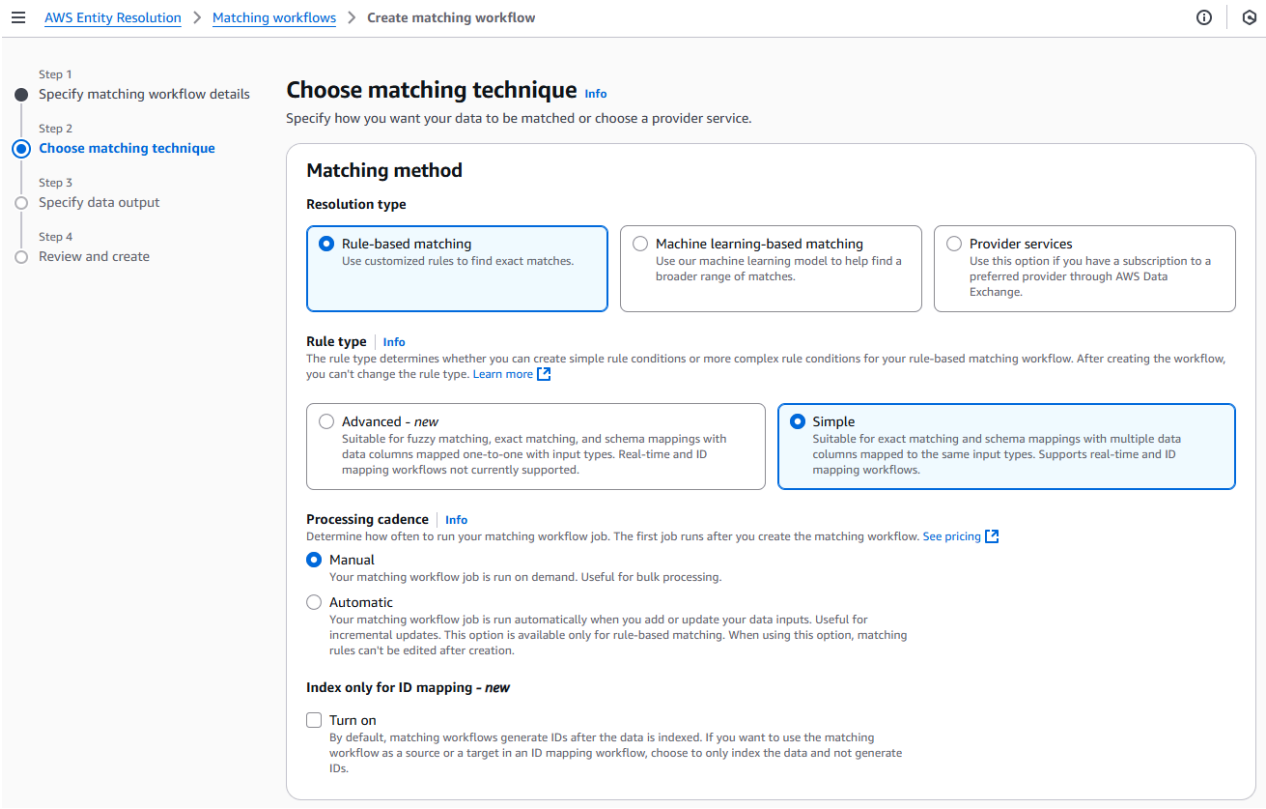
e. (Optional) To enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.

f. Choose **Next**.

5. For **Step 2: Choose matching technique**:

   a. For **Matching method**, choose **Rule-based matching**.

   b. For **Rule type**, choose **Simple**.



c. For **Processing cadence**, select one of the following options.

- Choose **Manual** to run a workflow on demand for a bulk update
- Choose **Automatic** to run a workflow as soon as new data is in your S3 bucket

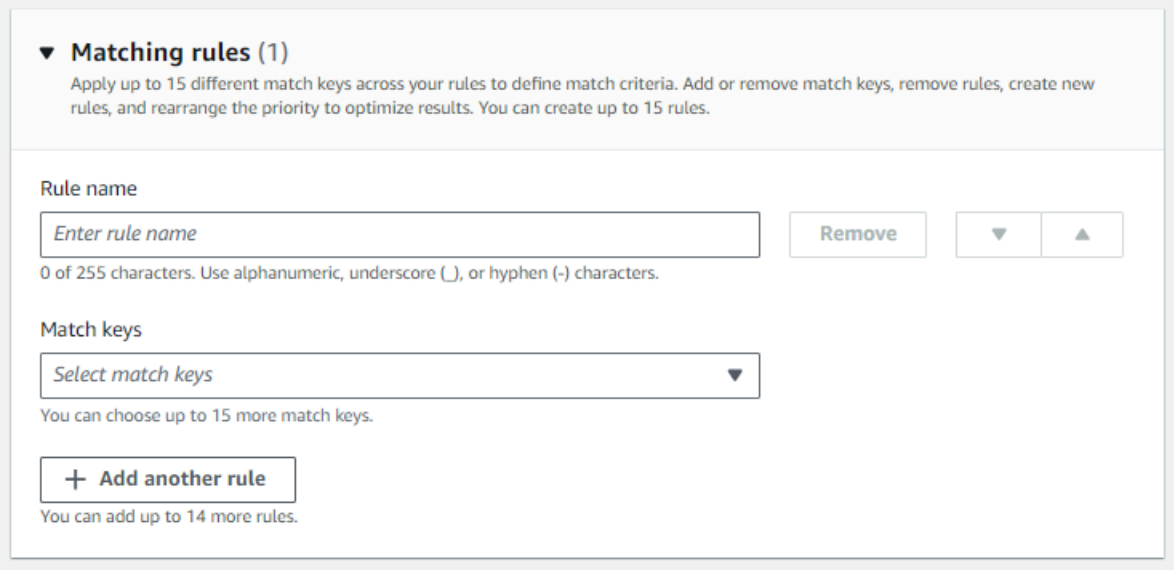
> **Note**
>
> If you choose **Automatic**, ensure that you have Amazon EventBridge notifications turned on for your S3 bucket. For instructions on enabling Amazon EventBridge using the S3 console, see Enabling Amazon EventBridge in the *Amazon S3 User Guide*.

d. (Optional) For **Index only for ID mapping**, You can choose to **Turn on** the ability to only index the data and not generate IDs.

   By default, matching workflow generate IDs after the data is indexed.

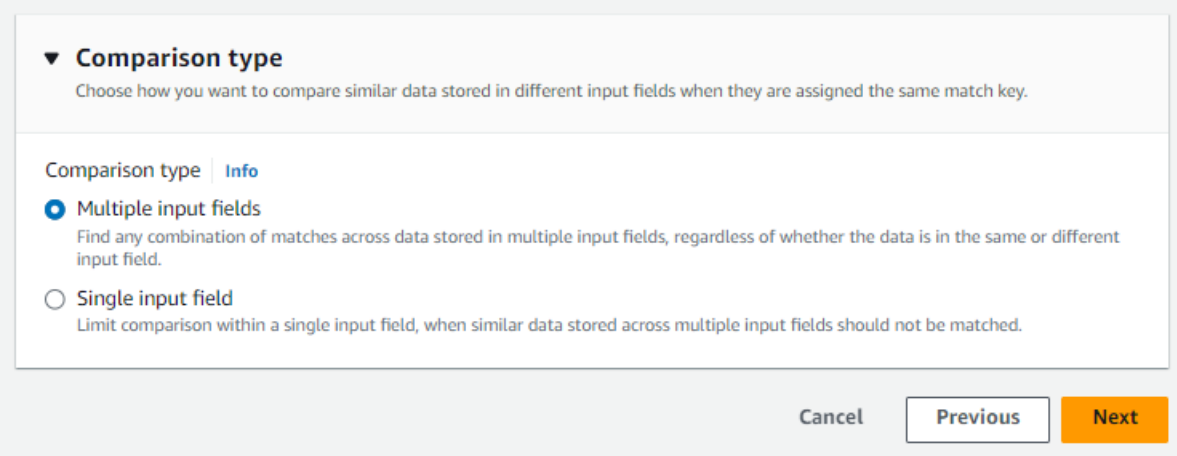
e. For **Matching rules**, enter a **Rule name** and then choose the **Match keys** for that rule.

You can create up to 15 rules and you can apply up to 15 different match keys across your rules to define match criteria.



f. For **Comparison type**, choose one of the following options based on your goal.

Your goal	Recommended option
Find any combination of matches across data stored in multiple input fields	**Multiple input fields**
Limit comparison to a single input field	**Single input field**



g. Choose **Next**.

6. For **Step 3: Specify data output and format**:

   a. For **Data output destination and format**, choose the **Amazon S3 location** for the data output and whether the **Data format** will be **Normalized data** or **Original data**.

   b. For **Encryption**, if you choose to **Customize encryption settings**, enter the **AWS KMS key** ARN.

   c. View the **System generated output**.

   d. For **Data output**, decide which fields you want to include, hide, or mask, and then take the recommended actions based on your goals.

Your goal	Recommended action
Include fields	Keep the output state as **Included**.
Hide fields (exclude from output)	Choose the **Output field**, and then choose **Hide**.
Mask fields	Choose the **Output field**, and then choose **Hash output**.
Reset the previous settings	Choose **Reset**.

   e. Choose **Next**.

7. For **Step 4: Review and create**:

   a. Review the selections that you made for the previous steps and edit if necessary.

   b. Choose **Create and run**.

   A message appears, indicating that the matching workflow has been created and that the job has started.

8. On the matching workflow details page, on the **Metrics** tab, view the following under **Last job metrics**:

   - The **Job ID**.
   - The **Status** of the matching workflow job: **Queued**, **In progress**, **Completed**, **Failed**
   - The **Time completed** for the workflow job.
   - The number of **Records processed**.

- The number of **Records not processed**.
- The **Unique match IDs generated**.
- The number of **Input records**.

You can also view the job metrics for matching workflow jobs that have been previously run under the **Job history**.

9. After the matching workflow job completes (**Status** is **Completed**), you can go to the **Data output** tab and then select your **Amazon S3 location** to view the results.
10. (**Manual** processing type only) If you have created a **Rule-based matching** workflow with the **Manual** processing type, you can run the matching workflow anytime by choosing **Run workflow** on the matching workflow details page.

API

To create a rule-based matching workflow with the Simple rule type using the API

> **Note**
>
> By default, the workflow uses standard (batch) processing. To use incremental (automatic processing, you must explicitly configure it.

1. Open a terminal or command prompt to make the API request.
2. Create a POST request to the following endpoint:

```
/matchingworkflows
```

3. In the request header, set the Content-type to application/json.

> **Note**
>
> For a complete list of supported programming languages, see the *AWS Entity Resolution API Reference*.

4. For the request body, provide the following required JSON parameters:

```
{
```

```
    "description": "string",
    "incrementalRunConfig": {
       "incrementalRunType": "string"
    },
    "inputSourceConfig": [
       {
          "applyNormalization": boolean,
          "inputSourceARN": "string",
          "schemaName": "string"
       }
    ],
    "outputSourceConfig": [
       {
          "applyNormalization": boolean,
          "KMSArn": "string",
          "output": [
             {
                "hashed": boolean,
                "name": "string"
             }
          ],
          "outputS3Path": "string"
       }
    ],
    "resolutionTechniques": {
       "providerProperties": {
          "intermediateSourceConfiguration": {
             "intermediateS3Path": "string"
          },
          "providerConfiguration": JSON value,
          "providerServiceArn": "string"
       },
       "resolutionType": "RULE_MATCHING",
       "ruleBasedProperties": {
          "attributeMatchingModel": "string",
          "matchPurpose": "string",
          "rules": [
             {
                "matchingKeys": [ "string" ],
                "ruleName": "string"
             }
          ]
       },
       "ruleConditionProperties": {
```

```
         "rules": [
            {
               "condition": "string",
               "ruleName": "string"
            }
         ]
      }
   },
   "roleArn": "string",
   "tags": {
      "string" : "string"
   },
   "workflowName": "string"
}
```

Where:

- `workflowName` (required) – Must be unique and between 1–255 characters matching pattern [a-zA-Z_0-9-]*
- `inputSourceConfig` (required) – List of 1–20 input source configurations
- `outputSourceConfig` (required) – Exactly one output source configuration
- `resolutionTechniques` (required) – Set to "RULE_MATCHING" for rule-based matching
- `roleArn` (required) – IAM role ARN for workflow execution
- `ruleConditionProperties` (required) – List of rule conditions and the name of the matching rule.

Optional parameters include:

- `description` – Up to 255 characters
- `incrementalRunConfig` – Incremental run type configuration
- `tags` – Up to 200 key-value pairs

5. (Optional) To use incremental processing instead of the default standard (batch) processing, add the following parameter to the request body:

```
"incrementalRunConfig": {
   "incrementalRunType": "AUTOMATIC"
```

```
}
```

6. Send the request.
7. If successful, you'll receive a response with status code 200 and a JSON body containing:

```
{
   "workflowArn": "string",
   "workflowName": "string",
   // Plus all configured workflow details
}
```

8. If the call is unsuccessful, you might receive one of these errors:
   - 400 – ConflictException if the workflow name already exists
   - 400 – ValidationException if the input fails validation
   - 402 – ExceedsLimitException if account limits are exceeded
   - 403 – AccessDeniedException if you don't have sufficient access
   - 429 – ThrottlingException if the request was throttled
   - 500 – InternalServerException if there's an internal service failure

Creating a machine learning-based matching workflow

Machine learning-based matching is a preset process that attempts to match records across all of the data that you input. The machine learning-based matching workflow enables you to compare cleartext data to find a broad range of matches using a machine learning model.

> **Note**
>
> The machine learning model doesn't support the comparison of hashed data.

When AWS Entity Resolution finds a match between two or more records in your data, it assigns:

- A Match ID to the records in the matched set of data
- The match confidence level percentage.

You can use the output of an ML-based matching workflow as an input for data service provider matching, or vice-versa to meet your specific goals. For example, you can run an ML-based matching to find matches across your data sources on your own records first. If a subset wasn't matched, you can then run provider service- based matching to find additional matches.

To create a ML-based matching workflow:

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **Matching**.
3. On the **Matching workflows** page, in the upper right corner, choose **Create matching workflow**.
4. For **Step 1: Specify matching workflow details**, do the following:

   a. Enter a **Matching workflow name** and an optional **Description**.

   b. For **Data input**, choose an **AWS Glue database** from the dropdown, select the **AWS Glue table**, and then the corresponding **Schema mapping**.

   You can add up to 20 data inputs.

   c. The **Normalize data** option is selected by default, so that data inputs are normalized before matching. If you don't want to normalize data, deselect the **Normalize data** option.

   Machine learning based-matching only normalizes Name, Phone, and Email.

   d. To specify the **Service access** permissions, choose an option and take the recommended action.

Option	Recommended action
Create and use a new service role	• AWS Entity Resolution creates a service role with the required policy for this table. • The default **Service role name** is `entityresolution-matching-workflow-<timestamp>` . • You must have permissions to create roles and attach policies.

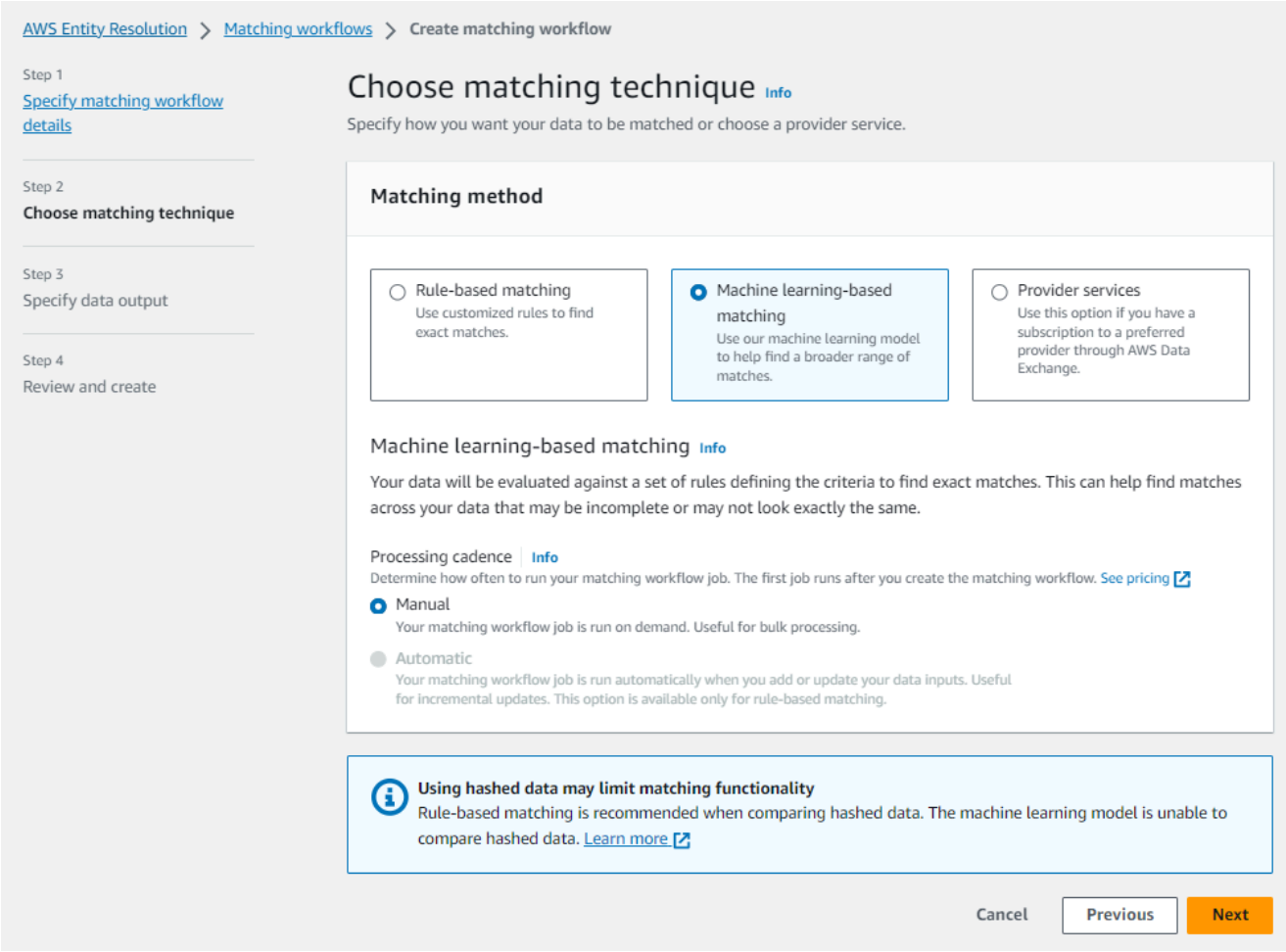
Option	Recommended action
	• If your input data is encrypted, choose the **This data is encrypted by a KMS key** option. Then, enter an **AWS KMS key** that is used to decrypt your data input.
Use an existing service role	1. Choose an **Existing service role name** from the dropdown list. The list of roles are displayed if you have permissions to list roles. If you don't have permissions to list roles, you can enter the Amazon Resource Name (ARN) of the role that you want to use. If there are no existing service roles, the option to **Use an existing service role** is unavailable. 2. View the service role by choosing the **View in IAM** external link. By default, AWS Entity Resolution doesn't attempt to update the existing role policy to add necessary permissio ns.

e. (Optional) To enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.

f. Choose **Next**.

5. For **Step 2: Choose matching technique**:

   a. For **Matching method**, choose **Machine learning-based matching**.



b. For **Processing cadence**, the **Manual** option is selected.

This option enables you to run a workflow on demand for a bulk update.

> **Note**
>
> Automatic (incremental) processing is not supported for machine learning-based matching workflows.

c. Choose **Next**.

6. For **Step 3: Specify data output and format**:

a. For **Data output destination and format**, choose the **Amazon S3 location** for the data output and whether the **Data format** will be **Normalized data** or **Original data**.

b. For **Encryption**, if you choose to **Customize encryption settings**, enter the **AWS KMS key** ARN.

c. View the **System generated output**.

d. For **Data output**, decide which fields you want to include, hide, or mask, and then take the recommended actions based on your goals.

Your goal	Recommended option
Include fields	Keep the output state as **Included**.
Hide fields (exclude from output)	Choose the **Output field**, and then choose **Hide**.
Mask fields	Choose the **Output field**, and then choose **Hash output**.
Reset the previous settings	Choose **Reset**.

e. Choose **Next**.

7. For **Step 4: Review and create**:

a. Review the selections that you made for the previous steps and edit if necessary.

b. Choose **Create and run**.

A message appears, indicating that the matching workflow has been created and that the job has started.

8. On the matching workflow details page, on the **Metrics** tab, view the following under **Last job metrics**:

- The **Job ID**.
- The **Status** of the matching workflow job: **Queued**, **In progress**, **Completed**, **Failed**
- The **Time completed** for the workflow job.
- The number of **Records processed**.
- The number of **Records not processed**.
- The **Unique match IDs generated**.
- The number of **Input records**.

You can also view the job metrics for matching workflow jobs that have been previously run under the **Job history**.

9. After the matching workflow job completes (**Status** is **Completed**), you can go to the **Data output** tab and then select your **Amazon S3 location** to view the results.
10. (**Manual** processing type only) If you have created a **Machine learning-based matching** workflow with the **Manual** processing type, you can run the matching workflow anytime by choosing **Run workflow** on the matching workflow details page.

Creating a provider service-based matching workflow

Provider service-based matching enables you to match your known identifiers with your preferred data service provider.

AWS Entity Resolution currently supports the following data provider services:

- LiveRamp
- TransUnion
- Unified ID 2.0

For more information about the supported provider services, see Preparing third-party input data.

You can use a public subscription for these providers on AWS Data Exchange or negotiate a private offer directly with the data provider. For more information about creating a new subscription or reusing an existing subscription to a provider service, see Step 1: Subscribe to a provider service on AWS Data Exchange.

The following sections describe how to create a provider-based matching workflow.

Topics

- Creating a matching workflow with LiveRamp
- Creating a matching workflow with TransUnion
- Creating a matching workflow with UID 2.0

Creating a matching workflow with LiveRamp

If you have a subscription to the LiveRamp service, you can create a matching workflow with the LiveRamp service to perform identity resolution.

The LiveRamp service provides an identifier called the RampID. The RampID is one of the most commonly used IDs in demand-side platforms to create an audience for an advertising campaign. Using a matching workflow with LiveRamp, you can resolve hashed email addresses to RAMPIDs.

> **Note**
>
> AWS Entity Resolution supports PII-based RampID assignment.

This workflow requires an Amazon S3 data staging bucket where you want the matching workflow output to be temporarily written. Before you create a ID mapping workflow with LiveRamp, add the following permissions to the data staging bucket.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::715724997226:root"

            },
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::<staging-bucket>",
                "arn:aws:s3:::<staging-bucket>/*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::715724997226:root"
            },
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicy",
                "s3:ListBucketVersions",
                "s3:GetBucketAcl"
            ],
            "Resource": [
                "arn:aws:s3:::<staging-bucket>",
                "arn:aws:s3:::<staging-bucket>/*"
            ]
        }
    ]
}
```

Replace each *<user input placeholder>* with your own information.

staging-bucket	Amazon S3 bucket that temporarily stores your data while running a provider service-based workflow.

To create a matching workflow with LiveRamp:

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **Matching**.
3. On the **Matching workflows** page, in the upper right corner, choose **Create matching workflow**.
4. For **Step 1: Specify matching workflow details**, do the following:
    a. Enter a **Matching workflow name** and an optional **Description**.
    b. For **Data input**, choose an **AWS Glue database** from the dropdown, select the **AWS Glue table**, and then select the corresponding **Schema mapping**.

You can add up to 20 data inputs.

c. The **Normalize data** option is selected by default, so that data inputs are normalized before matching.

> **Note**
>
> Normalization is only supported for the following scenarios in **Create schema mapping**:
>
> - If the following **Name** sub-types are grouped: **First name**, **Middle name**, **Last name**.
> - If the following **Address** sub-types are grouped: **Street address 1**, **Street address 2**: **Street address 3 name**, **City name**, **State**, **Country**, **Postal code**.
> - If the following **Phone** sub-types are grouped: **Phone number**, **Phone country code**.

If you are using the email-only resolution process, deselect the **Normalize data** option, because only hashed emails are used for input data.

d. To specify the **Service access** permissions, choose an option and take the recommended action.

Option	Recommended action
Create and use a new service role	• AWS Entity Resolution creates a service role with the required policy for this table. • The default **Service role name** is `entityresolution-matching-workflow-<timestamp>` . • You must have permissions to create roles and attach policies. • If your input data is encrypted, choose the **This data is encrypted by a KMS key** option. Then, enter an **AWS KMS key** that is used to decrypt your data input.

Option	Recommended action
Use an existing service role	1. Choose an **Existing service role name** from the dropdown list. The list of roles are displayed if you have permissions to list roles. If you don't have permissions to list roles, you can enter the Amazon Resource Name (ARN) of the role that you want to use. If there are no existing service roles, the option to **Use an existing service role** is unavailable. 2. View the service role by choosing the **View in IAM** external link. By default, AWS Entity Resolution doesn't attempt to update the existing role policy to add necessary permissio ns.

e. (Optional) To enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.

f. Choose **Next**.

5. For **Step 2: Choose matching technique**:

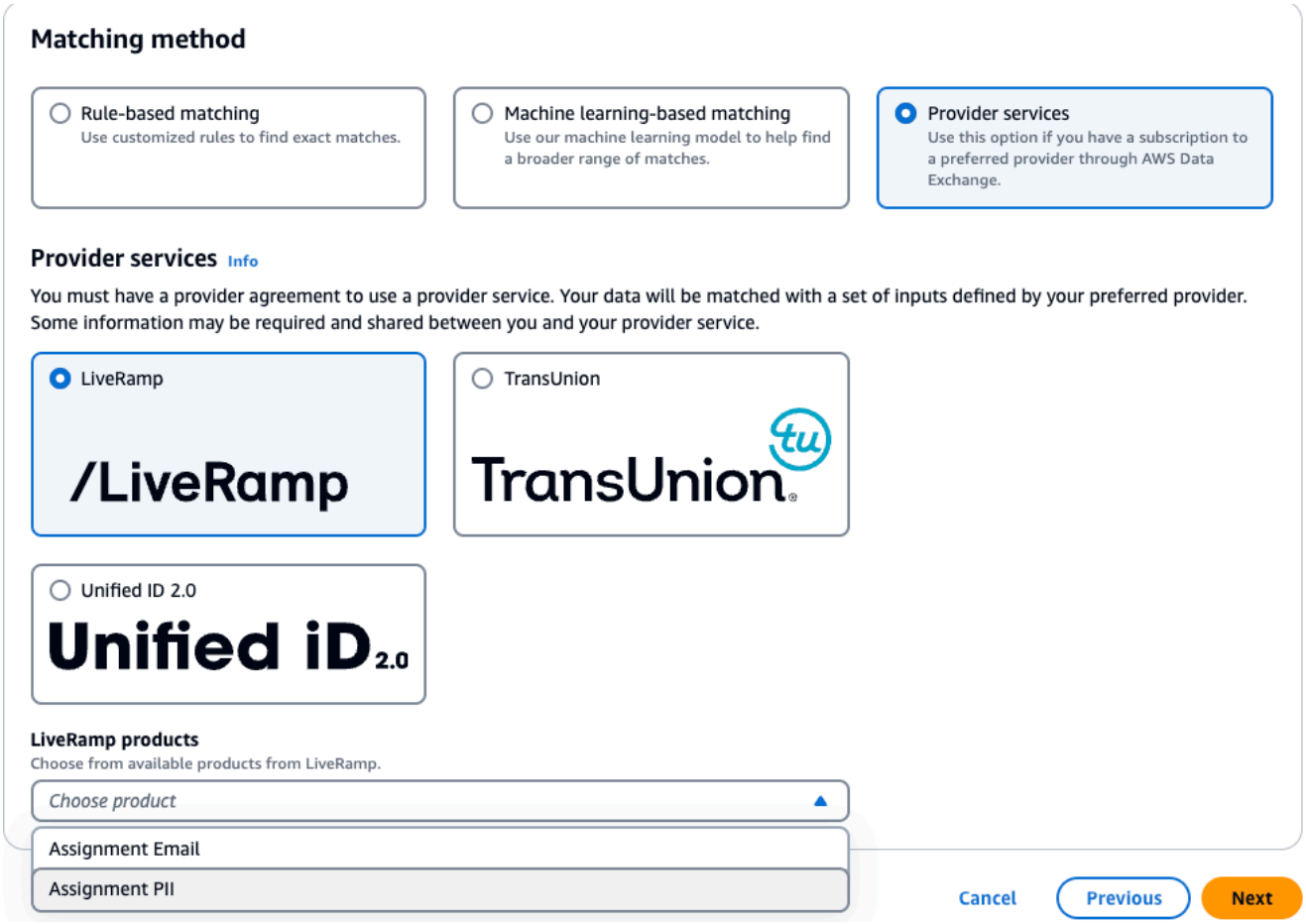
a. For **Matching method**, choose **Provider services**.

b. For **Provider services**, choose **LiveRamp**.

> **Note**
>
> Ensure that your data input file format and normalization is aligned with the provider service's guidelines.

For more information about input file formatting guidelines for the matching workflow, see Perform Identity Resolution Through ADX in the LiveRamp documentation.

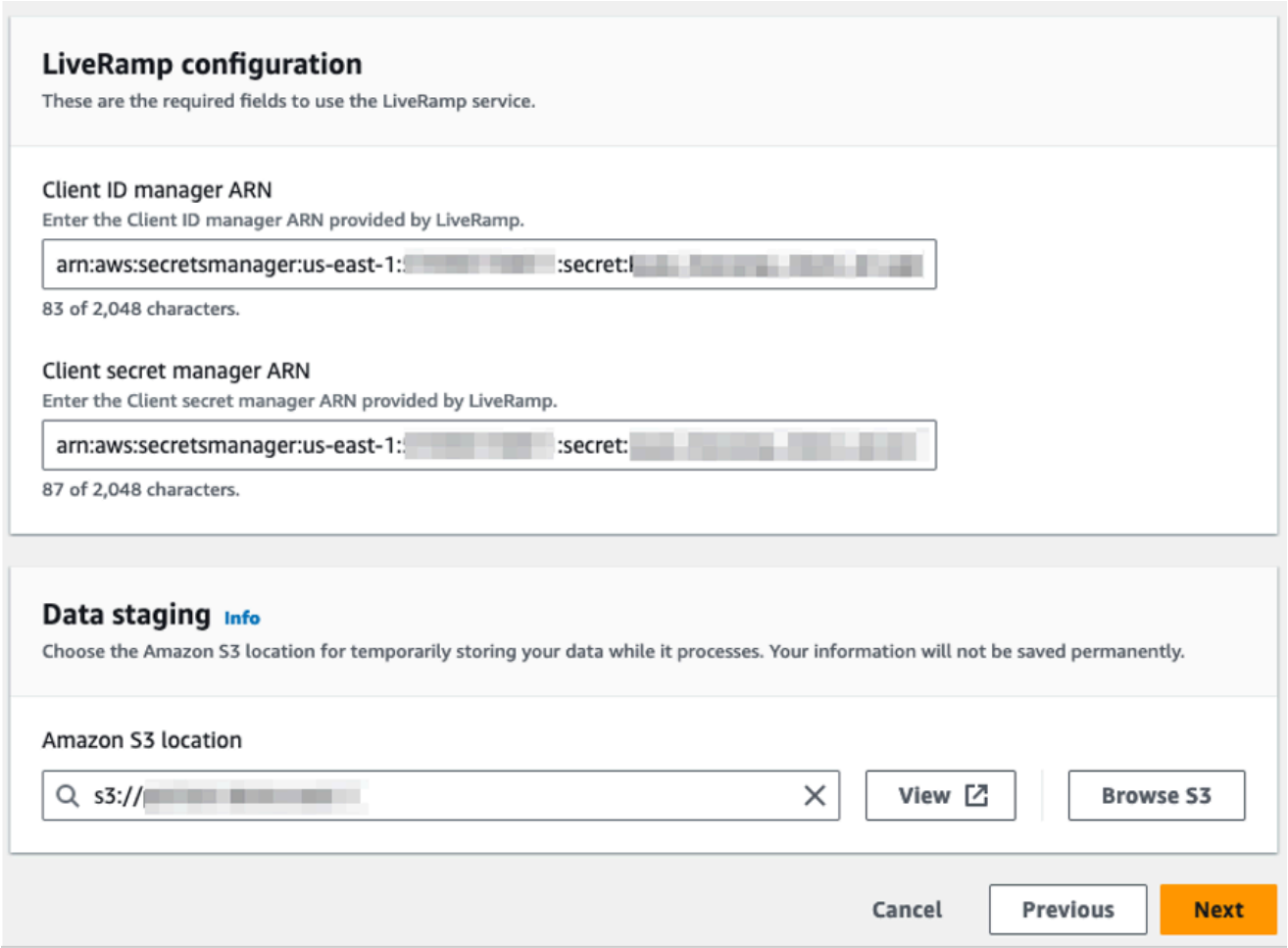
c. For **LiveRamp products**, choose a product from the dropdown list.



Note

If you choose **Assignment PII,** then you must provide at least one non-identifier column when performing entity resolution. For example, GENDER.

d. For **LiveRamp configuration**, enter a **Client ID manager ARN** and a **Client secret manager ARN**.



e. For **Data staging**, choose the **Amazon S3 location** for the temporary storage of your data while it processes.

   You must have permission to the data staging **Amazon S3 location**. For more information, see Creating a workflow job role for AWS Entity Resolution.

f. Choose **Next**.

6. For **Step 3: Specify data output**:

   a. For **Data output destination and format**, choose the **Amazon S3 location** for the data output and whether the **Data format** will be **Normalized data** or **Original data**.

   b. For **Encryption**, if you choose to **Customize encryption settings**, enter the **AWS KMS key** ARN.

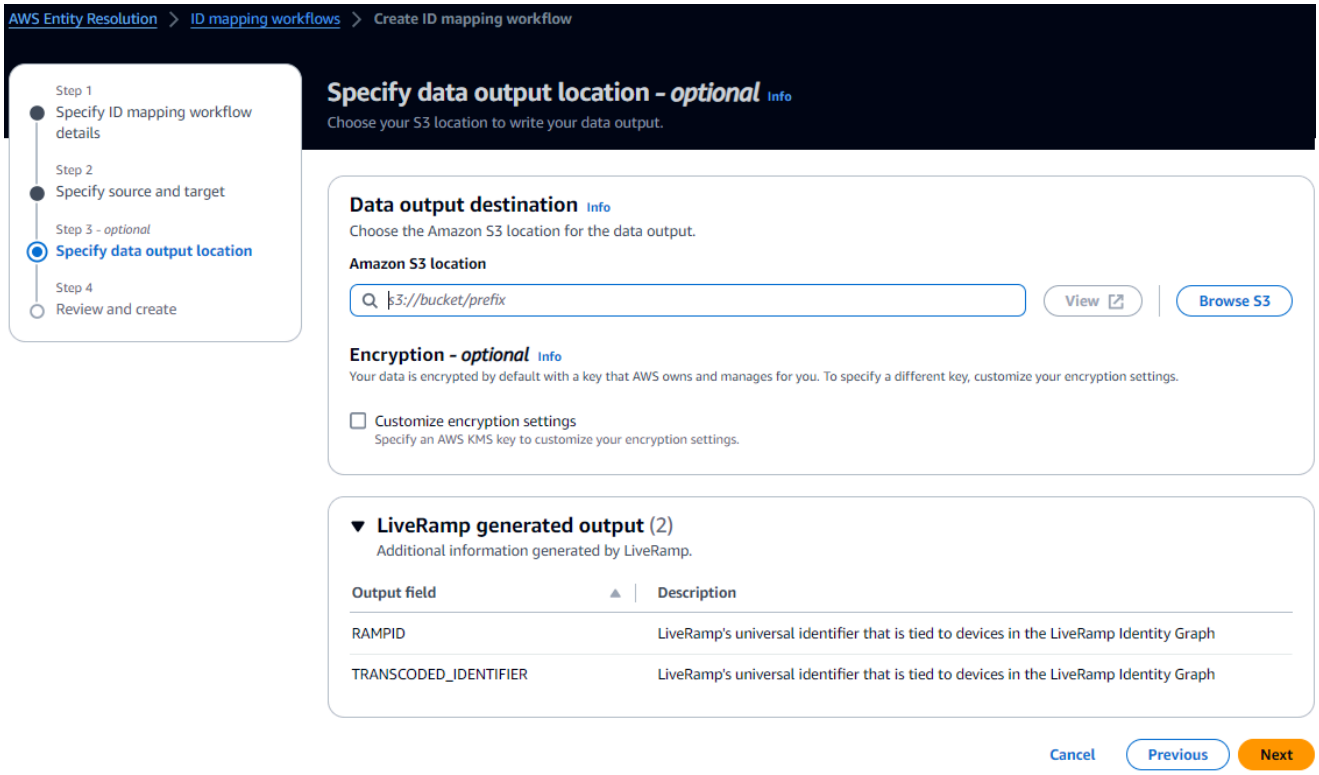
   c. View the **LiveRamp generated output**.

      This is the additional information generated by LiveRamp.

d. For **Data output**, decide which fields you want to include, hide, or mask, and then take the recommended actions based on your goals.

> **Note**
>
> If you have chosen **LiveRamp**, due to LiveRamp privacy filters that remove Personally Identifiable Information (PII), some fields will display an **Output** state of **Unavailable**.

Your goal	Recommended option
Include fields	Keep the output state as **Included**.
Hide fields (exclude from output)	Choose the **Output field**, and then choose **Hide**.
Mask fields	Choose the **Output field**, and then choose **Hash output**.
Reset the previous settings	Choose **Reset**.



e. Choose **Next**.

7. For **Step 4: Review and create**:

   a. Review the selections that you made for the previous steps and edit if necessary.

   b. Choose **Create and run**.

   A message appears, indicating that the matching workflow has been created and that the job has started.

8. On the matching workflow details page, on the **Metrics** tab, view the following under **Last job metrics**:

   - The **Job ID**.
   - The **Status** of the matching workflow job: **Queued**, **In progress**, **Completed**, **Failed**
   - The **Time completed** for the workflow job.
   - The number of **Records processed**.
   - The number of **Records not processed**.
   - The **Unique match IDs generated**.
   - The number of **Input records**.

You can also view the job metrics for matching workflow jobs that have been previously run under the **Job history**.

9. After the matching workflow job completes (**Status** is **Completed**), you can go to the **Data output** tab and then select your **Amazon S3 location** to view the results.

Creating a matching workflow with TransUnion

If you have a subscription to the TransUnion service, you can improve customer understanding by linking, matching, and enhancing customer-related records stored across disparate channels with TransUnion Person and Household E Keys and over 200 data attributes.

The TransUnion service provides identifiers known as the TransUnion Individual and Household IDs. TransUnion provides ID assignment (also known as encoding) of known identifiers such as name, address, phone number, and email address.

This workflow requires an Amazon S3 data staging bucket where you want the matching workflow output to be temporarily written. Before you create a matching workflow with TransUnion, add the following permissions to the data staging bucket.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::381491956555:root"

            },
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::<staging-bucket>",
```

```
                "arn:aws:s3:::<staging-bucket>/*"
            ]
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::381491956555:root"
            },
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicy",
                "s3:ListBucketVersions",
                "s3:GetBucketAcl"
            ],
            "Resource": [
                "arn:aws:s3:::<staging-bucket>",
                "arn:aws:s3:::<staging-bucket>/*"
            ]
        }
    ]
}
```

Replace each *<user input placeholder>* with your own information.

staging-bucket	Amazon S3 bucket that temporarily stores your data while running a provider service-based workflow.

To create a matching workflow with TransUnion:

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **Matching**.
3. On the **Matching workflows** page, in the upper right corner, choose **Create matching workflow**.
4. For **Step 1: Specify matching workflow details**, do the following:

    a. Enter a **Matching workflow name** and an optional **Description**.

b. For **Data input**, choose an **AWS Glue database** from the dropdown, select the **AWS Glue table**, and then select the corresponding **Schema mapping**.

   You can add up to 20 data inputs.

c. The **Normalize data** option is selected by default, so that data inputs are normalized before matching. If you don't want to normalize data, deselect the **Normalize data** option.

> **Note**
>
> Normalization is only supported for the following scenarios in **Create schema mapping**:
>
> - If the following **Name** sub-types are grouped: **First name**, **Middle name**, **Last name**.
> - If the following **Address** sub-types are grouped: **Street address 1**, **Street address 2**: **Street address 3 name**, **City name**, **State**, **Country**, **Postal code**.
> - If the following **Phone** sub-types are grouped: **Phone number**, **Phone country code**.

d. To specify the **Service access** permissions, choose an option and take the recommended action.

Option	Recommended action
Create and use a new service role	• AWS Entity Resolution creates a service role with the required policy for this table. • The default **Service role name** is `entityresolution-matching-workflow-<timestamp>` . • You must have permissions to create roles and attach policies. • If your input data is encrypted, choose the **This data is encrypted by a KMS key** option. Then, enter an **AWS KMS key** that is used to decrypt your data input.

Option	Recommended action
Use an existing service role	1. Choose an **Existing service role name** from the dropdown list. The list of roles are displayed if you have permissions to list roles. If you don't have permissions to list roles, you can enter the Amazon Resource Name (ARN) of the role that you want to use. If there are no existing service roles, the option to **Use an existing service role** is unavailable. 2. View the service role by choosing the **View in IAM** external link. By default, AWS Entity Resolution doesn't attempt to update the existing role policy to add necessary permissio ns.

e. (Optional) To enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.

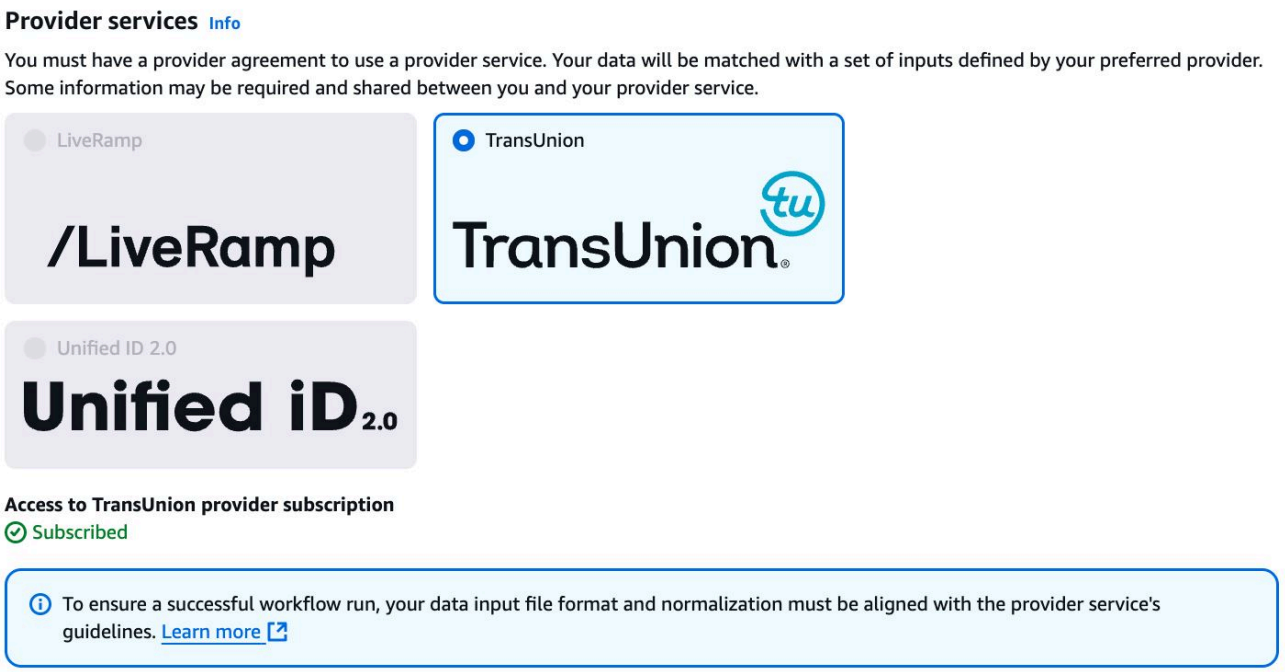
f. Choose **Next**.

5. For **Step 2: Choose matching technique**:

a. For **Matching method**, choose **Provider services**.

b. For **Provider services**, choose **TransUnion**.

> **Note**
>
> Ensure that your data input file format and normalization is aligned with the provider service's guidelines.



c. For **Data staging**, choose the **Amazon S3 location** for the temporary storage of your data while it processes.

   You must have permission to the data staging **Amazon S3 location**. For more information, see the section called “Creating a workflow job role”.

6. Choose **Next**.
7. For **Step 3: Specify data output**:

   a. For **Data output destination and format**, choose the **Amazon S3 location** for the data output and whether the **Data format** will be **Normalized data** or **Original data**.

   b. For **Encryption**, if you choose to **Customize encryption settings**, enter the **AWS KMS key** ARN.

   c. View the **TransUnion generated output**.

      This is the additional information generated by TransUnion.

   d. For **Data output**, decide which fields you want to include, hide, or mask, and then take the recommended actions based on your goals.

Your goal	Recommended option
Include fields	Keep the output state as **Included**.
Hide fields (exclude from output)	Choose the **Output field**, and then choose **Hide**.
Mask fields	Choose the **Output field**, and then choose **Hash output**.
Reset the previous settings	Choose **Reset**.

e. For **System generated output**, view all of the fields that are included.

f. Choose **Next**.

8. For **Step 4: Review and create**:

   a. Review the selections that you made for the previous steps and edit if necessary.

   b. Choose **Create and run**.

      A message appears, indicating that the matching workflow has been created and that the job has started.

9. On the matching workflow details page, on the **Metrics** tab, view the following under **Last job metrics**:

   - The **Job ID**.
   - The **Status** of the matching workflow job: **Queued**, **In progress**, **Completed**, **Failed**
   - The **Time completed** for the workflow job.
   - The number of **Records processed**.
   - The number of **Records not processed**.
   - The **Unique match IDs generated**.
   - The number of **Input records**.

   You can also view the job metrics for matching workflow jobs that have been previously run under the **Job history**.

10. After the matching workflow job completes (**Status** is **Completed**), you can go to the **Data output** tab and then select your **Amazon S3 location** to view the results.

Creating a matching workflow with UID 2.0

If you have a subscription to the Unified ID 2.0 service, you can activate advertising campaigns with deterministic identity and lean on interoperability with many UID2-enabled participants across the advertising ecosystem. For more information, see Unified ID 2.0 Overview.

The Unified ID 2.0 service provides raw UID 2, which is used for building advertising campaigns in The Trade Desk platform. UID 2.0 is generated using an open source framework.

In one workflow you can use either **`Email Address`** or **`Phone number`** for raw UID2 generation but not both. If both are present in the schema mapping, then the workflow will pick the **`Email Address`** and the **`Phone number`** will be a pass-through field. To support both, create a new schema mapping where **`Phone number`** is mapped but **`Email Address`** isn't mapped. Then, create a second workflow using this new schema mapping.

> **Note**
>
> Raw UID2s are created by adding salts from salt buckets which are rotated approximately once a year, causing the raw UID2 to also be rotated with it. Therefore, it's recommended that you refresh the raw UID2s daily. For more information, see https://unifiedid.com/docs/getting-started/gs-faqs#how-often-should-uid2s-be-refreshed-for-incremental-updates.

To create a matching workflow with UID 2.0:

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **Matching**.
3. On the **Matching workflows** page, in the upper right corner, choose **Create matching workflow**.
4. For **Step 1: Specify matching workflow details**, do the following:

    a. Enter a **Matching workflow name** and an optional **Description**.

b. For **Data input**, choose an **AWS Glue database** from the dropdown, select the **AWS Glue table**, and then select the corresponding **Schema mapping**.

   You can add up to 20 data inputs.

c. Leave the **Normalize data** option is selected, so that data inputs (**`Email Address`** or **`Phone number`**) are normalized before matching.

   For more information about **`Email Address`** normalization, see Email Address Normalization in the UID 2.0 documentation.

   For more information about **`Phone number`** normalization, see Phone Number Normalization in the UID 2.0 documentation.

d. To specify the **Service access** permissions, choose an option and take the recommended action.

Option	Recommended action
Create and use a new service role	• AWS Entity Resolution creates a service role with the required policy for this table. • The default **Service role name** is `entityresolution-matching-workflow-<timestamp>` . • You must have permissions to create roles and attach policies. • If your input data is encrypted, choose the **This data is encrypted by a KMS key** option. Then, enter an **AWS KMS key** that is used to decrypt your data input.

Option	Recommended action
Use an existing service role	1. Choose an **Existing service role name** from the dropdown list. The list of roles are displayed if you have permissions to list roles. If you don't have permissions to list roles, you can enter the Amazon Resource Name (ARN) of the role that you want to use. If there are no existing service roles, the option to **Use an existing service role** is unavailable. 2. View the service role by choosing the **View in IAM** external link. By default, AWS Entity Resolution doesn't attempt to update the existing role policy to add necessary permissio ns.

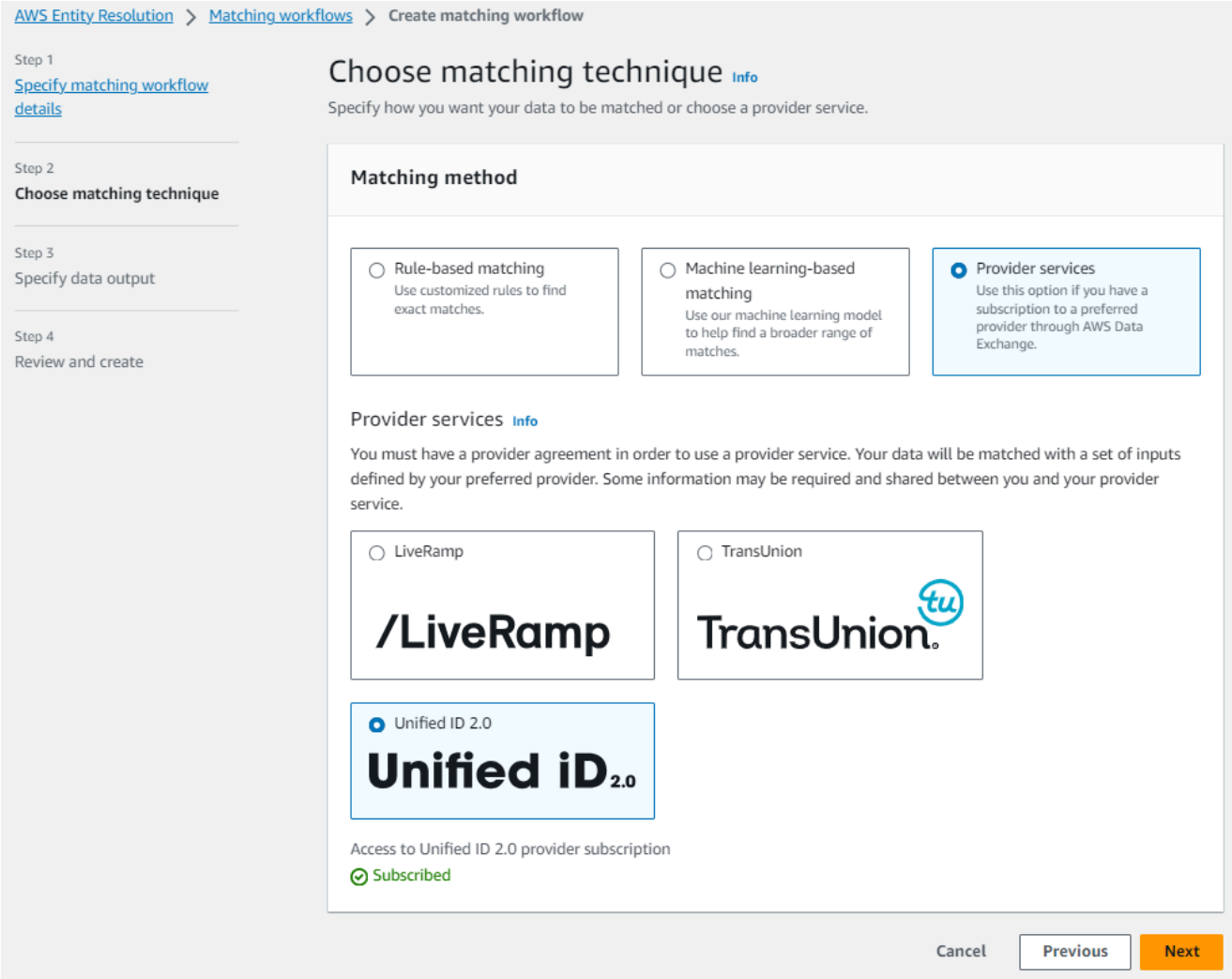
e. (Optional) To enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.

f. Choose **Next**.

5. For **Step 2: Choose matching technique**:

a. For **Matching method**, choose **Provider services**.

b. For **Provider services**, choose **Unified ID 2.0**.



c. Choose **Next**.

6. For **Step 3: Specify data output**:

   a. For **Data output destination and format**, choose the **Amazon S3 location** for the data output and whether the **Data format** will be **Normalized data** or **Original data**.

   b. For **Encryption**, if you choose to **Customize encryption settings**, enter the **AWS KMS key** ARN.

   c. View the **Unified ID 2.0 generated output**.

      This is a list of all of the additional information generated by UID 2.0

   d. For **Data output**, decide which fields you want to include, hide, or mask, and then take the recommended actions based on your goals.

Your goal	Recommended option
Include fields	Keep the output state as **Included**.
Hide fields (exclude from output)	Choose the **Output field**, and then choose **Hide**.
Mask fields	Choose the **Output field**, and then choose **Hash output**.
Reset the previous settings	Choose **Reset**.

e. For **System generated output**, view all of the fields that are included.

f. Choose **Next**.

7. For **Step 4: Review and create**:

   a. Review the selections that you made for the previous steps and edit if necessary.

   b. Choose **Create and run**.

      A message appears, indicating that the matching workflow has been created and that the job has started.

8. On the matching workflow details page, on the **Metrics** tab, view the following under **Last job metrics**:

   - The **Job ID**.
   - The **Status** of the matching workflow job: **Queued**, **In progress**, **Completed**, **Failed**
   - The **Time completed** for the workflow job.
   - The number of **Records processed**.
   - The number of **Records not processed**.
   - The **Unique match IDs generated**.
   - The number of **Input records**.

   You can also view the job metrics for matching workflow jobs that have been previously run under the **Job history**.

9. After the matching workflow job completes (**Status** is **Completed**), you can go to the **Data output** tab and then select your **Amazon S3 location** to view the results.

Editing a matching workflow

Editing the matching workflow allows you to keep your entity resolution processes up-to-date and responsive to your organization's changing requirements over time. You may want to adjust the matching criteria, techniques, or data outputs to improve the accuracy and efficiency of the entity resolution process. If you identify problems or errors in the results of the current workflow, editing it can help you diagnose and resolve those issues.

To edit a matching workflow:

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **Matching**.
3. Choose the matching workflow.
4. On the matching workflow details page, in the upper right corner, choose **Edit workflow**.
5. On the **Specify matching workflow details** page, make any necessary changes and then choose **Next**.
6. On the **Choose matching technique** page, make any necessary changes and then choose **Next**.

> **Important**
>
> You can change the **Processing cadence** from **Manual** to **Automatic**, but after you change it to **Automatic**, you can't change it back to **Manual**.
> If the **Processing cadence** is already set to **Automatic**, you can't change it to **Manual**.

7. On the **Specify data output** page, make any necessary changes and then choose **Next**.
8. On the **Review and save** page, make any necessary changes and then choose **Save**.

Deleting a matching workflow

If a matching workflow is no longer being used or has become obsolete, deleting it can help keep your workspace organized and uncluttered. If you've developed a new, improved workflow that

replaces an older one, deleting the old workflow can help ensure you're only using the most up-to-date processes.

To delete a matching workflow:

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **Matching**.
3. Choose the matching workflow.
4. On the matching workflow details page, in the upper right corner, choose **Delete**.
5. Confirm the deletion and then choose **Delete**.

Modifying or generating a Match ID for a rule-based matching workflow

A *Match ID* is the identifier generated by AWS Entity Resolution and applied to each matched record set after a matching workflow is run. This is part of the matching workflow metadata that is included in output.

When you need to update records for an existing customer or add a new customer to your dataset, you can use the AWS Entity Resolution console or the `GenerateMatchID` API. Modifying an existing match ID helps maintain consistency when updating customer information, while generating a new match ID is necessary when adding previously unidentified customers to your system.

Note

Additional charges apply, whether you use the console or the API. The processing type you choose affects both the accuracy and response time of the operation.

Important

If you revoke AWS Entity Resolution permissions to your S3 bucket while a job is in progress, AWS Entity Resolution will still process and charge for outputting results to S3 but can't deliver the results to your bucket. To avoid this issue, make sure that AWS

Entity Resolution has the correct permissions to write to your S3 bucket before starting a job. If permissions are revoked during processing, AWS Entity Resolution attempts to re-deliver results for up to 30 days after job completion once you restore the correct bucket permissions.

The following procedure guides you through the process of looking up or generating a Match ID, selecting a processing type, and viewing the results.

Console

To modify or generate a Match ID using the console

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **Matching**.
3. Choose the rule-based matching workflow that has been processed (**Job status** is **Completed**).
4. On the matching workflow details page, choose the **Match IDs** tab.
5. Choose **Modify or generate match ID**.

> **Note**
>
> The **Modify or generate match ID** option is only available for matching workflows that use the **Automatic** processing cadence. If you have selected the **Manual** processing cadence, this option will appear inactive. To use this option, edit your workflow to use the **Automatic** processing cadence. For more information about editing workflows, see Editing a matching workflow.

6. Select the **AWS Glue table** from the dropdown list.

   If there is only one AWS Glue table in the workflow, it's selected by default.

7. Choose the **Processing type**.

   - **Consistent** – You can look up an existing match ID or generate and save a new match ID immediately. This option has the highest accuracy and the slower response time.

- **Background** (shown as `EVENTUAL` in the API) – You can look up an existing match ID or generate a new match ID immediately. The updated record is saved in the background. This option has a fast initial response, with complete results available in S3 later.
- **Quick ID generation** (shown as `EVENTUAL_NO_LOOKUP` in the API) – You can create a new match ID without looking up an existing one. The updated record is saved in the background. This option has the fastest response. It is recommended for unique records only.

8. For **Record attributes**,
   a. Enter the **Value** for the **Unique ID**.
   b. Enter a **Value** for each **Match key** that will match with existing records based on the rules configured in your workflow.
9. Choose **Find match ID and save record**.

   A success message appears, stating that either the Match ID was found or a new Match ID was generated and the record was saved.
10. View the corresponding Match ID and the associated rule that was saved to the matching workflow in the success message.
11. (Optional) To copy the match ID, choose **Copy**.

API

To modify or generate a Match ID using the API

> **Note**
>
> To call this API successfully, you must have first successfully run a rule-based matching workflow using the StartMatchingJob API.
> For a complete list of supported programming languages, see the See Also section of the GenerateMatchID.

1. Open a terminal or command prompt to make the API request.
2. Create a POST request to the following endpoint:

```
/matchingworkflows/workflowName/generateMatches
```

3. In the request header, set the Content-type to application/json.
4. In the request URI, specify your `workflowName`.

   The `workflowName` must:

   - Be between 1 and 255 characters long
   - Match the pattern [a-zA-Z_0-9-]*
5. For the request body, provide the following JSON:

```
{
   "processingType": "string",
   "records": [
      {
         "inputSourceARN": "string",
         "recordAttributeMap": {
            "string" : "string"
         },
         "uniqueId": "string"
      }
   ]
}
```

   Where:

   - `processingType` (optional) - Defaults to `CONSISTENT`. Choose one of these values:
     - `CONSISTENT` - For highest accuracy with slower response time
     - `EVENTUAL` - For faster initial response with background processing
     - `EVENTUAL_NO_LOOKUP` - For fastest response when records are known to be unique
   - `records` (required) - Array containing exactly one record object
6. Send the request.

   If successful, you'll receive a response with status code 200 and a JSON body containing:

```
{
   "failedRecords": [
      {
         "errorMessage": "string",
         "inputSourceARN": "string",
         "uniqueId": "string"
```

```
      }
   ],
   "matchGroups": [
      {
         "matchId": "string",
         "matchRule": "string",
         "records": [
            {
               "inputSourceARN": "string",
               "recordId": "string"
            }
         ]
      }
   ]
}
```

If the call is unsuccessful, you might receive one of these errors:

- 403 - AccessDeniedException if you don't have sufficient access
- 404 - ResourceNotFoundException if the resource can't be found
- 429 - ThrottlingException if the request was throttled
- 400 - ValidationException if the input fails validation
- 500 - InternalServerException if there's an internal service failure

Looking up a Match ID for a rule-based matching workflow

After completing a rule-based matching workflow, you can retrieve the Match ID and associated rule for each processed record. This information helps you understand how records were matched and which rules were applied. The following procedure demonstrates how to access this data using either the AWS Entity Resolution console or the `GetMatchID` API.

Console

To look up a Match ID using the console

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **Matching**.

3. Choose the rule-based matching workflow that has been processed (**Job status** is **Completed**).
4. On the matching workflow details page, choose the **Match IDs** tab.
5. Choose **Look up match ID**.

> **Note**
>
> The **Look up match ID** option is only available for matching workflows that use the **Automatic** processing cadence. If you have selected the **Manual** processing cadence, this option will appear inactive. To use this option, edit your workflow to use the **Automatic** processing cadence. For more information about editing workflows, see Editing a matching workflow.

6. Do one of the following:

If ...	Then ...
There is only one schema mapping associated with this workflow.	View the **Schema mapping** that's selected by default.
There is more than one schema mapping associated with this workflow.	Choose the **Schema mapping** from the dropdown list.

7. For **Record attributes**, enter the **Value** for an existing **Match key** to look up for each existing record.

> **Tip**
>
> Enter as many values as you can to help find the Match ID.

8. The **Normalize data** option is selected by default, so that data inputs are normalized before matching. If you don't want to normalize data, deselect the **Normalize data** option.
9. If you want to view the matching rules expand the **View matching rules**.
10. Choose **Look up**.

    A success message appears, stating that the Match ID was found.
11. View the corresponding Match ID and the associated rule that was found.

API

To look up a Match ID using the API

> **Note**
>
> To call this API successfully, you must have first successfully run a rule-based matching workflow using the StartMatchingJob API.
> For a complete list of supported programming languages, see the See Also section of the GetMatchID API.

1. Open a terminal or command prompt to make the API request.
2. Create a POST request to the following endpoint:

```
/matchingworkflows/workflowName/matches
```

3. In the request header, set the Content-type to application/json.
4. In the request URI, specify your `workflowName`.

   The `workflowName` must:

   - Be between 1 and 255 characters long
   - Match the pattern [a-zA-Z_0-9-]*
5. For the request body, provide the following JSON:

```
{
   "applyNormalization": boolean,
   "record": {
      "string" : "string"
   }
}
```

   Where:

   `applyNormalization` (optional) - Set to `true` to normalize attributes defined in the schema

   `record` (required) - The record to fetch the Match ID for

6. Send the request.

   If successful, you'll receive a response with status code 200 and a JSON body containing:

   ```
   {
      "matchId": "string",
      "matchRule": "string"
   }
   ```

   The `matchId` is the unique identifier for this group of matched records, and `matchRule` indicates which rule the record matched on.

   If the call is unsuccessful, you might receive one of these errors:

   - 403 - AccessDeniedException if you don't have sufficient access
   - 404 - ResourceNotFoundException if the resource can't be found
   - 429 - ThrottlingException if the request was throttled
   - 400 - ValidationException if the input fails validation
   - 500 - InternalServerException if there's an internal service failure

Deleting records from a rule-based or ML-based matching workflow

If you need to comply with data management regulations, you can delete the records from either a rule-based or ML-based matching workflow.

To delete records from a rule-based or ML-based matching workflow

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **Matching**.
3. Choose the rule-based or ML-based matching workflow.
4. On the matching workflow details page, choose **Delete unique IDs** from the **Actions** dropdown list.
5. Enter the unique ID you want to delete in the **Unique IDs** section.

   You can enter up to 10 unique IDs.

6. Specify the **Input source** from which to delete the unique IDs.

   If there is only one **Input source** for the workflow, the **Input source** is listed by default.

   If you only specify one **Input source**, the unique IDs in other input sources won't be affected.
7. Choose **Delete unique IDs**.

Troubleshooting matching workflows

Use the following information to help you diagnose and fix common issues that you might encounter when running matching workflows.

I received an error file after running a matching workflow

Common cause

A matching workflow can have multiple runs and the results (successes or errors) are written to a folder with the `jobId` as the name.

The successful results for a matching workflow are written to a `success` folder that contains multiple files, and each file contains a subset of the successful records.

The errors for a matching workflow are written to an `error` folder with multiple fields, with each containing a subset of the error records.

The error file can be created for the following reasons:

- The Unique ID is:
  - null
  - missing in a row of data
  - missing in a record in the data table
  - repeated in another row of data in the data table
  - not specified
  - not unique within the same source
  - not unique across multiple sources
  - overlaps across sources
  - exceeds 38 characters (rule-based matching workflow only)

- One of the fields in the schema mapping includes a reserved name:
  - EmailAddress
  - InputSourceARN
  - MatchRule
  - MatchID
  - HashingProtocol
  - ConfidenceLevel
  - Source

> **Note**
>
> If the record in the error file is created due to the reasons listed previously, you are charged, because it incurs processing cost for the service. If the record in the error file is because of an internal server error, you aren't charged.

Resolution

To resolve this issue

1. Check to see if the Unique ID is valid.

   If the Unique ID isn't valid, update the Unique ID in your data table, save the new data table, create a new schema mapping, and run the matching workflow again.
2. Check if one of the fields in the schema mapping includes a reserved name.

   If one of the fields includes a reserved name, create a new schema mapping with a new name, and run the matching workflow again.

Map input data using an ID mapping workflow

An *ID mapping workflow* is a data processing job that maps data from an input data source to an input data target based on the specified ID mapping method. It produces an ID mapping table.

An ID mapping workflow requires an input data source and an input data target. Your data input source and target depends on the type of ID mapping that you want to perform. There are two ways to perform ID mapping: rule-based or provider services:

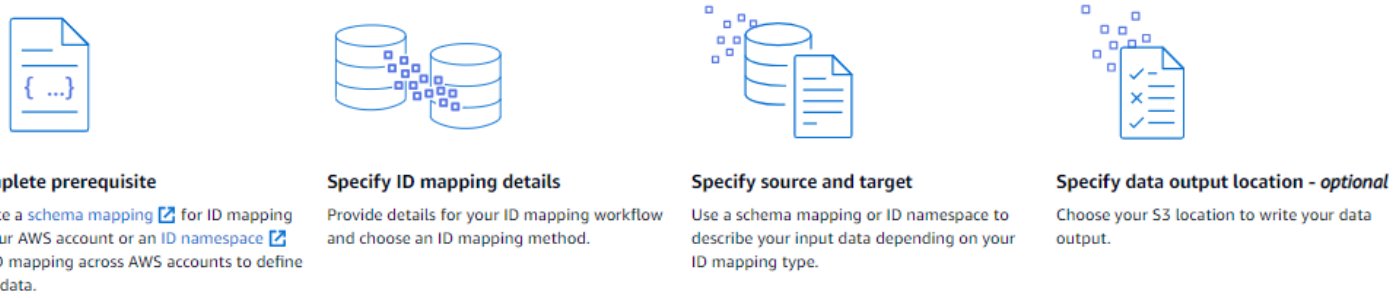
- Rule-based ID mapping – You use matching rules to translate first-party data from a source to a target.
- Provider services ID mapping – You use the LiveRamp provider service to translate third-party data from a source to a target.

> **Note**
>
> The provider services ID mapping workflow in AWS Entity Resolution is currently integrated with LiveRamp. If you have a subscription to the LiveRamp service, then you can create an ID mapping workflow with LiveRamp to perform transcoding. With LiveRamp transcoding, you can translate a set of source RampIDs into any target destination RampID. By using the RampID as a token to represent your customers, you can avoid sharing customer data directly with advertising platforms.
> For more information, see Perform Translation Through ADX on the LiveRamp documentation website.

You can perform ID mapping between two datasets in either of the following scenarios:

- Within your own AWS account
- Across two different AWS accounts

The following diagram summarizes how to set up an ID mapping workflow.



Topics

- ID mapping workflow for one AWS account
- ID mapping workflow across two AWS accounts
- Running an ID mapping workflow
- Running an ID mapping workflow with a new output destination
- Editing an ID mapping workflow
- Deleting an ID mapping workflow
- Adding or updating a resource policy for an ID mapping workflow

ID mapping workflow for one AWS account

An *ID mapping workflow for one AWS account* enables you to perform ID mapping between two datasets on your own AWS account.

Before you create an ID mapping workflow on your own AWS account, you must first complete the prerequisites.

After you create and run an ID mapping workflow, you can view the output (the ID mapping table) and use it for analysis.

The following topics guide you through a set of steps to create an ID mapping workflow in the same AWS account.

Topics

- Prerequisites
- Creating an ID mapping workflow (rule-based)
- Creating an ID mapping workflow (provider services)

Prerequisites

Before you create an ID mapping workflow for one AWS account using either the **Rule-based** or the **Provider services** ID mapping method, you must first do the following:

- Complete the tasks in Setting up AWS Entity Resolution.
- Complete the tasks in Prepare input data tables, depending on the type of input data you are using.
- Create a schema mapping or Create a matching workflow.
- (**Provider services** ID mapping only) Before you create an ID mapping workflow with LiveRamp, you must choose an Amazon Simple Storage Service (Amazon S3) data staging bucket where you want to temporarily write the ID mapping workflow output.

  If you are using the LiveRamp provider service to translate third-party data, add the following permissions policy, which allows you to access the data staging bucket.

  JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::715724997226:root"

            },
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::<staging-bucket>",
                "arn:aws:s3:::<staging-bucket>/*"
            ]
        },
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": "arn:aws:iam::715724997226:root"
            },
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicy",
                "s3:ListBucketVersions",
                "s3:GetBucketAcl"
            ],
            "Resource": [
                "arn:aws:s3:::<staging-bucket>",
                "arn:aws:s3:::<staging-bucket>/*"
            ]
        }
    ]
}
```

In the preceding permissions policy, replace each *<user input placeholder>* with your own information.

staging-bucket	The Amazon S3 bucket that temporarily stores your data while running a provider service-based workflow.

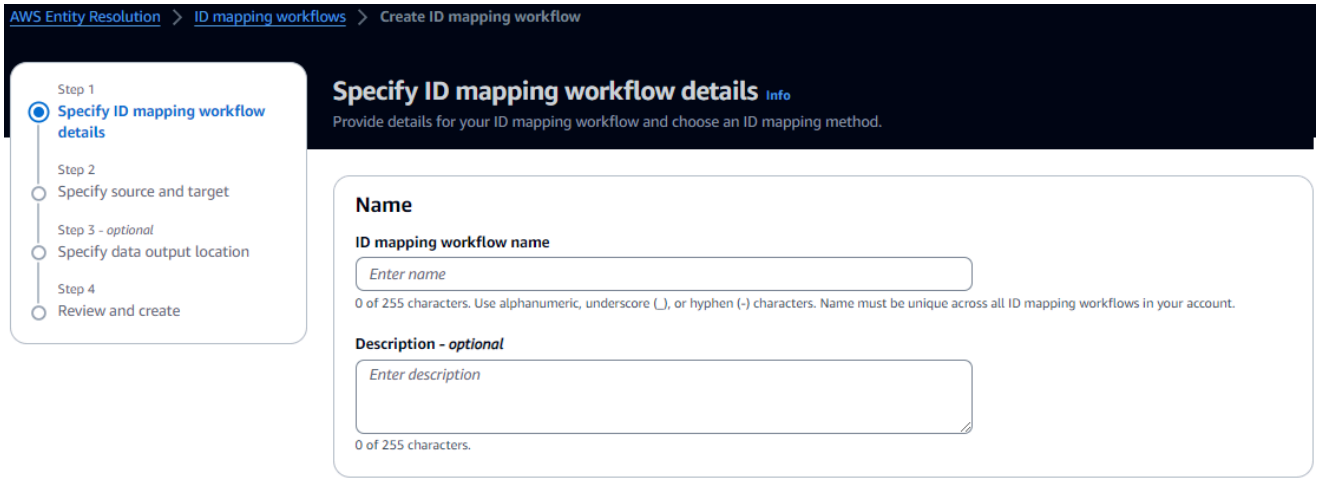
Creating an ID mapping workflow (rule-based)

This topic describes the process of creating an ID mapping workflow for one AWS account that uses matching rules to translate first-party data from a source to a target.

To create a rule-based ID mapping workflow for one AWS account

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **ID mapping**.
3. On the **ID mapping workflows** page, in the upper right corner, choose **Create ID mapping workflow**.
4. For **Step 1: Specify ID mapping workflow details**, do the following.

a. Enter an **ID mapping workflow name** and an optional **Description**.



b. For the ID mapping method, choose **Rule-based**.

c. (Optional) To enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.

d. Choose **Next**.

5. For **Step 2: Specify source and target**, do the following.

a. For **Source**, choose the scenario that applies to you and then take the recommended action.

Scenario	Recommended action
Use your own AWS Glue database, AWS Glue table, and schema mapping in the ID mapping workflow.	1. Choose **Schema mapping**. 2. Select an **AWS Glue** database from the dropdown, select the **AWS Glue table**, and then select the correspon ding **Schema mapping**. You can add up to 19 data inputs.
Use an existing matching workflow that points to the record data you want to use in the ID mapping workflow.	1. Choose **Matching workflow**. 2. Select an existing **Matching workflow** from the dropdown list.

b. For **Target**, select an existing **Matching workflow** from the dropdown list.

c. For **Rule parameters**, do the following.

i. Specify the **Rule controls** by choosing one of the following options based on your source type.

Source type	Recommended action
Matching workflow	Specify the **Rule controls** by choosing whether a **Source**, **Target**, or both can provide rules in an ID mapping workflow. **Rule controls** must be compatible between the source and the target to be used in an ID mapping workflow. For example, if a source ID namespace limits rules to the target but the target ID namespace limits rules to the source, this results in an error.
Schema mapping	Skip this step.

ii. For **Comparison and matching parameters**, the **Comparison type** is automatically set to **Multiple input fields**.

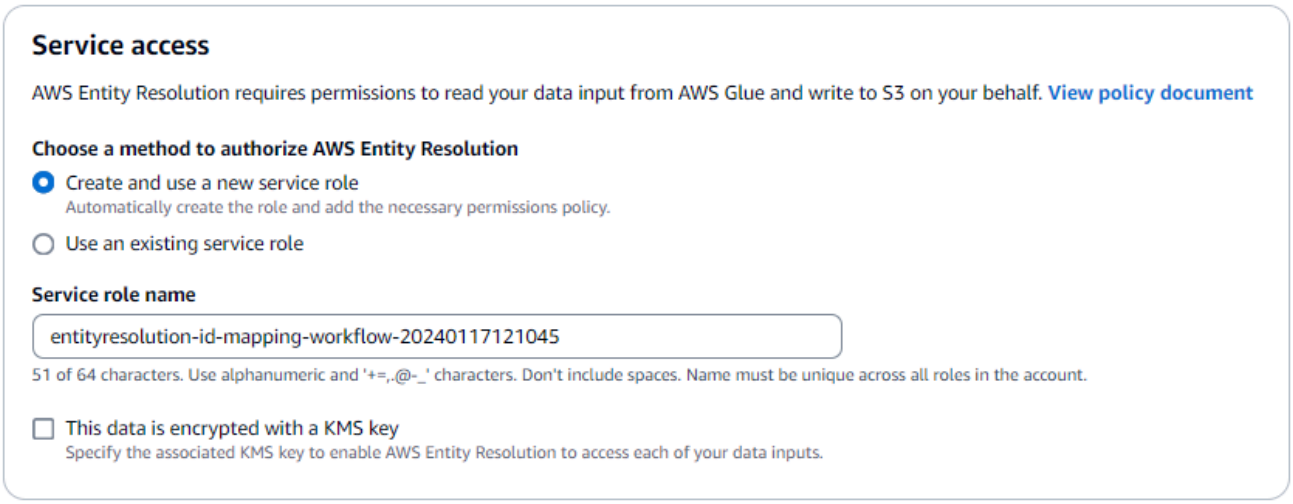
This is because both participants had selected this option previously.

d. Specify the **Record matching type** by choosing one of the following options based on your goal.

Your goal	Recommended option
Limit the record matching type to store only one matching record in the source for each matched record in the target when you create the ID mapping workflow.	**One source to one target**
Limit the record matching type to store all matching records in the source for each matched record in the target when you create the ID mapping workflow.	**Many sources to one target**

> **Note**
>
> You must specify compatible limitations for the source and target ID namespaces.

e. To specify the **Service access** permissions, choose an option and take the recommended action.



Option	Recommended action
Create and use a new service role	• AWS Entity Resolution creates a service role with the required policy for this table. • The default **Service role name** is `entityresolution-id-mapping-workflow-<timestamp>` . • You must have permissions to create roles and attach policies. • If your input data is encrypted, choose the **This data is encrypted by a KMS key** option. Then, enter an **AWS KMS key** that is used to decrypt your data input.

Option	Recommended action
Use an existing service role	1. Choose an **Existing service role name** from the dropdown list. The list of roles are displayed if you have permissions to list roles. If you don't have permissions to list roles, you can enter the Amazon Resource Name (ARN) of the role that you want to use. If there are no existing service roles, the option to **Use an existing service role** is unavailable. 2. View the service role by choosing the **View in IAM** external link. By default, AWS Entity Resolution doesn't attempt to update the existing role policy to add necessary permissio ns.

6. Choose **Next**.
7. For **Step 3: Specify data output location – *optional***, do the following.
   a. For **Data output destination**, do the following:
      i. Choose the **Amazon S3 location** for the data output.
      ii. For **Encryption**, if you choose to **Customize encryption settings**, then enter the **AWS KMS key** ARN or choose **Create an AWS KMS key**.
   b. Choose **Next**.
8. For **Step 4: Review and create**, do the following.
   a. Review the selections that you made for the previous steps and edit them if necessary.
   b. Choose **Create**.

A message appears, indicating that the ID mapping workflow has been created.

After you create the ID mapping workflow, you're ready to run an ID mapping workflow.

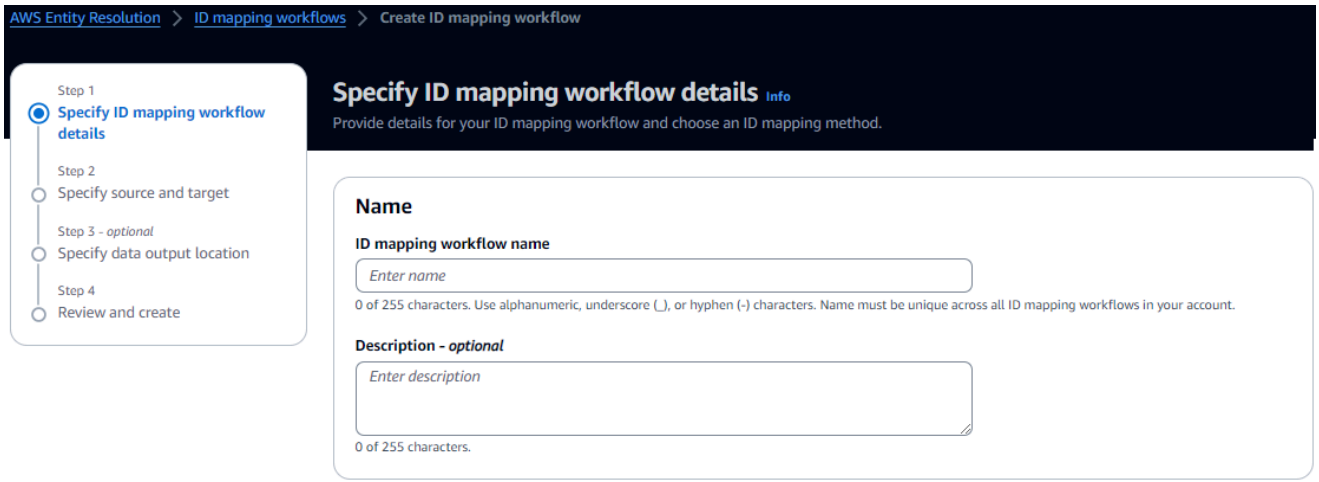
Creating an ID mapping workflow (provider services)

This topic describes the process of creating an ID mapping workflow for one AWS account using a provider service called LiveRamp. LiveRamp translates a set of source RampIDs to another set using either maintained or derived RampIDs.

To create a provider service-based ID mapping workflow for one AWS account

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **ID mapping**.
3. On the **ID mapping workflows** page, in the upper right corner, choose **Create ID mapping workflow**.
4. For **Step 1: Specify ID mapping workflow details**, do the following.

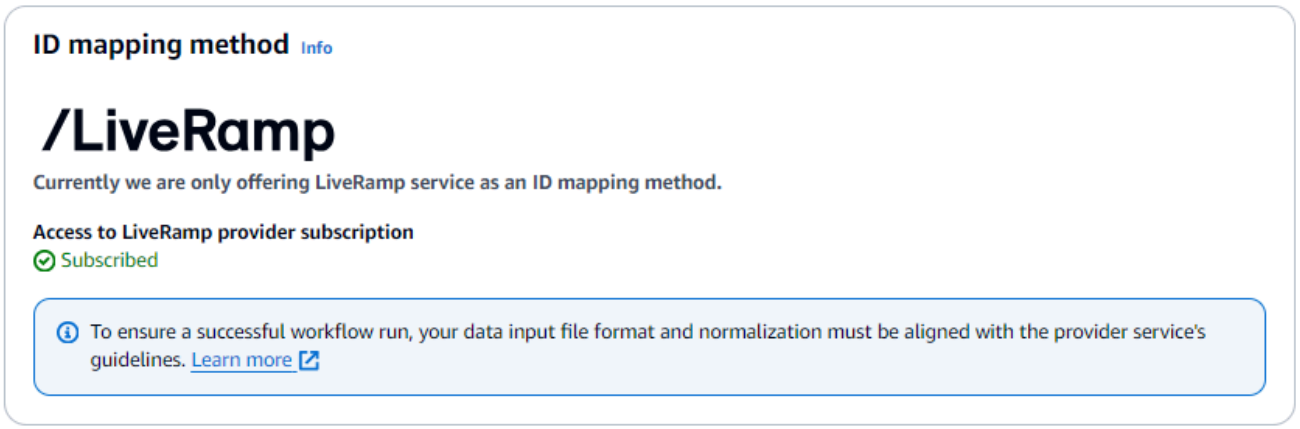
   a. Enter an **ID mapping workflow name** and an optional **Description**.



   b. For the **ID mapping method**, choose **Provider services**.

      AWS Entity Resolution currently offers the LiveRamp provider service as an ID mapping method. If you have a subscription to LiveRamp, then the status appears as **Subscribed**.

For more information about how to subscribe to LiveRamp, see Step 1: Subscribe to a provider service on AWS Data Exchange.



> **Note**
>
> Ensure that your data input file format aligns with the provider service's guidelines. For more information about LiveRamp's input file formatting guidelines, see Perform Translation Through ADX on the LiveRamp documentation website.

c. For **LiveRamp configuration**, enter the following values that LiveRamp provides:

- **Client ID manager ARN**
- **Client secret manager ARN**

d. (Optional) To enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.

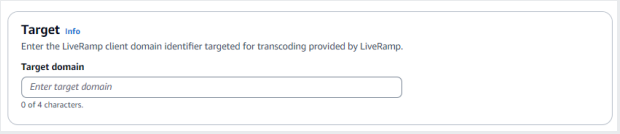
e. Choose **Next**.

5. For **Step 2: Specify source and target**, do the following.

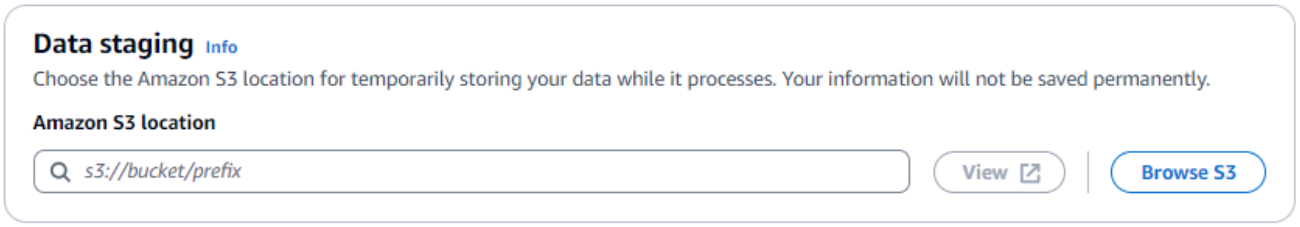
   a. For **Source**, choose the scenario that applies to you and then take the recommended action.

Scenario	Recommended action
Use your own AWS Glue database, AWS Glue table, and schema mapping in the ID mapping workflow.	1. Choose **Schema mapping**. 2. Select an **AWS Glue** database from the dropdown, select the **AWS Glue table**, and then select the correspon ding **Schema mapping**. You can add up to 19 data inputs.
Use an existing matching workflow that points to the record data you want to use in the ID mapping workflow.	1. Choose **Matching workflow**. 2. Select an existing **Matching workflow** from the dropdown list.

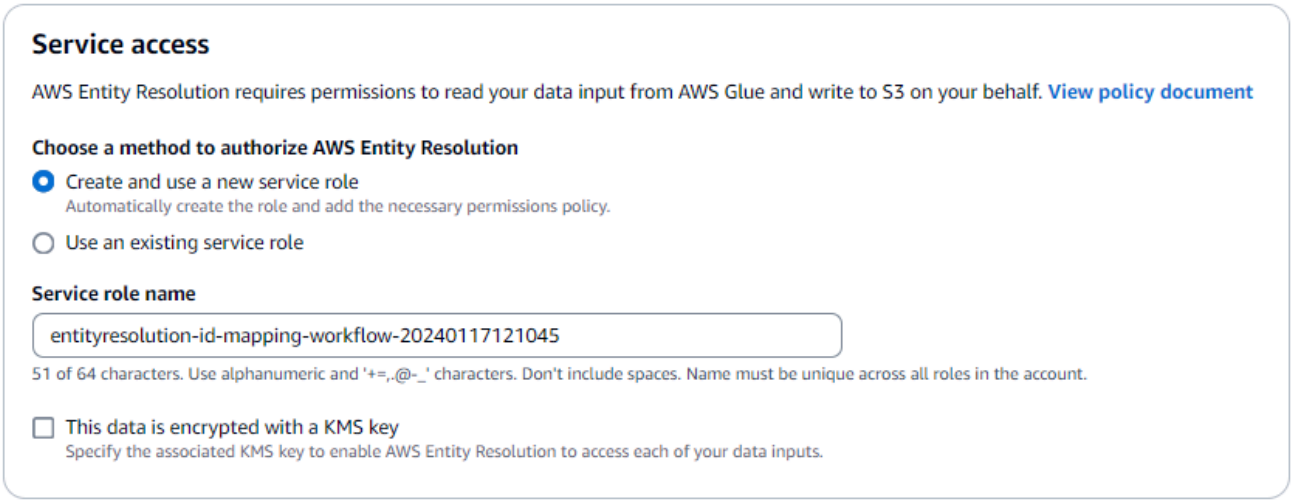
   b. For **Target**, take one of the following actions based on your chosen ID mapping method.

ID mapping method	Recommended action
Rule-based	Select an existing **Matching workflow** from the dropdown list.
Provider services	Enter the LiveRamp client domain identifier targeted for transcoding that LiveRamp provides in the **Target domain**. 

   c. For **Data staging**, choose the **Amazon S3 location** where you want to temporarily write the ID mapping workflow output.



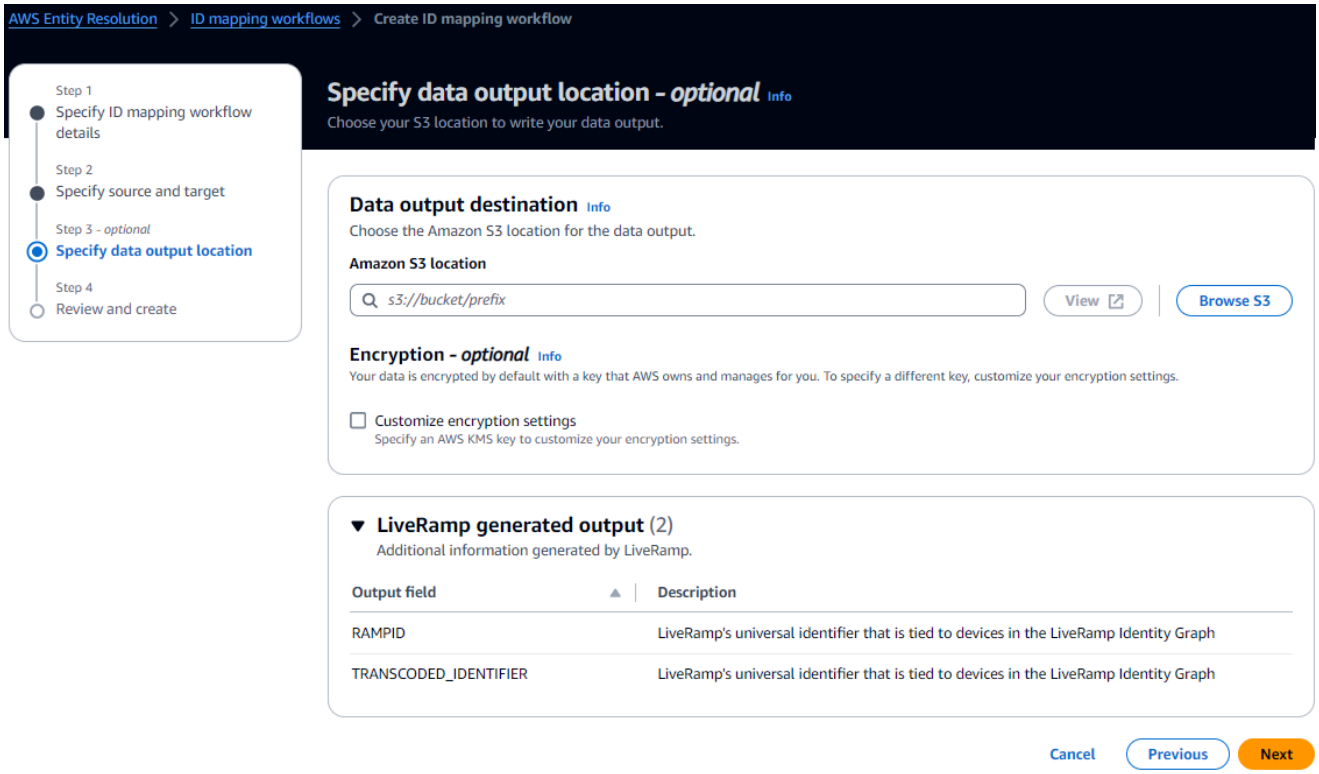
d. To specify the **Service access** permissions, choose an option and take the recommended action.



Option	Recommended action
Create and use a new service role	• AWS Entity Resolution creates a service role with the required policy for this table. • The default **Service role name** is `entityresolution-id-mapping-workflow-<timestamp>` . • You must have permissions to create roles and attach policies. • If your input data is encrypted, choose the **This data is encrypted by a KMS key** option. Then, enter an **AWS KMS key** that is used to decrypt your data input.

Option	Recommended action
Use an existing service role	1. Choose an **Existing service role name** from the dropdown list. The list of roles are displayed if you have permissions to list roles. If you don't have permissions to list roles, you can enter the Amazon Resource Name (ARN) of the role that you want to use. If there are no existing service roles, the option to **Use an existing service role** is unavailable. 2. View the service role by choosing the **View in IAM** external link. By default, AWS Entity Resolution doesn't attempt to update the existing role policy to add necessary permissio ns.

6. Choose **Next**.
7. For **Step 3: Specify data output location – *optional***, do the following.
   a. For **Data output destination**, do the following:
      i. Choose the **Amazon S3 location** for the data output.
      ii. For **Encryption**, if you choose to **Customize encryption settings**, then enter the **AWS KMS key** ARN or choose **Create an AWS KMS key**.
   b. View the **LiveRamp generated output**.
   c. Choose **Next**.



8. For **Step 4: Review and create**, do the following.

   a. Review the selections that you made for the previous steps and edit them if necessary.

   b. Choose **Create**.

   A message appears, indicating that the ID mapping workflow has been created.

9. After you create the ID mapping workflow, you're ready to run an ID mapping workflow.

ID mapping workflow across two AWS accounts

An *ID mapping workflow across two AWS accounts* enables you to perform ID mapping between two datasets across two AWS accounts. This is typically done between your own AWS account and another AWS account.

For example, a publisher can create an ID mapping workflow using their own target ID namespace (in their own AWS account) and an advertiser's source ID namespace (in another AWS account).

Before you create an ID mapping workflow across two AWS accounts, you must first complete the prerequisites.

After you create an ID mapping workflow, you can view the output (the ID mapping table) and use it for analysis.

The following topics guide you through a set of steps to create an ID mapping workflow across two AWS accounts:

Topics

Prerequisites

Before you create an ID mapping workflow across two AWS accounts, you must first do the following:

- Complete the tasks in Set up AWS Entity Resolution.
- Create an ID namespace source.
- Create an ID namespace target.
- Acquire the ID namespace ARN if you are using an ID namespace source from another AWS account.
- (**Provider services** only) Creating an ID mapping workflow across two AWS accounts requires permission for LiveRamp to access the S3 bucket and the AWS Key Management Service (AWS KMS) customer managed key.

  Before you create an ID mapping workflow across two AWS accounts with LiveRamp, add the following permission policy, which allows LiveRamp to access the S3 bucket and the customer managed key.

  In the preceding permissions policy, replace each *<user input placeholder>* with your own information.

<KMSKeyARN>	The ARN of an AWS KMS customer managed key.

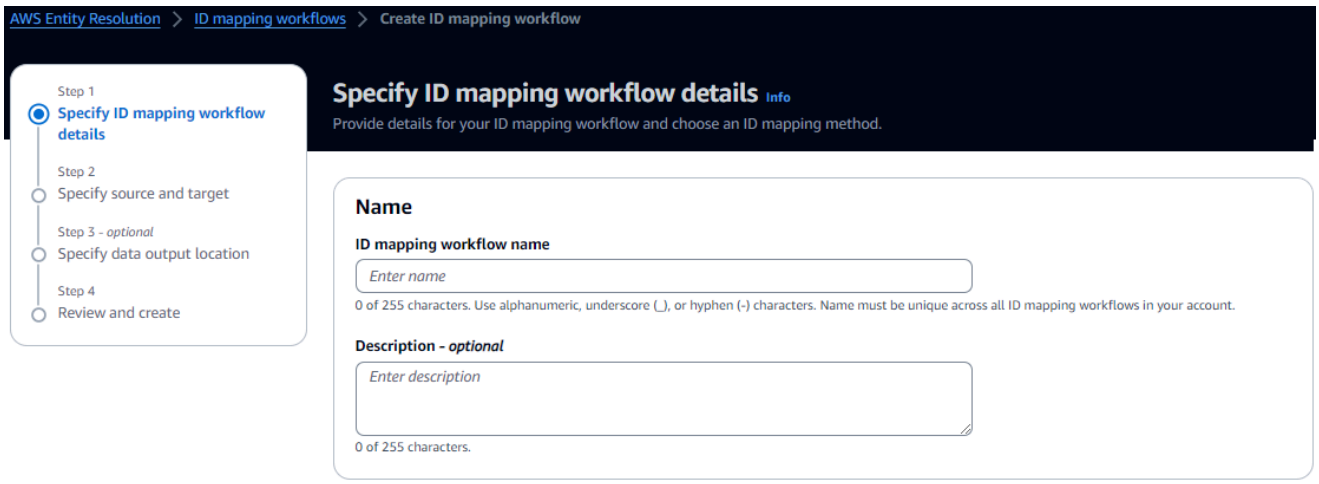
Creating an ID mapping workflow (rule-based)

After you've completed the prerequisites, you can create one or more ID mapping workflows to use matching rules to translate first-party data from a source to a target.

To create a rule-based ID mapping workflow across two AWS accounts

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **ID mapping**.
3. On the **ID mapping workflows** page, in the upper right corner, choose **Create ID mapping workflow**.
4. For **Step 1: Specify ID mapping workflow details**, do the following.

   a. Enter an **ID mapping workflow name** and an optional **Description**.



   b. For the **ID mapping method**, choose **Rule-based**.

   c. (Optional) To enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.

   d. Choose **Next**.

5. For **Step 2: Specify source and target**, do the following.

   a. Turn on **Advanced options**.

   b. For **Source**, choose **Matching workflow**, and then select the existing **Matching workflow** from the dropdown list.

c. For **Target**, choose **Matching workflow**, and then select the existing **Matching workflow** from the dropdown list.

d. For **Rule parameters**, specify the **Rule controls** by choosing whether a **Source** or a **Target** can provide rules in an ID mapping workflow.

   Rule controls must be compatible between the source and the target to be used in an ID mapping workflow. For example, if a source ID namespace limits rules to the target but the target ID namespace limits rules to the source, this results in an error.

e. For **Comparison and matching parameters**, do the following.

   i. Specify the **Comparison type** by choosing an option based on your goal.

Your goal	Recommended option
Find any combination of matches across data stored in multiple input fields, regardless of whether the data is in the same or different input field.	**Multiple input fields**
Limit comparison within a single input field, when similar data stored across multiple input fields shouldn't be matched.	**Single input field**

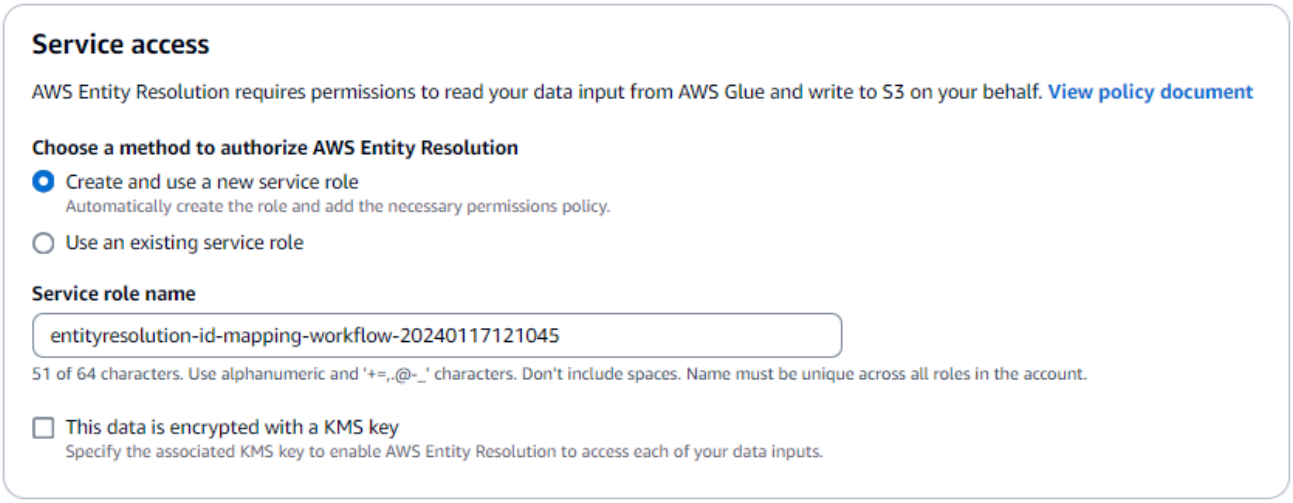
   ii. Specify the **Record matching type** by choosing an option based on your goal.

Your goal	Recommended option
Limit the record matching type to store only one matching record in the source for each matched record in the target when you create the ID mapping workflow.	**One source to one target**

Your goal	Recommended option
Limit the record matching type to store all matching records in the source for each matched record in the target when you create the ID mapping workflow.	**Many sources to one target**

> **Note**
>
> You must specify compatible limitations for the source and target ID namespaces.

f. To specify the **Service access** permissions, choose an option and take the recommended action.



Option	Recommended action
Create and use a new service role	• AWS Entity Resolution creates a service role with the required policy for this table. • The default **Service role name** is `entityresolution-id-mapping-workflow-<timestamp>` . • You must have permissions to create roles and attach policies. • If your input data is encrypted, choose the **This data is encrypted by a KMS key** option. Then, enter an **AWS KMS key** that is used to decrypt your data input.

Option	Recommended action
Use an existing service role	1. Choose an **Existing service role name** from the dropdown list. The list of roles are displayed if you have permissions to list roles. If you don't have permissions to list roles, you can enter the Amazon Resource Name (ARN) of the role that you want to use. If there are no existing service roles, the option to **Use an existing service role** is unavailable. 2. View the service role by choosing the **View in IAM** external link. By default, AWS Entity Resolution doesn't attempt to update the existing role policy to add necessary permissio ns.

6. Choose **Next**.
7. For **Step 3: Specify data output location – *optional***, do the following.
   a. For **Data output destination**, do the following.
      i. Choose the **Amazon S3 location** for the data output.
      ii. For **Encryption**, if you choose to **Customize encryption settings**, then enter the **AWS KMS key** ARN or choose **Create an AWS KMS key**.
   b. View the **LiveRamp generated output**.
   c. Choose **Next**.
8. For **Step 4: Review and create**, do the following.
   a. Review the selections that you made for the previous steps and edit them if necessary.
   b. Choose **Create**.

A message appears, indicating that the ID mapping workflow has been created.

After you create the ID mapping workflow, you're ready to run an ID mapping workflow.

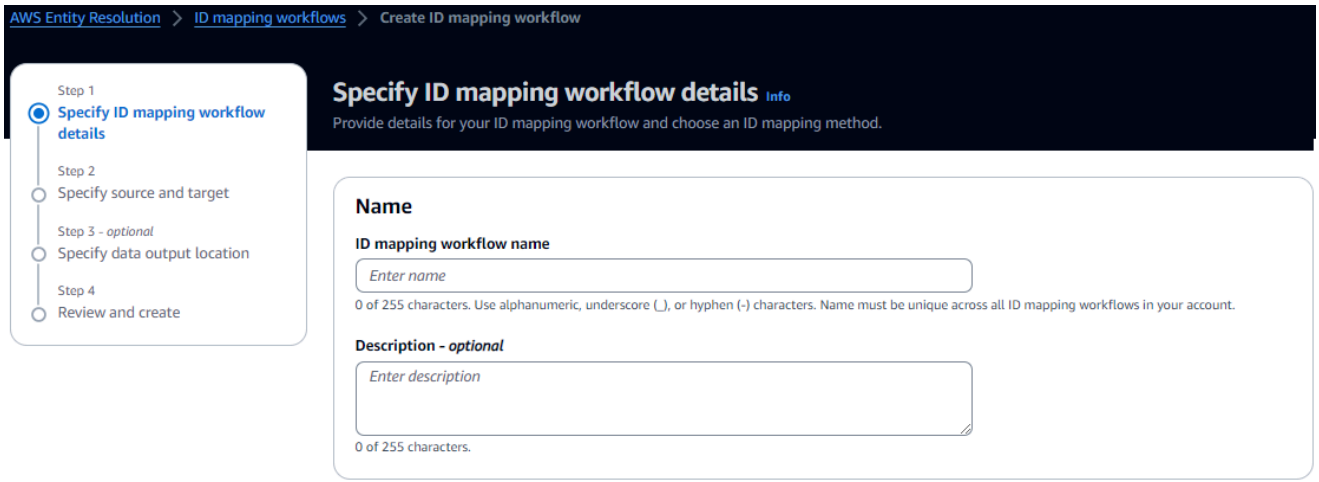
Creating an ID mapping workflow (provider services)

After completing the prerequisites, you can create one or more ID mapping workflows using the LiveRamp provider service. LiveRamp translates a set of source RampIDs to another set using either maintained or derived RampIDs.

To create an ID mapping workflow using the provider service

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **ID mapping**.
3. On the **ID mapping workflows** page, in the upper right corner, choose **Create ID mapping workflow**.
4. For **Step 1: Specify ID mapping workflow details**, do the following.

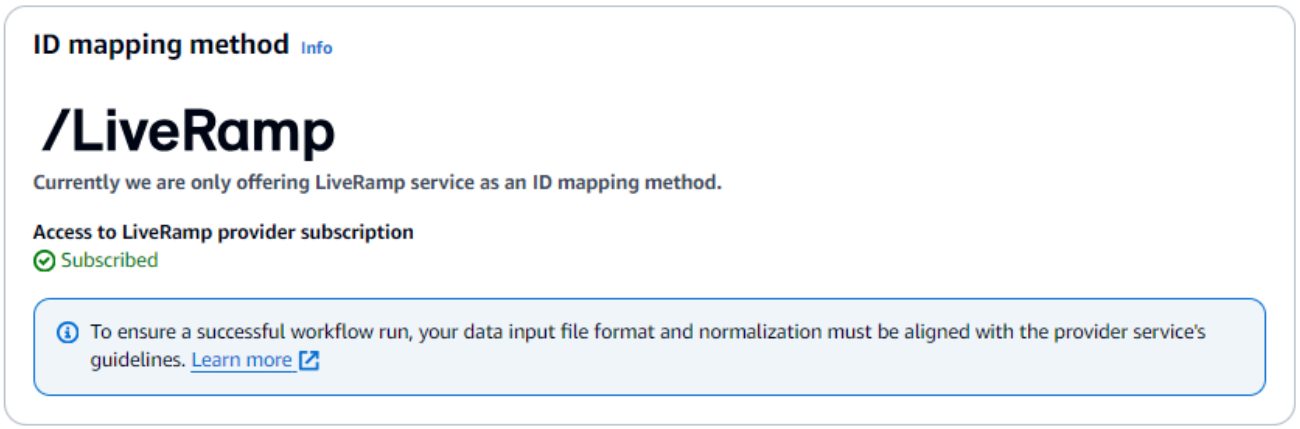
   a. Enter an **ID mapping workflow name** and an optional **Description**.



   b. For the **ID mapping method**, choose **Provider services**.

      AWS Entity Resolution currently offers the LiveRamp provider service as an ID mapping method. If you have a subscription to LiveRamp, then the status appears as **Subscribed**.

For more information about how to subscribe to LiveRamp, see Step 1: Subscribe to a provider service on AWS Data Exchange.



> **Note**
>
> Ensure that your data input file format aligns with the provider service's guidelines. For more information about LiveRamp's input file formatting guidelines, see Perform Translation Through ADX on the LiveRamp documentation website.

c. For **LiveRamp configuration**, enter the following values that LiveRamp provides:

- **Client ID manager ARN**
- **Client secret manager ARN**

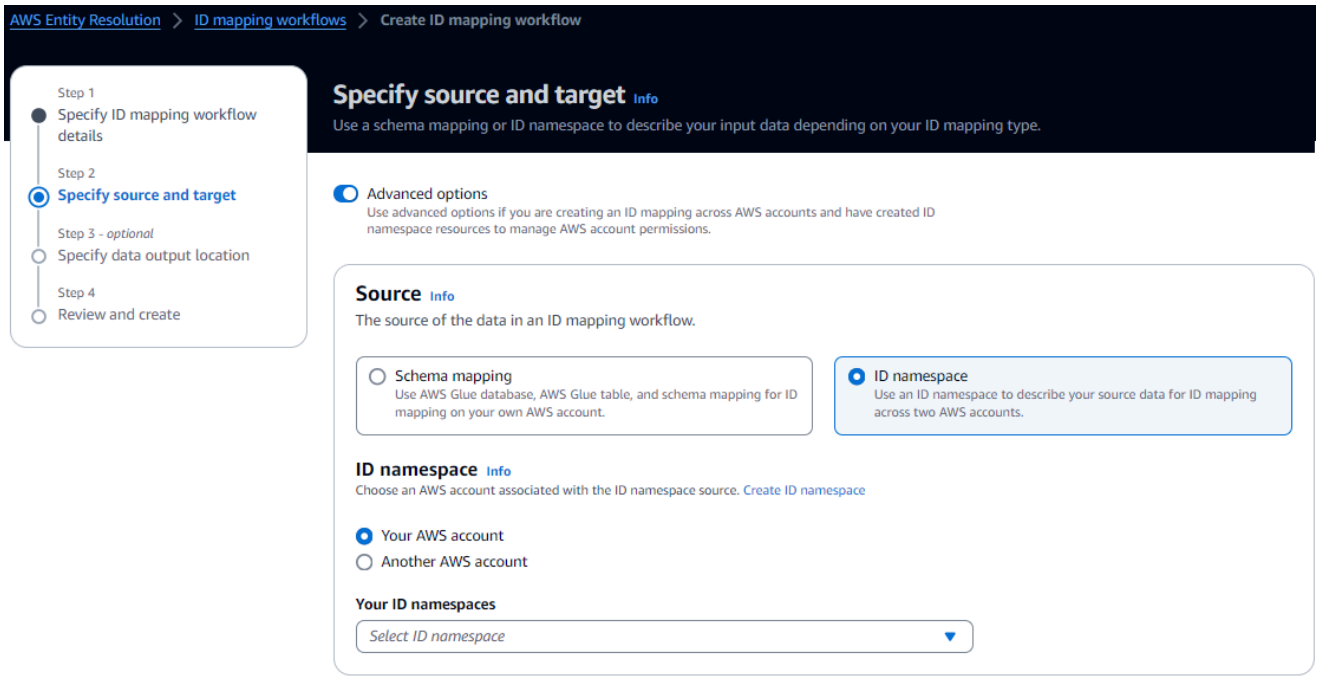
d. (Optional) To enable **Tags** for the resource, choose **Add new tag**, and then enter the **Key** and **Value** pair.

e. Choose **Next**.

5. For **Step 2: Specify source and target**, do the following.

a. Turn on **Advanced options**.

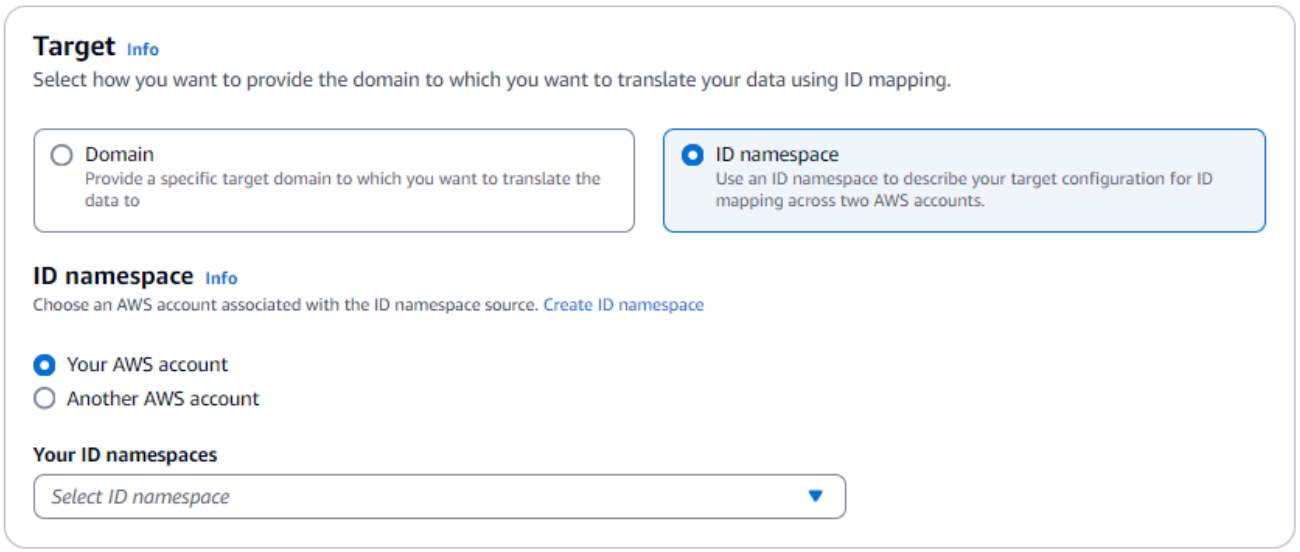
b. For **Source**, choose **ID namespace**.



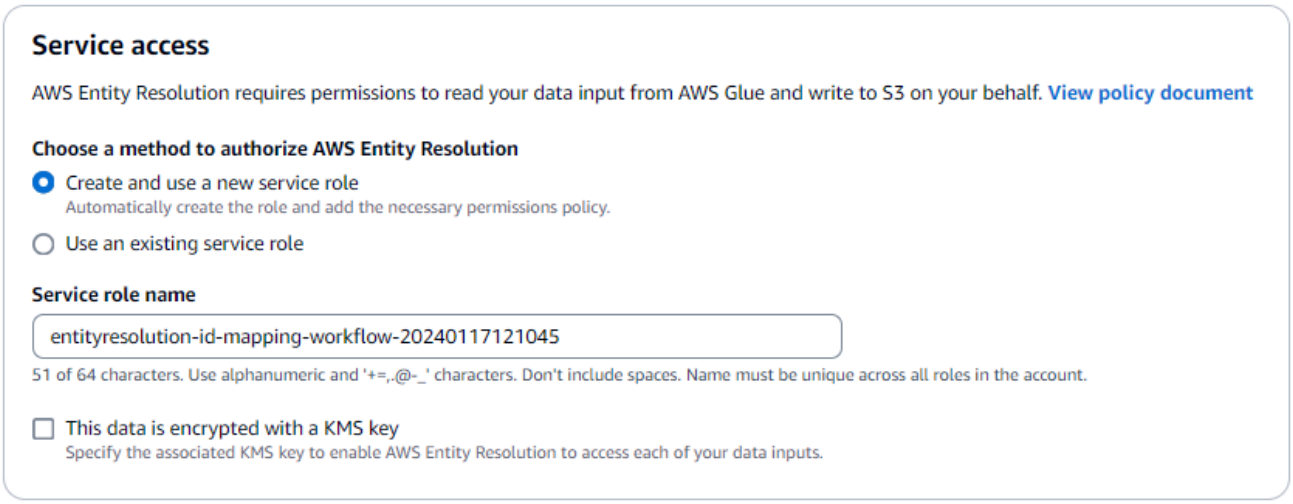
c. For ID namespace, identify where the ID namespace is located, and then take the recommended action.

Location of ID namespace	Recommended action
Your own AWS account	1. Choose **Your AWS account**. 2. Select the ID namespace from the **Your ID namespaces** dropdown list.
Someone else's AWS account	1. Choose **Another AWS account**. 2. Enter the **ID namespace ARN**.

d. For **Target**, choose **ID namespace**.



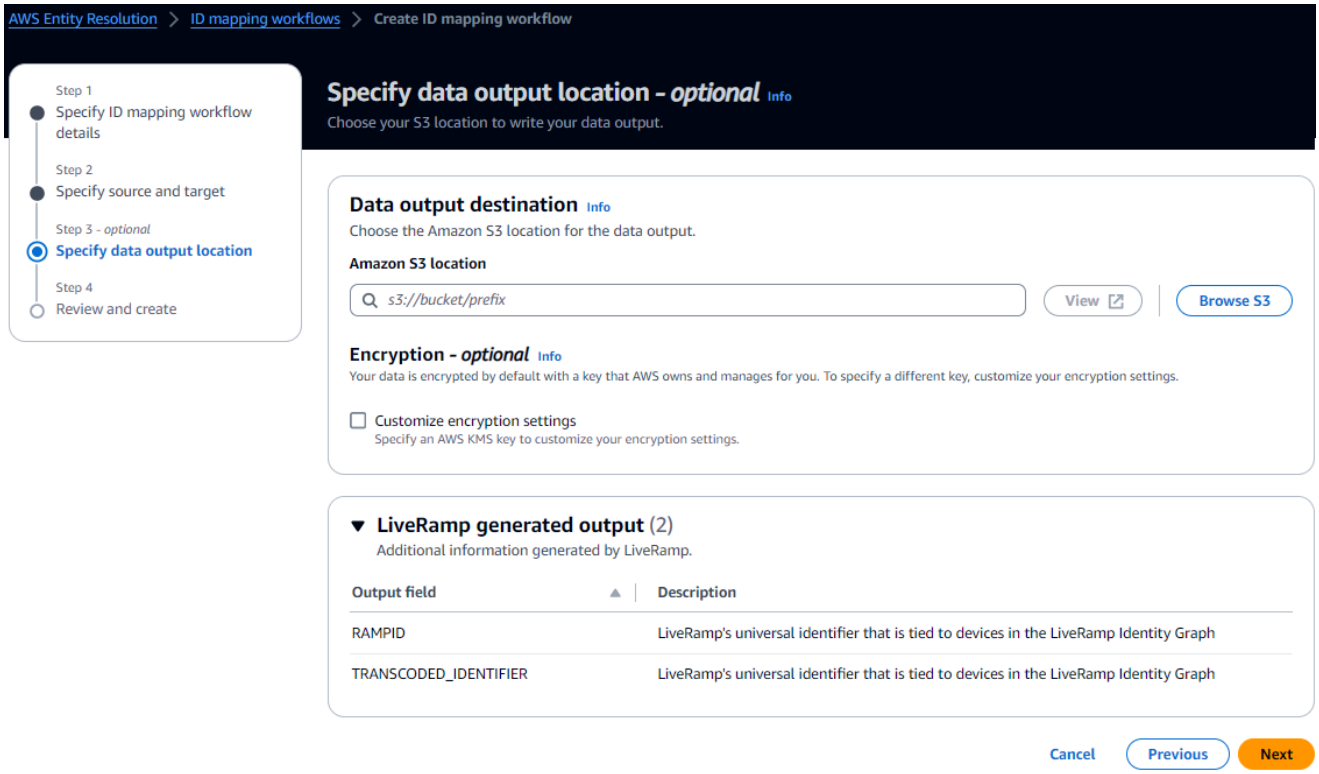
e. To specify the **Service access** permissions, choose an option and take the recommended action.



Option	Recommended action
Create and use a new service role	• AWS Entity Resolution creates a service role with the required policy for this table. • The default **Service role name** is `entityresolution-id-mapping-workflow-<timestamp>` . • You must have permissions to create roles and attach policies. • If your input data is encrypted, choose the **This data is encrypted by a KMS key** option. Then, enter an **AWS KMS key** that is used to decrypt your data input.

Option	Recommended action
Use an existing service role	1. Choose an **Existing service role name** from the dropdown list. The list of roles are displayed if you have permissions to list roles. If you don't have permissions to list roles, you can enter the Amazon Resource Name (ARN) of the role that you want to use. If there are no existing service roles, the option to **Use an existing service role** is unavailable. 2. View the service role by choosing the **View in IAM** external link. By default, AWS Entity Resolution doesn't attempt to update the existing role policy to add necessary permissio ns.

6. Choose **Next**.
7. For **Step 3: Specify data output location – *optional***, do the following.
   a. For **Data output destination**, do the following.
      i. Choose the **Amazon S3 location** for the data output.
      ii. For **Encryption**, if you choose to **Customize encryption settings**, then enter the **AWS KMS key** ARN or choose **Create an AWS KMS key**.
   b. View the **LiveRamp generated output**.
   c. Choose **Next**.



8. For **Step 4: Review and create**, do the following.

   a. Review the selections that you made for the previous steps and edit them if necessary.

   b. Choose **Create**.

      A message appears, indicating that the ID mapping workflow has been created.

After you create the ID mapping workflow, you're ready to run an ID mapping workflow.

Running an ID mapping workflow

After you create an ID mapping workflow for one AWS account or create an ID mapping workflow across two AWS accounts, you can run the ID mapping workflow. The ID mapping workflow outputs a CSV file.

To run an ID mapping workflow

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **ID mapping**.

3. Choose the ID mapping workflow.
4. On the ID mapping workflow details page, in the upper right corner, choose **Run**.
5. On the matching workflow details page, on the **Metrics** tab, view the following under **Last job metrics**:

   - The **Job ID**
   - The **Time completed** for the workflow job
   - The **Status** of the matching workflow job: **Queued**, **In progress**, **Completed**, **Failed**
   - The number of **Records processed**
   - The number of **Records not processed**
   - The number of **Input records**

   Under **Job history**, you can also view the job metrics for previously run ID mapping workflow jobs.
6. After the ID mapping workflow job completes (status is **Completed**), choose **Data output**, and then choose your **Amazon S3 location** to view the results.

   After you get your CSV file, you can join the `RAMPID` with the `TRANSCODED_ID`.

Running an ID mapping workflow with a new output destination

After you create an ID mapping workflow for one AWS account or create an ID mapping workflow across two AWS accounts, you can choose a different S3 location to write your data output.

To run an ID mapping workflow with a new output destination

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **ID mapping**.
3. Choose the ID mapping workflow.
4. On the ID mapping workflow details page, in the upper right corner, choose **Run with new output destination** from the **Run workflow** dropdown list.
5. For **Data output destination**, do the following.

a. Choose the **Amazon S3 location** for the data output.

b. For **Encryption**, if you choose to **Customize encryption settings**, then enter the **AWS KMS key** ARN or choose **Create an AWS KMS key**.

6. To specify the **Service access** permissions, choose an option and take the recommended action.

Option	Recommended action
Create and use a new service role	• AWS Entity Resolution creates a service role with the required policy for this table. • The default **Service role name** is `entityresolution-id-mapping-workflow-<timestamp>` . • You must have permissions to create roles and attach policies. • If your input data is encrypted, choose the **This data is encrypted by a KMS key** option. Then, enter an **AWS KMS key** that is used to decrypt your data input.
Use an existing service role	1. Choose an **Existing service role name** from the dropdown list. The list of roles are displayed if you have permissions to list roles. If you don't have permissions to list roles, you can enter the Amazon Resource Name (ARN) of the role that you want to use. If there are no existing service roles, the option to **Use an existing service role** is unavailable. 2. View the service role by choosing the **View in IAM** external link.

Option	Recommended action
	By default, AWS Entity Resolution doesn't attempt to update the existing role policy to add necessary permissions.

7. Choose **Run**.
8. On the matching workflow details page, on the **Metrics** tab, view the following under **Last job metrics**:

   - The **Job ID**
   - The **Time completed** for the workflow job
   - The **Status** of the matching workflow job: **Queued**, **In progress**, **Completed**, **Failed**
   - The number of **Records processed**
   - The number of **Records not processed**
   - The number of **Input records**

   Under **Job history**, you can also view the job metrics for previously run ID mapping workflow jobs.
9. After the ID mapping workflow job completes (status is **Completed**), choose **Data output**, and then choose your **Amazon S3 location** to view the results.

   After you get your CSV file, you can join the `RAMPID` with the `TRANSCODED_ID`.

Editing an ID mapping workflow

Editing the ID mapping workflow allows you to keep your entity resolution capabilities up-to-date and aligned with your evolving business needs over time. You may want to adjust the mapping rules, techniques, and parameters, you can optimize the workflow to provide more accurate and reliable ID matching results. You may also want to add new data sources, expand the types of IDs being mapped, or incorporate additional matching criteria into the workflow. If you identify problems or errors in ID mapping results, editing with workflow can help you diagnose and resolve those issues.

To edit an ID mapping workflow:

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **ID mapping**.
3. Choose the ID mapping workflow.
4. On the ID mapping workflow details page, in the upper right corner, choose **Edit**.
5. On the **Specify ID mapping workflow details** page, make any necessary changes and then choose **Next**.
6. On the **Specify data output** page, make any necessary changes and then choose **Next**.
7. On the **Review and save** page, make any necessary changes and then choose **Save**.

Deleting an ID mapping workflow

If you no longer use an ID mapping workflow, deleting it can help streamline your workflow management. In addition, deleting redundant or less efficient ID mapping workflows that serve similar purposes can help you consolidate your processes.

To delete an ID mapping workflow:

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **ID mapping**.
3. Choose the ID mapping workflow.
4. On the ID mapping workflow details page, in the upper right corner, choose **Delete**.
5. Confirm the deletion and then choose **Delete**.

Adding or updating a resource policy for an ID mapping workflow

A resource policy allows the creator of the ID mapping resource to access your ID mapping workflow resource.

To add or update a resource policy

1. Sign in to the AWS Management Console and open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/.
2. In the left navigation pane, under **Workflows**, choose **ID mapping**.
3. Choose the ID mapping workflow.
4. On the ID mapping workflow details page, choose the **Permissions** tab.
5. In the **Resource policy**, section choose **Edit**.
6. Add or update the policy in the JSON editor.
7. Choose **Save changes**.

Integrate with AWS Entity Resolution as a provider

AWS Entity Resolution third-party provider integrations help customers protect consumer privacy and maintain compliance with data sovereignty laws. Third-party providers, such as LiveRamp and TransUnion, translate consumer identifiers into advertising IDs, such as Ramp IDs and Fabrick IDs. These advertising identifiers are commonly used in advertising and marketing tools, to prevent consumer data from being exported to non-AWS managed systems. This section provides guidance for providers to integrate with AWS Entity Resolution to encode or transcode consumer identifiers into advertising IDs for use in a provider service-based matching workflow.

For more information about the provider services that are currently integrated with AWS Entity Resolution, see Creating a provider service-based matching workflow.

Topics

Requirements

Before integrating as a provider service with AWS Entity Resolution, complete the following requirements.

Topics

List a provider service on AWS Data Exchange

As a third-party provider, you must list your product on the AWS Data Exchange (ADX) Product Catalog. After your product is listed on the AWS Data Exchange Product Catalog, subscribers can subscribe to your product through either a public or private offer.

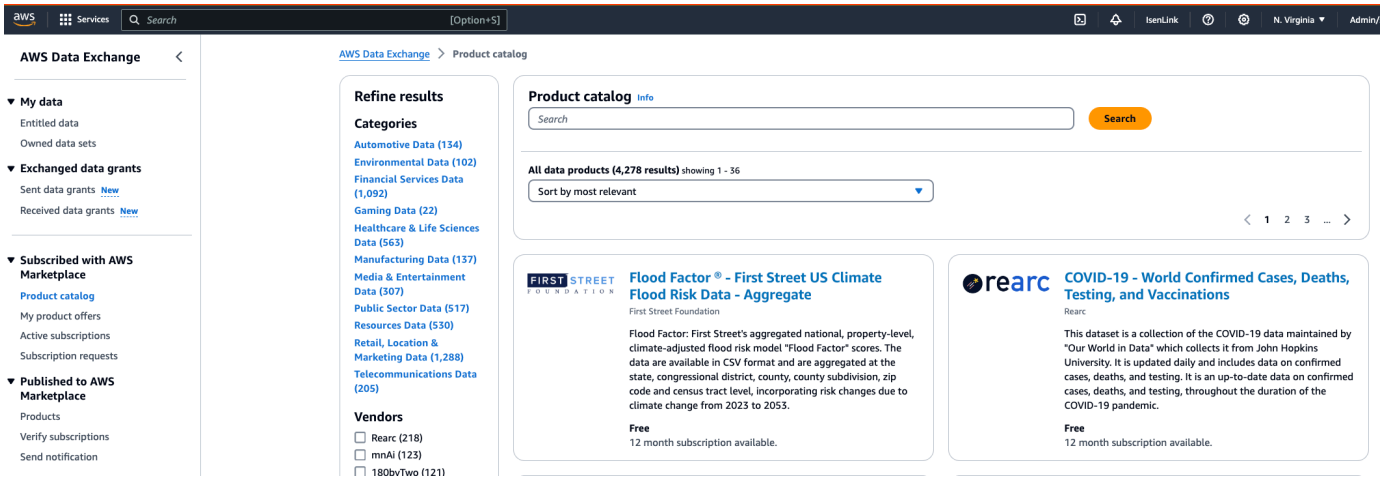
To list a provider service on AWS Data Exchange

1. If you are a new data product provider on AWS Data Exchange, complete the steps in the section titled Getting started as a provider in the *AWS Data Exchange User Guide*.
2. Create a REST API data set and publish a new product that contains APIs on AWS Data Exchange by following the steps in the section titled How to publish a product containing APIs in the *AWS Data Exchange User Guide*. You can complete the process by using either the AWS Data Exchange console or the AWS Command Line Interface.

   If you've set the product visibility **Public**, the public offer is available to all subscribers.

   If you've set the product visibility **Private**, complete the steps in the section titled Create custom offers in the *AWS Data Exchange User Guide*, depending on your use case.

   The following image shows an example of an available product in the AWS Data Exchange Product Catalog.



3. After the product is available on the AWS Data Exchange Product Catalog, the subscriber can subscribe to the product in the following ways.

   - Subscribe the public product.
   - Use a private offer (custom offer) that has been issued by the provider service.
   - Use a Bring Your Own Subscription (BYOS) offer.

   For more information, see Subscribe to and access a product containing APIs in the *AWS Data Exchange User Guide*.

Identify your attributes

Attributes of the input data are the type definitions of the entities to be resolved in a workflow. Some examples of attributes are `FirstName`, `LastName`, `Email`, or `Custom String`.

When you identify your attributes, you should note any requirements or guidelines.

Example Example

The following is an example of validations for identifying provider attributes.

- Either the `FirstName` or `LastName` attribute is mandatory.
- If the `Email` attribute is present, it must be hashed.

As a provider, you must identify the attributes in your provider service product and then communicate these attributes to the AWS Entity Resolution Business Development team at `<aws-entity-resolution-bd@amazon.com>` for additional validation before proceeding.

Request the AWS Entity Resolution OpenAPI specification

AWS Entity Resolution has an OpenAPI specification that you as a provider can use as a handshake that contains the APIs involved in the integration. For more information, see Using the AWS Entity Resolution OpenAPI specification.

To request the OpenAPI definition, contact the AWS Entity Resolution Business Development team at `<aws-entity-resolution-bd@amazon.com>`.

Using the AWS Entity Resolution OpenAPI specification

The OpenAPI specification defines all the protocols associated with AWS Entity Resolution. This specification is necessary to implement the integration.

The OpenAPI definition contains the following API operations:

- `POST AssignIdentities`
- `POST CreateJob`
- `GET GetJob`
- `POST StartJob`

- `POST MapIdentities`
- `GET Schema`

To request the OpenAPI specification, contact the AWS Entity Resolution Business Development team at <aws-entity-resolution-bd@amazon.com>.

The OpenAPI specification support two types of integrations for both encoding and transcoding consumer identifiers batch processing and synchronous processing. After you have obtained the OpenAPI specification, implement the type of processing integration for your use case.

Topics

Batch processing integration

The batch processing integration follows an asynchronous design pattern. After a workflow is initiated on AWS Data Exchange, it submits a job via a provider integration endpoint and then the workflow waits on this job completion by periodically polling for job status. This solution is more desirable for job runs that may take longer and have a lower provider throughput. The provider will intake the dataset location as an Amazon S3 link, which they can process on their end and write the results to a predetermined output S3 location.

The batch processing integration is enabled using three the API definitions. AWS Entity Resolution will call the provider endpoint which is available through AWS Data Exchange in the following order:

1. `POST CreateJob`: This API operation submits the job information to the provider to process. These informations are about the type of job; Encoding or Transcoding, S3 locations, Schema provided by customer, and any additional job properties required.

   This API returns a `JobId`, and the Status for the Job will be one of the following: `PENDING`, `READY`, `IN_PROGRESS`, `COMPLETE`, or `FAILED`.

   Sample request for encoding

```
POST /jobs
```

```
{
  "actionType": "ID_ASSIGNMENT",
  "s3SourceLocation": "string",
  "s3TargetLocation": "string",
  "jobProperties": {
    "assignmentJobProperties": {
      "fieldMappings": [
        {
          "name": "string",
          "type": "NAME"
        }
      ]
    }
  },
  "customerSpecifiedJobProperties": {
    "property1": "string",
    "property2": "string"
  },
  "outputSourceConfiguration": {
    "KMSArn": "string"
  }
}
```

Sample response

```
{
  "jobId": "string",
  "status": "PENDING"
}
```

2. POST `StartJob`: This API lets the provider know to start the job based on the `JobId` provided. This allows the provider to perform any validations needed from `CreateJob` until `StartJob`.

   This API returns a `JobId`, the `Status` for the Job, the `statusMessage`, and `statusCode`.

   Sample request for encoding

```
POST/jobs/{jobId}
{
  "customerSpecifiedJobProperties": {
    "property1": "string",
    "property2": "string"
  }
```

```
}
```

Sample response

```
{
  "jobId": "string",
  "status": "PENDING",
  "statusMessage": "string",
  "statusCode": 200
}
```

3. `GET GetJob`: This API informs AWS Entity Resolution if the job has been completed or any other status.

   This API returns a `JobId`, the `Status` for the Job, the `statusMessage`, and `statusCode`.

   Sample request for encoding

```
GET /jobs/{jobId}
```

   Sample response

```
{
  "jobId": "string",
  "status": "PENDING",
  "statusMessage": "string",
  "statusCode": 200
}
```

The full definition of these APIs are provided in the AWS Entity Resolution OpenAPI specification.

Synchronous processing integration

The synchronous processing solution is more desirable for the providers that have a near real-time response time with real-time response time with higher throughput and higher TPS. This AWS Entity Resolution workflow partitions the dataset and makes multiple API requests in parallel. The AWS Entity Resolution workflow then handles writing the results to desired output location.

This process is enabled using one of the API definitions. AWS Entity Resolution calls the provider endpoint which is available through AWS Data Exchange:

`POST AssignIdentities`: This API sends data to the provider using a `source_id` identifier and `recordFields` associated with that record.

This API returns the `assignedRecords`.

Sample request for encoding

```
POST /assignment
{
  "sourceRecords": [
    {
      "sourceId": "string",
      "recordFields": [
        {
          "name": "string",
          "type": "NAME",
          "value": "string"
        }
      ]
    }
  ]
}
```

Sample response

```
{
  "assignedRecords": [
    {
      "sourceRecord": {
        "sourceId": "string",
        "recordFields": [
          {
            "name": "string",
            "type": "NAME",
            "value": "string"
          }
        ]
      },
      "identity": any
    }
  ]
}
```

The full definition of these APIs are provided in the AWS Entity Resolution OpenAPI specification.

Depending on which approach the provider chooses, AWS Entity Resolution will create a configuration for that the provider that will be used to initiate the encoding or transcoding. In addition, these configurations are available to the customers using the APIs provided by AWS Entity Resolution.

This configuration is accessible using an Amazon Resource Name (ARN), which is derived from where the provider service offering on AWS Data Exchange is hosted, and the type of the provider service. AWS Entity Resolution refers to this ARN as the `providerServiceARN`.

Testing a provider integration

While AWS Entity Resolution hosts data matching services, a provider integration is a crucial third-party component for the end-to-end matching workflow. There are several tests that AWS Entity Resolution has defined for the providers that adds a safeguard when this integration fails. This approach provides an opportunity for providers to monitor their service health according to these end-to-end test cases.

Providers can use their test accounts and their own data to run these end-to-end test cases using the AWS Entity Resolution Software Development Kit (SDK). If there are any issues from providers, AWS Entity Resolution uses the preferred escalation path to escalate the issue. In addition, providers need to implement their own monitoring on the test results. Providers need to share their AWS account IDs that are used to run these tests with AWS Entity Resolution.

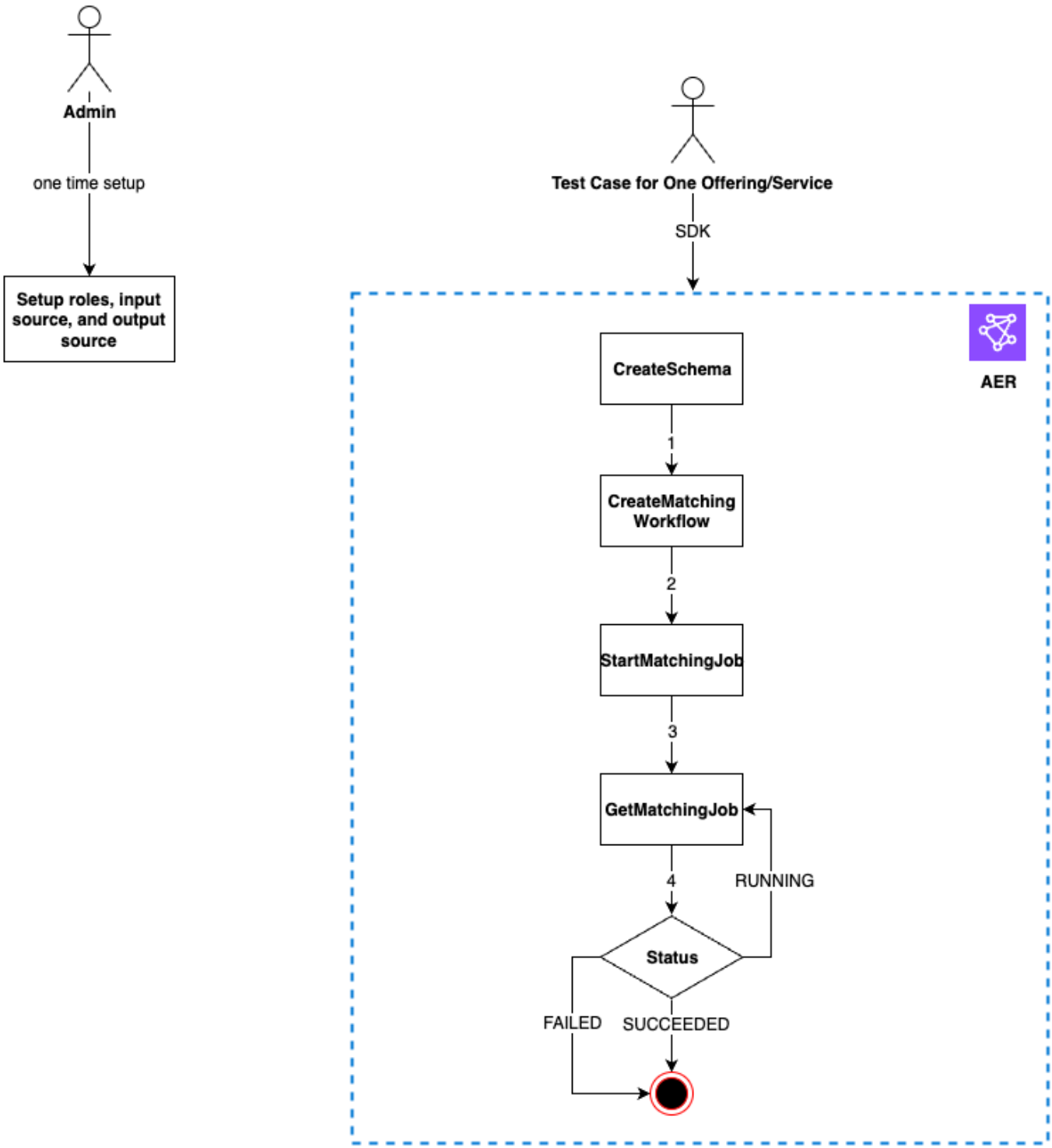
A successful run means a provider can set up their data, use their own service through AWS Entity Resolution, and job status returns **Completed** with no errors. This can be accomplished programmatically using the APIs provided by AWS Entity Resolution.

For example, providers can set up their S3 bucket, input source, roles, schema, and workflows according to their services. After these setups are completed, providers can run these workflows once a day with 200 records to test their service. In this approach, providers use their choice of SDK and run an end-to-end test for their services that are offered through AWS Data Exchange using their test accounts. Providers are expected to run these tests for each of their offerings or services.

> **Note**
>
> Providers need to provide AWS Entity Resolution the AWS account ID (`accountId`) that they use to run these workflows for testing. Additionally, providers need to monitor these

tests and ensure that they pass, meaning that providers need enable notification in case of failures an address the issue accordingly.

The following diagram shows a typical end-to-end workflow test case.



To test a provider integration

1. (*One-time setup*) Set up resources for AWS Entity Resolution by following the procedures in Set up AWS Entity Resolution.

After you have completed the one-time setup procedures, you should have your roles, data, and data source ready. You are now ready to test the provider integration using either the AWS Entity Resolution console or APIs.

2. Test the provider integration using either the AWS Entity Resolution APIs or console.

API

To test a provider integration using the AWS Entity Resolution APIs

1. Create a schema mapping using the CreateSchemaMapping API. For a complete list of supported programming languages, see the See Also section of the CreateSchemaMapping API.

   Schema mapping is the process by which you tell AWS Entity Resolution how to interpret your data for matching. You define the schema of the input data table that you want AWS Entity Resolution to read into a matching workflow.

   When creating a schema mapping, a unique identifier must be designated and assigned to each row of input data that AWS Entity Resolution reads. For example: `Primary_key`, `Row_ID`, `Record_ID`.

   Example Creating a schema mapping for data source containing `id` and `email`

   The following is an example of a schema mapping for a data source that contains `id` and `email`:

```
[
  {
    "fieldName": "id",
    "type": "UNIQUE_ID"
  },
  {
    "fieldName": "email",
    "type": "EMAIL_ADDRESS"
  }
]
```

Example Creating a schema mapping for data source containing `id` and `email` using Java SDK

The following is an example of a schema mapping for a data source that contains `id` and `email` using the Java SDK:

```
EntityResolutionClient.createSchemaMapping(
            CreateSchemaMappingRequest.builder()
                                      .schemaName(<schema-name>)
                                      .mappedInputFields([

 SchemaInputAttribute.builder().fieldName("id").type("UNIQUE_ID").build(),

 SchemaInputAttribute.builder().fieldName("email").type("EMAIL_ADDRESS").build()
                                       ])
                                      .build()

)
```

2. Create a matching workflow using the CreateMatchingWorkflow API. For a complete list of supported programming languages, see the See Also section of the CreateMatchingWorkflow API.

Example Creating a matching workflow using Java SDK

The following is an example of a matching workflow using the Java SDK:

```
EntityResolutionClient.createMatchingWorkflow(
            CreateMatchingWorkflowRequest.builder()
                                         .workflowName(<workflow-name>)
                                         .inputSourceConfig(

 InputSource.builder().inputSourceARN(<glue-inputsource-from-
step1>).schemaName(<schema-name-from-step2>).build()
                                         )

 .outputSourceConfig(OutputSource.builder().outputS3Path(<output-s3-
path>).output(<output-1>, <output-2>, <output-3>).build())

 .resolutionTechniques(ResolutionTechniques.builder()

   .resolutionType(PROVIDER)
```

```
   .providerProperties(ProviderProperties.builder()
                                          .providerServiceArn(<provider-arn>)
                                           .providerConfiguration(<configuration-
depending-on-service>)

 .intermediateSourceConfiguration(<intermedaite-s3-path>)
                                          .build())
   .build()
                                          .roleArn(<role-from-step1>)
                                          .build()

)
```

After the matching workflow is set up, you can run a workflow.

3. Run a matching workflow using the StartMatchingJob API. To run a matching workflow, you must have created a matching workflow using the `CreateMatchingWorkflow` endpoint.

   For a complete list of supported programming languages, see the See Also section of the StartMatchingJob API.

   Example Running a matching workflow using Java SDK

   The following is an example of a running matching workflow using the Java SDK:

```
EntityResolutionClient.startMatchingJob(StartMatchingJobRequest.builder()
                .workflowName(<name-of-workflow-from-step3)
                .build()
)
```

4. Monitor the status of a workflow using the GetMatchingJob API.

   This API returns the status, metrics, and errors (if there are any) that are associated with a job.

Example Monitoring a matching workflow using Java SDK

The following is an example of a monitoring a matching workflow job using the Java SDK:

```
EntityResolutionClient.getMatchingJob(GetMatchingJobRequest.builder()
                .workflowName(<name-of-workflow-from-step3)
                .jobId(jobId-from-startMatchingJob)
                .build()
)
```

The end-to-end test is complete if the workflow has completed successfully.

Console

To test a provider integration using the AWS Entity Resolution console

1. Create a schema mapping by following the steps in Creating a schema mapping.

   Schema mapping is the process by which you tell AWS Entity Resolution how to interpret your data for matching. You define the schema of the input data table that you want AWS Entity Resolution to read into a matching workflow.

   When creating a schema mapping, a unique identifier must be designated and assigned to each row of input data that AWS Entity Resolution reads. For example: `Primary_key`, `Row_ID`, `Record_ID`.

   Example Schema mapping for data source containing `id` and `email`

   The following is an example of a schema mapping for a data source that contains `id` and `email`:

```
[
  {
    "fieldName": "id",
    "type": "UNIQUE_ID"
  },
  {
    "fieldName": "email",
    "type": "EMAIL_ADDRESS"
  }
```

```
]
```

2. Create and run matching workflow by following the steps in Creating a provider service-based matching workflow.

   Creating a matching workflow is the process that you set up to specify the input data to match together and how the matching should be performed. In the provider-based workflow, if an account has a subscription with a provider service through AWS Data Exchange, you can match your known identifiers with your preferred provider. Depending on which provider and which service you are using to perform an end to end test, you can configure your matching workflow accordingly.

   The AWS Entity Resolution console combines the actions of create and run in a single button. After you select **Create and run,** a message appears, indicating that the matching workflow has been created and that the job has started.

3. Monitor the status of the workflow on the **Matching workflows** page.

   The end-to-end test is complete if the workflow has completed successfully (**Job status** is **Completed**).

   On the **Metrics** tab of the matching workflow detail page, you can view the following under **Last job metrics**:

   - The **Job ID**.
   - The **Status** of the matching workflow job: **Queued**, **In progress**, **Completed**, **Failed**
   - The **Time completed** for the workflow job.
   - The number of **Records processed**.
   - The number of **Records not processed**.
   - The **Unique match IDs generated.**
   - The number of **Input records**.

   You can also view the job metrics for matching workflow jobs that have been previously run under the **Job history**.

Security in AWS Entity Resolution

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to AWS Entity Resolution, 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 AWS Entity Resolution. The following topics show you how to configure AWS Entity Resolution to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS Entity Resolution resources.

Topics

Data protection in AWS Entity Resolution

The AWS shared responsibility model applies to data protection in AWS Entity Resolution. 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. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- 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 140-3 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) 140-3.

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 AWS Entity Resolution 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 at rest for AWS Entity Resolution

AWS Entity Resolution provides encryption by default to protect sensitive customer data at rest using AWS owned encryption keys.

AWS owned keys – AWS Entity Resolution uses these keys by default to automatically encrypt personally identifiable data. You can't view, manage, or use AWS owned keys, or audit their use.

However, you don't have to take any action to protect the keys that encrypt your data. For more information, see AWS owned keys in the *AWS Key Management Service Developer Guide*.

Encryption of data at rest by default helps reduce the operational overhead and complexity involved in protecting sensitive data. At the same time, you can use it to build secure applications that meet strict encryption compliance and regulatory requirements.

Alternatively, you can also provide a customer managed KMS key for encryption when you create your matching workflow resource.

Customer managed keys – AWS Entity Resolution supports the use of a symmetric customer managed KMS key that you create, own, and manage to allow encryption of your sensitive data. Because you have full control of this layer of encryption, you can perform such tasks as:

- Establishing and maintaining key policies
- Establishing and maintaining IAM policies and grants
- Enabling and disabling key policies
- Rotating key cryptographic material
- Adding tags
- Creating key aliases
- Scheduling keys for deletion

For more information, see customer managed key in the *AWS Key Management Service Developer Guide*.

For more information about AWS KMS, see What is AWS Key Management Service?

Key management

How AWS Entity Resolution uses grants in AWS KMS

AWS Entity Resolution requires a grant to use your customer managed key. When you create a matching workflow encrypted with a customer managed key, AWS Entity Resolution creates a grant on your behalf by sending a CreateGrant request to AWS KMS. Grants in AWS KMS are used to give AWS Entity Resolution access to a KMS key in a customer account. AWS Entity Resolution requires the grant to use your customer managed key for the following internal operations:

- Send GenerateDataKey requests to AWS KMS to generate data keys encrypted by your customer managed key.
- Send Decrypt requests to AWS KMS to decrypt the encrypted data keys so that they can be used to encrypt your data.

You can revoke access to the grant, or remove the service's access to the customer managed key at any time. If you do, AWS Entity Resolution won't be able to access any of the data encrypted by the customer managed key, which affects operations that are dependent on that data. For example, if you remove the service access to your key through the grant and attempt to start a job for a matching workflow encrypted with a customer key, then the operation would return an `AccessDeniedException` error.

Creating a customer managed key

You can create a symmetric customer managed key by using the AWS Management Console, or the AWS KMS APIs.

To create a symmetric customer managed key

AWS Entity Resolution supports encryption using Symmetric encryption KMS keys. Follow the steps for Creating symmetric customer managed key in the *AWS Key Management Service Developer Guide*.

Key policy statement

Key policies control access to your customer managed key. Every customer managed key must have exactly one key policy, which contains statements that determine who can use the key and how they can use it. When you create your customer managed key, you can specify a key policy. For more information, see Managing access to customer managed keys in the *AWS Key Management Service Developer Guide*.

To use your customer managed key with your AWS Entity Resolution resources, the following API operations must be permitted in the key policy:

- `kms:DescribeKey` – Provides information such as the key ARN, creation date (and deletion date, if applicable), the key state, and the origin and expiration date (if any) of the key material. It includes fields, like `KeySpec`, that help you distinguish different types of KMS keys. It also displays the key usage (encryption, signing, or generating and verifying MACs) and the

algorithms that the KMS key supports. AWS Entity Resolution validates that the `KeySpec` is `SYMMETRIC_DEFAULT` and `KeyUsage` is `ENCRYPT_DECRYPT`.

- `kms:CreateGrant` – Adds a grant to a customer managed key. Grants control access to a specified KMS key, which allows access to grant operations AWS Entity Resolution requires. For more information about Using Grants, see the *AWS Key Management Service Developer Guide*.

This allows AWS Entity Resolution to do the following:

- Call `GenerateDataKey` to generate an encrypted data key and store it, because the data key isn't immediately used to encrypt.
- Call `Decrypt` to use the stored encrypted data key to access encrypted data.
- Set up a retiring principal to allow the service to `RetireGrant`.

The following are policy statement examples you can add for AWS Entity Resolution:

```
{
      "Sid" : "Allow access to principals authorized to use AWS Entity Resolution",
      "Effect" : "Allow",
      "Principal" : {
        "AWS" : "*"
      },
      "Action" : ["kms:DescribeKey","kms:CreateGrant"],
      "Resource" : "*",
      "Condition" : {
        "StringEquals" : {
          "kms:ViaService" : "entityresolution.region.amazonaws.com",
          "kms:CallerAccount" : "111122223333"
        }
      }
}
```

Permissions for users

When you configure a KMS key as the default key for encryption, the default KMS key policy allows any user with access to the required KMS actions to use this KMS key to encrypt or decrypt resources. You must grant users permission to call the following actions in order to use customer managed KMS key encryption:

- `kms:CreateGrant`

- `kms:Decrypt`
- `kms:DescribeKey`
- `kms:GenerateDataKey`

During a `CreateMatchingWorkflow` request, AWS Entity Resolution will send a DescribeKeyand a CreateGrant request to AWS KMS on your behalf. This will require the IAM entity making the `CreateMatchingWorkflow` request with a customer managed KMS key to have the `kms:DescribeKey` permissions on the KMS key policy.

During a `CreateIdMappingWorkflow` and `StartIdMappingJob` request, AWS Entity Resolution will send a DescribeKey and a CreateGrant request to AWS KMS on your behalf. This will require the IAM entity making the `CreateIdMappingWorkflow` and `StartIdMappingJob` request with a customer managed KMS key to have the `kms:DescribeKey` permissions on the KMS key policy. Providers will be able to access the customer managed key to decrypt the data in the AWS Entity Resolution Amazon S3 bucket.

The following are policy statement examples you can add for providers to decrypt the data in the AWS Entity Resolution Amazon S3 bucket:

Replace each *<user input placeholder>* with your own information.

<KMSKeyARN>	AWS KMS Amazon Resource Name.

Similarly, the IAM entity invoking the `StartMatchingJob` API must have `kms:Decrypt` and `kms:GenerateDataKey` permissions on the customer managed KMS key provided in the matching workflow.

For more information about specifying permissions in a policy, see the *AWS Key Management Service Developer Guide*.

For more information about troubleshooting key access, see the *AWS Key Management Service Developer Guide*.

Specifying a customer managed key for AWS Entity Resolution

You can specify a customer managed key as a second layer encryption for the following resources:

Matching workflow – When you create a matching workflow resource, you can specify the data key by entering a **KMSArn**, which AWS Entity Resolution uses to encrypt the identifiable personal data stored by the resource.

KMSArn – Enter a key ARN, which is a key identifier for an AWS KMS customer managed key.

You can specify a customer managed key as a second layer encryption for the following resources if you are creating or running an ID mapping workflow across two AWS accounts:

ID mapping workflow or Start ID mapping workflow – When you create a ID mapping workflow resource or start an ID mapping workflow job, you can specify the data key by entering a **KMSArn**, which AWS Entity Resolution uses to encrypt the identifiable personal data stored by the resource.

KMSArn – Enter a key ARN, which is a key identifier for an AWS KMS customer managed key.

Monitoring your encryption keys for AWS Entity Resolution Service

When you use an AWS KMS customer managed key with your AWS Entity Resolution Service resources, you can use AWS CloudTrail or Amazon CloudWatch Logs to track requests that AWS Entity Resolution sends to AWS KMS.

The following examples are AWS CloudTrail events for `CreateGrant`, `GenerateDataKey`, `Decrypt`, and `DescribeKey` to monitor AWS KMS operations called by AWS Entity Resolution to access data encrypted by your customer managed key:

Topics

- CreateGrant
- DescribeKey
- GenerateDataKey
- Decrypt

CreateGrant

When you use an AWS KMS customer managed key to encrypt your matching workflow resource, AWS Entity Resolution sends a `CreateGrant` request on your behalf to access the KMS key in your AWS account. The grant that AWS Entity Resolution creates are specific to the resource associated with the AWS KMS customer managed key. In addition, AWS Entity Resolution uses the `RetireGrant` operation to remove a grant when you delete a resource.

The following example event records the `CreateGrant` operation:


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-04-22T17:02:00Z"
            }
        },
        "invokedBy": "entityresolution.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:07:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "retiringPrincipal": "entityresolution.region.amazonaws.com",
        "operations": [
            "GenerateDataKey",
            "Decrypt",
        ],
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "granteePrincipal": "entityresolution.region.amazonaws.com"
    },
    "responseElements": {
```

```
        "grantId":
 "0ab0ac0d0b000f00ea00cc0a0e00fc00bce000c000f0000000c0bc0a0000aaafSAMPLE",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
    },
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```

DescribeKey

AWS Entity Resolution uses the `DescribeKey` operation to verify if the AWS KMS customer managed key associated with your matching resource exists in the account and Region.

The following example event records the `DescribeKey` operation.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE3",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAIGDTESTANDEXAMPLE:Sampleuser01",
                "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
                "accountId": "111122223333",
```

```
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-04-22T17:02:00Z"
            }
        },
        "invokedBy": "entityresolution.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:07:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "keyId": "00dd0db0-0000-0000-ac00-b0c000SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333"
}
```


GenerateDataKey

When you enable an AWS KMS customer managed key for your matching workflow resource, AWS Entity Resolution sends a `GenerateDataKey` request through Amazon Simple Storage Service (Amazon S3) to AWS KMS that specifies the AWS KMS customer managed key for the resource.

The following example event records the `GenerateDataKey` operation.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "s3.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:07:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "keySpec": "AES_256",
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "57f5dbee-16da-413e-979f-2c4c6663475e"
}
```

Decrypt

When you enable an AWS KMS customer managed key for your matching workflow resource, AWS Entity Resolution sends a `Decrypt` request through Amazon Simple Storage Service (Amazon S3) to AWS KMS that specifies the AWS KMS customer managed key for the resource.

The following example event records the `Decrypt` operation.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "s3.amazonaws.com"
    },
    "eventTime": "2021-04-22T17:10:51Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "172.12.34.56",
    "userAgent": "ExampleDesktop/1.0 (V1; OS)",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE",
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "eventID": "ff000af-00eb-00ce-0e00-ea000fb0fba0SAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-123456SAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "111122223333",
    "sharedEventID": "dc129381-1d94-49bd-b522-f56a3482d088"
}
```

Considerations

AWS Entity Resolution doesn't support updating a matching workflow with a new customer managed KMS key. In such cases, you can create a new workflow with the customer managed KMS key.

Learn more

The following resources provide more information about data encryption at rest.

For more information about AWS Key Management Service basic concepts, see the *AWS Key Management Service Developer Guide*.

For more information about Security best practices for AWS Key Management Service, see the *AWS Key Management Service Developer Guide*.

Access AWS Entity Resolution using an interface endpoint (AWS PrivateLink)

You can use AWS PrivateLink to create a private connection between your VPC and AWS Entity Resolution. You can access AWS Entity Resolution as if it were in your VPC, without the use of an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC don't need public IP addresses to access AWS Entity Resolution.

You establish this private connection by creating an *interface endpoint*, powered by AWS PrivateLink. We create an endpoint network interface in each subnet that you enable for the interface endpoint. These are requester-managed network interfaces that serve as the entry point for traffic destined for AWS Entity Resolution.

For more information, see Access AWS services through AWS PrivateLink in the *AWS PrivateLink Guide*.

Considerations for AWS Entity Resolution

Before you set up an interface endpoint for AWS Entity Resolution, review Considerations in the *AWS PrivateLink Guide*.

AWS Entity Resolution supports making calls to all of its API actions through the interface endpoint.

VPC endpoint policies are supported for AWS Entity Resolution. By default, full access to AWS Entity Resolution is allowed through the interface endpoint. Alternatively, you can associate a security group with the endpoint network interfaces to control traffic to AWS Entity Resolution through the interface endpoint.

Create an interface endpoint for AWS Entity Resolution

You can create an interface endpoint for AWS Entity Resolution using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Create an interface endpoint in the *AWS PrivateLink Guide*.

Create an interface endpoint for AWS Entity Resolution using the following service name:

```
com.amazonaws.region.entityresolution
```

If you enable private DNS for the interface endpoint, you can make API requests to AWS Entity Resolution using its default Regional DNS name. For example, `entityresolution.us-east-1.amazonaws.com`.

Create an endpoint policy for your interface endpoint

An endpoint policy is an IAM resource that you can attach to an interface endpoint. The default endpoint policy allows full access to AWS Entity Resolution through the interface endpoint. To control the access allowed to AWS Entity Resolution from your VPC, attach a custom endpoint policy to the interface endpoint.

An endpoint policy specifies the following information:

- The principals that can perform actions (AWS accounts, IAM users, and IAM roles).
- The actions that can be performed.
- The resources on which the actions can be performed.

For more information, see Control access to services using endpoint policies in the *AWS PrivateLink Guide*.

Example: VPC endpoint policy for AWS Entity Resolution actions

The following is an example of a custom endpoint policy. When you attach this policy to your interface endpoint, it grants access to the listed AWS Entity Resolution actions for all principals on all resources.

```
{
   "Statement": [
      {
         "Principal": "*",
```

```
            "Effect": "Allow",
            "Action": [
               "entityresolution:CreateMatchingWorkflow",
               "entityresolution:StartMatchingJob",
               "entityresolution:GetMatchingJob"
            ],
            "Resource":"*"
         }
      ]
}
```

Identity and access management for AWS Entity Resolution

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 AWS Entity Resolution resources. IAM is an AWS service that you can use with no additional charge.

> **Note**
>
> AWS Entity Resolution supports cross account policies. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in AWS Entity Resolution.

Service user – If you use the AWS Entity Resolution service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS Entity Resolution 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 AWS Entity Resolution, see Troubleshooting AWS Entity Resolution identity and access.

Service administrator – If you're in charge of AWS Entity Resolution resources at your company, you probably have full access to AWS Entity Resolution. It's your job to determine which AWS Entity Resolution 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 AWS Entity Resolution, see How AWS Entity Resolution 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 AWS Entity Resolution. To view example AWS Entity Resolution identity-based policies that you can use in IAM, see Identity-based policy examples for AWS Entity Resolution.

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 AWS Signature Version 4 for 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 AWS Multi-factor authentication in IAM 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 Use cases for IAM users 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. To temporarily assume an IAM role in the AWS Management Console, you can switch from a user to an IAM role (console). 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 Methods to assume a role 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 Create a role for a third-party identity provider (federation) 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 Cross account resource access in IAM 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 Create 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 Use an IAM role to grant permissions to applications running on Amazon EC2 instances 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 Define custom IAM permissions with customer managed 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 Choose 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 Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – RCPs are JSON policies that you can use to set the maximum available permissions for resources in your accounts without updating the IAM policies attached to each resource that you own. The RCP limits permissions for resources in member accounts and can impact the effective permissions for identities, including the AWS account root user, regardless of whether they belong to your organization. For more information about

Organizations and RCPs, including a list of AWS services that support RCPs, see Resource control policies (RCPs) 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 AWS Entity Resolution works with IAM

Before you use IAM to manage access to AWS Entity Resolution, learn what IAM features are available to use with AWS Entity Resolution.

IAM features you can use with AWS Entity Resolution

IAM feature	AWS Entity Resolution support
Identity-based policies	Yes
Resource-based policies	Yes
Policy actions	Yes
Policy resources	Yes
Policy condition keys	Yes
ACLs	No
ABAC (tags in policies)	Partial
Temporary credentials	Yes

IAM feature	AWS Entity Resolution support
Forward access sessions (FAS)	Yes
Service roles	Yes
Service-linked roles	No

To get a high-level view of how AWS Entity Resolution 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 AWS Entity Resolution

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 Define custom IAM permissions with customer managed 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 AWS Entity Resolution

To view examples of AWS Entity Resolution identity-based policies, see Identity-based policy examples for AWS Entity Resolution.

Resource-based policies within AWS Entity Resolution

Supports resource-based policies: Yes

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 Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for AWS Entity Resolution

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 AWS Entity Resolution actions, see Actions Defined by AWS Entity Resolution in the *Service Authorization Reference*.

Policy actions in AWS Entity Resolution use the following prefix before the action:

```
entityresolution
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "entityresolution:action1",
```

```
      "entityresolution:action2"
         ]
```

To view examples of AWS Entity Resolution identity-based policies, see Identity-based policy examples for AWS Entity Resolution.

Policy resources for AWS Entity Resolution

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 AWS Entity Resolution resource types and their ARNs, see Resources Defined by AWS Entity Resolution in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions Defined by AWS Entity Resolution.

To view examples of AWS Entity Resolution identity-based policies, see Identity-based policy examples for AWS Entity Resolution.

Policy condition keys for AWS Entity Resolution

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 AWS Entity Resolution condition keys, see Condition Keys for AWS Entity Resolution in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions Defined by AWS Entity Resolution.

To view examples of AWS Entity Resolution identity-based policies, see Identity-based policy examples for AWS Entity Resolution.

ACLs in AWS Entity Resolution

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 AWS Entity Resolution

Supports ABAC (tags in policies): Partial

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 Define permissions with ABAC authorization 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*.

Using temporary credentials with AWS Entity Resolution

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 Switch from a user to an IAM 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.

Forward access sessions for AWS Entity Resolution

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 AWS Entity Resolution

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 Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break AWS Entity Resolution functionality. Edit service roles only when AWS Entity Resolution provides guidance to do so.

Service-linked roles for AWS Entity Resolution

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.

Identity-based policy examples for AWS Entity Resolution

By default, users and roles don't have permission to create or modify AWS Entity Resolution 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 Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by AWS Entity Resolution, including the format of the ARNs for each of the resource types, see Actions, Resources, and Condition Keys for AWS Entity Resolution in the *Service Authorization Reference*.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete AWS Entity Resolution 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 Validate policies with IAM Access Analyzer 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 Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the AWS Entity Resolution console

To access the AWS Entity Resolution console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the AWS Entity Resolution 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.

To ensure that users and roles can still use the AWS Entity Resolution console, also attach the AWS Entity Resolution `ConsoleAccess` or `ReadOnly` AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

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": "*"
        }
    ]
}
```

AWS managed policies for AWS Entity Resolution

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AWSEntityResolutionConsoleFullAccess

You can attach the `AWSEntityResolutionConsoleFullAccess` policy to your IAM identities.

This policy grants full access to AWS Entity Resolution endpoints and resources.

This policy also allows certain read access to related AWS services like S3, AWS Glue, Tagging and AWS KMS so that the console can display choices and use the selected ones to perform entity resolution actions. Some resources are narrowed down to contain the service name `entityresolution`.

Because AWS Entity Resolution relies on a passed role to perform actions on related AWS resources, this policy also grants the permissions to select and pass a desired role.

Permissions details

This policy includes the following permissions.

- `EntityResolutionAccess` – Allows principals full access to AWS Entity Resolution endpoints and resources.
- `GlueSourcesConsoleDisplay` – Grants the access to list AWS Glue tables as data source options and import table schema of a data source for user experience.
- `S3BucketsConsoleDisplay` – Grants the access to list all S3 buckets as data source options.
- `S3SourcesConsoleDisplay` – Grants the access to display S3 buckets as data source options.
- `TaggingConsoleDisplay` – Grants the access to read tagging keys and values.
- `KMSConsoleDisplay` – Grants the access to describe keys and list aliases in AWS Key Management Service to decrypt and encrypt data sources.
- `ListRolesToPickForPassing` – Grants the access to list all roles so that the user can pick the role to be passed.
- `PassRoleToEntityResolutionService` – Grants the access to pass a narrowed down role to the AWS Entity Resolution service.

- `ManageEventBridgeRules` – Grants the access to create, update, and delete the Amazon EventBridge rule for getting S3 notifications.
- `ADXReadAccess` – Grants the access to AWS Data Exchange to verify if the customer has an entitlement or a subscription.

To view the permissions for this policy, see AWSEntityResolutionConsoleFullAccess in the *AWS Managed Policy Reference*.

AWS managed policy: AWSEntityResolutionConsoleReadOnlyAccess

You can attach `AWSEntityResolutionConsoleReadOnlyAccess` to your IAM entities.

This policy grants read-only access to AWS Entity Resolution endpoints and resources.

Permissions details

This policy includes the following permissions.

- `EntityResolutionRead` – Allows principals read-only access to AWS Entity Resolution endpoints and resources.

To view the permissions for this policy, see AWSEntityResolutionConsoleReadOnlyAccess in the *AWS Managed Policy Reference*.

AWS Entity Resolution updates to AWS managed policies

View details about updates to AWS managed policies for AWS Entity Resolution since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the AWS Entity Resolution Document history page.

Change	Description	Date
`AWSEntityResolutionConsoleFullAccess` – Update to existing policy	Added `ADXReadAccess` and `ManageEventBridgeRules` to enable the provider services option in the matching workflow.	October 16, 2023

Change	Description	Date
AWS Entity Resolution started tracking changes	AWS Entity Resolution started tracking changes for its AWS managed policies.	August 18, 2023

Troubleshooting AWS Entity Resolution identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS Entity Resolution and IAM.

Topics

I am not authorized to perform an action in AWS Entity Resolution

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but does not have the fictional `entityresolution:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 entityresolution:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *`my-example-widget`* resource using the `entityresolution:`*`GetWidget`* action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to AWS Entity Resolution.

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 AWS Entity Resolution. 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 AWS Entity Resolution 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 AWS Entity Resolution supports these features, see How AWS Entity Resolution 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 Cross account resource access in IAM in the *IAM User Guide*.

Compliance validation for AWS Entity Resolution

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. 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 AWS services 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 Compliance & Governance – These solution implementation guides discuss architectural considerations and provide steps for deploying security and compliance features.
- HIPAA Eligible Services Reference – Lists HIPAA eligible services. Not all AWS services are HIPAA eligible.
- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- 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. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- Amazon GuardDuty – This AWS service detects potential threats to your AWS accounts, workloads, containers, and data by monitoring your environment for suspicious and malicious activities. GuardDuty can help you address various compliance requirements, like PCI DSS, by meeting intrusion detection requirements mandated by certain compliance frameworks.

- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

AWS Entity Resolution compliance best practices

This section provides best practices and recommendations for compliance when you use AWS Entity Resolution.

Payment Card Industry Data Security Standards (PCI DSS)

AWS Entity Resolution supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being compliant with the Payment Card Industry (PCI) Data Security Standard (DSS). For more information about PCI DSS, including how to request a copy of the AWS PCI Compliance Package, see PCI DSS Level 1.

System and Organization Controls (SOC)

AWS Entity Resolution is compliant with System and Organization Controls (SOC) measures, including SOC 1, SOC 2, and SOC 3. SOC reports are independent, third-party examination reports that demonstrate how AWS achieves key compliance controls and objectives. These audits ensure that the appropriate safeguards and procedures are in place to protect against risks that might affect the security, confidentiality, and availability of customer and company data. The results of these third-party audits are available on the AWS SOC Compliance website, where you can view the published reports to get more information about the controls that support AWS operations and compliance.

Resilience in AWS Entity Resolution

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, AWS Entity Resolution offers several features to help support your data resiliency and backup needs.

Monitoring AWS Entity Resolution

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Entity Resolution and your other AWS solutions. AWS provides the following monitoring tools to watch AWS Entity Resolution, report when something is wrong, and take automatic actions when appropriate:

- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP discuss from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.
- *Amazon CloudWatch Logs* enables you to check, store, and access your logs from Amazon EC2 instances, CloudTrail, and other sources. CloudWatch Logs can check information in the log files and tell you when certain thresholds are met. You can also archive your log data in highly durable storage. For more information, see the Amazon CloudWatch Logs User Guide.

Topics

Logging AWS Entity Resolution API calls using AWS CloudTrail

AWS Entity Resolution is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Entity Resolution. CloudTrail captures all API calls for AWS Entity Resolution as events. The calls captured include calls from the AWS Entity Resolution console and code calls to the AWS Entity Resolution API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Entity Resolution. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to AWS Entity Resolution, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

AWS Entity Resolution information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS Entity Resolution, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing events with CloudTrail Event history.

For an ongoing record of events in your AWS account, including events for AWS Entity Resolution, 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 AWS 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 the following:

- 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

All AWS Entity Resolution actions are logged by CloudTrail and are documented in the AWS Entity Resolution API Reference.

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 AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

Understanding AWS Entity Resolution 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 aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

Monitoring and logging workflows using Amazon CloudWatch Logs

AWS Entity Resolution provides comprehensive logging capabilities that help you check and analyze your matching and ID mapping workflows. Through integration with Amazon CloudWatch Logs, you can capture detailed information about workflow execution, including event types, timestamps, processing statistics, and error counts. You can choose to deliver these logs to CloudWatch Logs, Amazon S3, or Amazon Data Firehose destinations. By analyzing these logs, you can evaluate service performance, troubleshoot issues, gain insights into your customer base, and better understand your AWS Entity Resolution usage and billing. While logging is disabled by default, you can enable it for both new and existing workflows through the console or API.

Standard Amazon CloudWatch vending charges apply when you enable logging for AWS Entity Resolution workflows, including costs associated with log ingestion, storage, and analysis; for detailed pricing information, visit the CloudWatch pricing page..

Topics

Setting up log delivery

This section will explain the necessary permissions required to use AWS Entity Resolution logging and how to enable log delivery using the console and APIs.

Topics

Permissions

AWS Entity Resolution uses CloudWatch vended logs to deliver workflow logging. To deliver workflow logs, you need permissions to the logging destination that you specify.

To see the required permissions for each logging destination, choose from the following AWS services in the *Amazon CloudWatch Logs User Guide*.

- Amazon CloudWatch Logs
- Amazon Simple Storage Service (Amazon S3)
- Amazon Data Firehose

To create, view, or change logging configuration in AWS Entity Resolution, you must have the required permissions. Your IAM role must include the following minimum permissions to manage workflow logging in the AWS Entity Resolution console.

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowLogDeliveryActionsConsoleCWL",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:*"
            ]
        },
        {
            "Sid": "AllowLogDeliveryActionsConsoleS3",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
```

```
                "arn:aws:s3:::*"
            ]
        },
        {
            "Sid": "AllowLogDeliveryActionsConsoleFH",
            "Effect": "Allow",
            "Action": [
                "firehose:ListDeliveryStreams",
                "firehose:DescribeDeliveryStream"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

For more information about permissions to manage workflow logging, see Enable logging from AWS services in the *Amazon CloudWatch Logs User Guide*.

Enabling logging for a new workflow (console)

After you set up permissions to the logging destination, you can enable logging for a new workflow in AWS Entity Resolution using the console.

To enable logging for a new workflow (console)

1. Open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/home.
2. Under **Workflows**, select either **Matching** workflows or **ID mapping** workflows.
3. Follow the steps to create one of the following workflows:

   - Rule-based matching workflow
   - Machine learning-based matching workflow
   - Provider service-based matching workflow
   - ID mapping workflow for one account
   - ID mapping workflow across two accounts
4. For **Step 1 Specify Matching workflow details**, for **Log deliveries – EntityResolution Workflow Logs**, choose **Add**.

- Choose one of the following logging destinations.
  - **To Amazon CloudWatch Logs**
  - **To Amazon S3**
  - **To Amazon Data Firehose**

> **Tip**
>
> If you choose Amazon S3 or Firehose, you can deliver your logs to a **Cross account** or **In current account**.
> To enable cross-account delivery, both AWS accounts must have the required permissions. For more information, see the Cross-account delivery example in the *Amazon CloudWatch Logs User Guide*.

5. For the **Destination log group**, the log groups that are prefixed with **'/aws/vendedlogs/'** are created automatically. If you are using other log groups, you them before setting up a log delivery. For more information, see Working with log groups and log streams in the *Amazon CloudWatch Logs User Guide*.
6. For **More settings - optional**, choose the following:
   a. For **Field selection**, select the log fields to include in each log record.
   b. (CloudWatch Logs) For **Output format**, choose the output format for the log.
   c. For **Field delimiter**, choose how to separate each log field.
   d. (Amazon S3) For **Suffix**, specify the suffix path to partition your data.
   e. (Amazon S3) For **Hive-compatible**, choose **Enable** if you want to use Hive-compatible S3 paths.
7. To create another log destination, choose **Add** and repeat steps 4 – 6.
8. Complete the remaining steps to set up and run the workflow.
9. After the workflow jobs completes, check the workflow logs in the log delivery destination you specified.

Enabling logging for a new workflow (API)

After you set up permissions to the logging destination, you can enable logging for a new workflow in AWS Entity Resolution using the Amazon CloudWatch Logs APIs.

To enable logging for a new workflow (API)

1. After you a create a workflow in the AWS Entity Resolution console, get the Amazon Resource Name (ARN) of the workflow.

   You can find the ARN from the workflow page in the AWS Entity Resolution console or you call the `GetMatchingWorkflow` or `GetIdMappingWorkflow` API operation.

   A workflow ARN follows this format:

   ```
   arn:(aws|aws-us-gov|aws-cn):entityresolution:[a-z]{2}-[a-z]{1,10}-
   [0-9]:[0-9]{12}:(matchingworkflow/[a-zA-Z_0-9-]{1,255})
   ```

   An ID mapping ARN follows this format:

   ```
   arn:(aws|aws-us-gov|aws-cn):entityresolution:[a-z]{2}-[a-z]{1,10}-
   [0-9]:[0-9]{12}:(idmappingworkflow/[a-zA-Z_0-9-]{1,255})
   ```

   For more information, see GetMatchingWorkflow or GetIdMappingWorkflow in the *AWS Entity Resolution API Reference*.
2. Use the CloudWatch Logs `PutDeliverySource` API operation to create a delivery source for the workflow logs.

   For more information, see PutDeliverySource in the Amazon CloudWatch Logs API Reference.

   a. Pass the `resourceArn`.

   b. For `logType`, the type of logs that are collected are `WORKFLOW_LOGS`:

   Example

   Example `PutDeliverySource` API operation

   ```
   {
       "logType": "WORKFLOW_LOGS",
       "name": "my-delivery-source",
   ```

```
    "resourceArn": "arn:aws:entityresolution:region:accoungId:matchingworkflow/
XXXWorkflow"
}
```

3. Use the `PutDeliveryDestination` API operation to configure where to store your logs.

   You can choose either CloudWatch Logs, Amazon S3, or Firehose as the destination. You must specify the ARN of one of the destination options for where your logs will be stored.

   For more information, see PutDeliveryDestination in the Amazon CloudWatch Logs API Reference.

   Example

   Example `PutDeliveryDestination` API operation

```
{
   "delivery-destination-configuration": {
      "destinationResourceArn": "arn:aws:logs:region:accountId:log-group:my-log-
group"
   },
   "name": "my-delivery-destination",
   "outputFormat": "json",
   }
}
```

   > **Note**
   >
   > If you're delivering logs cross-account, you must use the **PutDeliveryDestinationPolicy** API to assign an AWS Identity and Access Management (IAM) policy to the destination account. The IAM policy allows delivery from one account to another account.

4. Use the `CreateDelivery` API operation to link the delivery source to the destination that you created in the earlier steps. This API operation associates the delivery source with the end destination.

   For more information, see PutDeliveryDestination in the Amazon CloudWatch Logs API Reference.

Example

Example `CreateDelivery` API operation

```
{
   "delivery-destination-arn": "arn:aws:logs:region:accountId:log-group:my-log-
group",
   "delivery-source-name": "my-delivery-source",
   "tags": {
      "string" : "string"
   }
}
```

5. Run the workflow.
6. After the workflow jobs completes, check the workflow logs in the log delivery destination you specified.

Enabling logging for an existing workflow (console)

After you set up permissions to the logging destination, you can enable logging for an existing workflow in AWS Entity Resolution using the **Log deliveries** tab on the console.

To enable logging for an existing workflow using the Log deliveries tab (console)

1. Open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/home.
2. Under **Workflows**, select either **Matching** workflows or **ID mapping** workflows, and then select your existing workflow.
3. On the **Log deliveries** tab, under **Log delivery**, select **Add**, and then choose one of the following logging destinations.
   - To Amazon CloudWatch Logs
   - To Amazon S3
     - Cross account
     - In current account
   - To Amazon Data Firehose
     - Cross account

- In current account

> **Tip**
>
> If you choose Amazon S3 or Firehose, you can deliver your logs to a **Cross account** or **In current account**.
> To enable cross-account delivery, both AWS accounts must have the required permissions. For more information, see the Cross-account delivery example in the *Amazon CloudWatch Logs User Guide*.

4. In the modal, do the following, depending on the type of Log delivery you chose.

   a. View the **Log type**: **WORKFLOW_LOGS**.

      The **Log type** can't be changed.

   b. (CloudWatch Logs) For the **Destination log group**, the log groups that are prefixed with **'/aws/vendedlogs/'** are created automatically. If you are using other log groups, you them before setting up a log delivery. For more information, see Working with log groups and log streams in the *Amazon CloudWatch Logs User Guide*.

      (Amazon S3 in current account) For **Destination S3 bucket**, select a bucket or enter an ARN.

      (Amazon S3 cross account) For **Delivery destination ARN**, enter a delivery destination ARN.

      (Firehose in current account) For **Destination delivery stream**, enter the ARN of the delivery destination resource that was created in another account.

      (Firehose cross account) For **Delivery destination ARN**, enter a delivery destination ARN.

5. For **More settings - optional**, choose the following:

   a. For **Field selection**, select the log fields to include in each log record.

   b. (CloudWatch Logs) For **Output format**, choose the output format for the log.

   c. For **Field delimiter**, choose how to separate each log field.

   d. (Amazon S3) For **Suffix**, specify the suffix path to partition your data.

e. (Amazon S3) For **Hive-compatible**, choose **Enable** if you want to use Hive-compatible S3 paths.

6. Choose **Add**.
7. On the workflow page, choose **Run**.
8. After the workflow jobs completes, check the workflow logs in the log delivery destination you specified.

Disabling logging (console)

You can disable logging for your AWS Entity Resolution workflow at any time in the console.

To disable workflow logging (console)

1. Open the AWS Entity Resolution console at https://console.aws.amazon.com/entityresolution/home.
2. Under **Workflows**, select either **Matching** workflows or **ID mapping** workflows, and then select your workflow.
3. On the **Log deliveries** tab, under **Log delivery**, select the destination, and then choose **Delete**.
4. Review your changes and then navigate to the next step to save your changes.

Reading the logs

Reading Amazon CloudWatch Logs helps you maintain efficient AWS Entity Resolution workflows. Logs give detailed visibility into your workflow execution, including important metrics like the number of records processed and any errors encountered, helping you ensure your data processing is running smoothly. In addition, the logs offer real-time tracking of workflow progression through timestamps and event types, allowing you to quickly identify bottlenecks or issues in your data processing pipeline. The comprehensive error tracking and record count information helps you keep data quality and completeness by showing exactly how many records were processed successfully and if any remained unprocessed.

If you're using CloudWatch Logs as the destination, you can use CloudWatch Logs Insights to read the workflow logs. Typical CloudWatch Logs charges apply. For more information, see Analyzing Log Data with CloudWatch Logs Insights in the *Amazon CloudWatch Logs User Guide*.

Note

Workflow logs can take a few minutes to appear in your destination. If you don't see the logs, wait a few minutes and refresh the page.

The workflow logs consist of a sequence of formatted log records, where each log record represents one workflow. The order of the fields within the log can vary.

```
{
  "resource_arn": "arn:aws:ses:us-east-1:1234567890:mailmanager-ingress-point/inp-
xxxxx",
  "event_type": "JOB_START",
  "event_timestamp": 1728562395042,
  "job_id": "b01eea4678d4423a4b43eeada003f6",
  "workflow_name": "TestWorkflow",
  "workflow_start_time": "2025-03-11 10:19:56",
  "data_procesing_progression": "Matching Job Starts ...",
  "total_records_processed": 1500,
  "total_records_unprocessed": 0,
  "incremental_records_processed": 0,
  "error_message": "sample error that caused workflow failure"
}
```

The following list describes the log record fields, in order:

`resource_arn`

The Amazon Resource Name (ARN) that uniquely identifies the AWS resource being used in the workflow.

`event_type`

The type of event that occurred during the workflow execution. AWS Entity Resolution currently supports:

`JOB_START`

`DATA_PROCESSING_STEP_START`

`DATA_PROCESSING_STEP_END`

`JOB_SUCCESS`

`JOB_FAILURE`

`event_timestamp`

The Unix timestamp indicating when the event occurred during the workflow.

`job_id`

A unique identifier assigned to the specific workflow job execution.

`workflow_name`

The name given to the workflow being executed.

`workflow_start_time`

The date and time when the workflow execution began.

`data_procesing_progression`

A description of the current stage in the data processing workflow. Examples: `"Matching Job Starts"`, `"Loading Step Starts"`, `"ID_Mapping Job Ends Successfully"`.

`total_records_processed`

The total number of records that were successfully processed during the workflow.

`total_records_unprocessed`

The number of records that weren't processed during the workflow execution.

`incremental_records_processed`

The number of new records processed in an incremental workflow update.

`error_message`

The root cause of workflow failure.

Create AWS Entity Resolution resources with AWS CloudFormation

AWS Entity Resolution 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 want (such as AWS::EntityResolution::MatchingWorkflow, AWS::EntityResolution::SchemaMapping, AWS::EntityResolution:IdMappingWorkflow, AWS::EntityResolution::IdNamespace and AWS::EntityResolution::PolicyStatement), and AWS CloudFormation provisions and configures those resources for you.

When you use AWS CloudFormation, you can reuse your template to set up your AWS Entity Resolution resources consistently and repeatedly. Describe your resources once, and then provision the same resources over and over in multiple AWS accounts and Regions.

AWS Entity Resolution and AWS CloudFormation templates

To provision and configure resources for AWS Entity Resolution 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*.

AWS Entity Resolution supports creating AWS::EntityResolution::MatchingWorkflow, AWS::EntityResolution::SchemaMapping, AWS::EntityResolution:IdMappingWorkflow, AWS::EntityResolution::IdNamespace and AWS::EntityResolution::PolicyStatement in AWS CloudFormation. For more information, including examples of JSON and YAML templates for AWS::EntityResolution::MatchingWorkflow, AWS::EntityResolution::SchemaMapping, AWS::EntityResolution:IdMappingWorkflow, AWS::EntityResolution::IdNamespace and AWS::EntityResolution::PolicyStatement, see the AWS Entity Resolution resource type reference in the *AWS CloudFormation User Guide*.

The following templates are available:

- *Matching workflow*

Create a `MatchingWorkflow` object, which stores the configuration of the data processing job to be run.

For more information, see the following topics:

AWS::EntityResolution::MatchingWorkflow in the *AWS CloudFormation User Guide*

CreateMatchingWorkflow in the *AWS Entity Resolution API Reference*

- *Schema mapping*

  Create a schema mapping, which defines the schema of the input customer records table.

  For more information, see the following topics:

  AWS::EntityResolution::SchemaMapping in the *AWS CloudFormation User Guide*

  CreateSchemaMapping in the *AWS Entity Resolution API Reference*

- *ID mapping workflow*

  Create an `IdMappingWorkflow` object, which stores the configuration of the data processing job to run.

  For more information, see the following topics:

  AWS::EntityResolution::IdMappingWorkflow in the *AWS CloudFormation User Guide*

  CreateIdMappingWorkflow in the *AWS Entity Resolution API Reference*

- *ID namespace*

  Create an `IdNamespace` object, which stores the metadata explaining the dataset and how to use it.

  For more information, see the following topics:

  AWS::EntityResolution::IdNamespace in the *AWS CloudFormation User Guide*

  CreateIdNamespace in the *AWS Entity Resolution API Reference*

- *PolicyStatement*

  Create an `PolicyStatement` object.

For more information, see the following topics:

AWS::EntityResolution::PolicyStatement in the *AWS CloudFormation User Guide*

AddPolicyStatement in the *AWS Entity Resolution API Reference*

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

Quotas for AWS Entity Resolution

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, but other quotas can't be increased.

To view the quotas for AWS Entity Resolution, open the Service Quotas console. In the navigation pane, choose **AWS services** and select **AWS Entity Resolution**.

To request a quota increase, see Requesting a Quota Increase in the *Service Quotas User Guide*. If the quota isn't yet available in Service Quotas, use the limit increase form.

Your AWS account has the following quotas related to AWS Entity Resolution.

Name	Default	Adjustable	Description
Concurrent ID mapping jobs	1	No	The maximum number of ID mapping jobs that can be processed concurrently in the current AWS Region.
Concurrent matching jobs	1	No	The maximum number of matching jobs that can be processed concurrently in the current AWS Region.
Concurrent provider service matching jobs	1	No	The maximum number of provider service matching jobs that can be processed concurrently in the current AWS Region.
Data input	20	No	This is the list of input tables that you want to use in a matching workflow. Each input corresponds to a column in your AWS Glue input data table, which contains the column name and additional information that AWS Entity Resolution uses for matching purposes. Inputs must contain a

Name	Default	Adjustable	Description
			Unique ID plus at least one additional input field.
Data output	750	No	This is a list of `OutputAttribute` objects, each of which have the fields **Name** and **Hashed**. Each of these objects represent a column to be included in the AWS Glue output table and whether you want the values in the column to be hashed.
Data schema	25	No	The maximum number of data schema input fields.
ID mapping workflows	10	Yes	The maximum number of ID mapping workflows that you can create in this AWS account in the current AWS Region.
ID namespaces	10	Yes	The maximum number of ID namespaces that you can create in this AWS account in the current AWS Region.
Match IDs	500	No	The maximum number of records that can be consolidated under one MatchID per workload.
Match rule	15	No	For rule-based matching, this is the rule number applied that generated a matched record set. This is part of matching workflow metadata that will be included in output.
Matching workflows	10	Yes	The maximum number of matching workflows.

Name	Default	Adjustable	Description
Rate of `GetMatchId` API requests	50	Yes	The maximum number of `GetCustomerID` API requests per second.
Records per machine learning-based workflow	250M	Yes	The maximum number of records that can be processed by a machine learning-based matching workflow.
Records per rule-based matching workflow	100M	Yes	The maximum number of records that can be processed by a rule-based matching workflow.
Rules per workflow	15	No	The maximum number of rules per matching workflow.
Schema mappings	50	Yes	The maximum number of schema mappings that you can create in this account in the current AWS Region.
Unique match keys per across ruleset	15	No	The maximum number of unique match keys per rule set. A match key instructs AWS Entity Resolution which input fields are to be considered as similar data and which are to be considered as different data. This helps AWS Entity Resolution automatically configure rule-based matching rules and compare similar data stored in different input fields.

API throttling quotas

Resource	Rate limit	Description
Rate of `CreateMatchingWorkflow` requests	5 TPS	Maximum number of `CreateMatchingWorkflow` API calls per second.
Rate of `DeleteMatchingWorkflow` requests	5 TPS	Maximum number of `DeleteMatchingWorkflow` API calls per second.
Rate of `GetMatchingWorkflow` requests	5 TPS	Maximum number of `GetMatchingWorkflow` API calls per second.
Rate of `ListMatchingWorkflows` requests	5 TPS	Maximum number of `ListMatchingWorkflows` API calls per second.
Rate of `UpdateMatchingWorkflow` requests	5 TPS	Maximum number of `UpdateMatchingWorkflow` API calls per second.
Rate of `CreateSchemaMapping` requests	5 TPS	Maximum number of `CreateSchemaMapping` API calls per second.
Rate of `DeleteSchemaMapping` requests	5 TPS	Maximum number of `DeleteSchemaMapping` API calls per second.
Rate of `GetSchemaMapping` requests	5 TPS	Maximum number of `GetSchemaMapping` API calls per second.
Rate of `ListSchemaMappings` requests	5 TPS	Maximum number of `ListSchemaMappings` API calls per second.

Resource	Rate limit	Description
Rate of UpdateSchemaMapping requests	5 TPS	Maximum number of UpdateSchemaMapping API calls per second.
Rate of GetPartnerComponent requests	5 TPS	Maximum number of GetPartnerComponent API calls per second.
Rate of ListPartnerComponents requests	5 TPS	Maximum number of ListPartnerComponents API calls per second.
Rate of TagResource requests	5 TPS	Maximum number of TagResource API calls per second.
Rate of UntagResource requests	5 TPS	Maximum number of UntagResource API calls per second.
Rate of ListTagsForResource requests	5 TPS	Maximum number of ListTagsForResource API calls per second.
Rate of CreateIdMappingWorkflow requests	5 TPS	Maximum number of CreateIdMappingWorkflow API calls per second.
Rate of DeleteIdMappingWorkflow requests	5 TPS	Maximum number of DeleteIdMappingWorkflow API calls per second.
Rate of GetIdMappingWorkflow requests	5 TPS	Maximum number of GetIdMappingWorkflow API calls per second.

Resource	Rate limit	Description
Rate of `ListIdMappingWorkflow` requests	5 TPS	Maximum number of `ListIdMappingWorkflow` API calls per second.
Rate of `UpdateIdMappingWorkflow` requests	5 TPS	Maximum number of `UpdateIdMappingWorkflow` API calls per second.
Rate of `ListProviderServices` requests	5 TPS	Maximum number of `ListProviderServices` API calls per second.
Rate of `GetProviderService` requests	5 TPS	Maximum number of `GetProviderService` API calls per second.
Rate of `CreateIdNamespace` requests	5 TPS	Maximum number of `CreateIdNamespace` API calls per second.
Rate of `DeleteIdNamespace` requests	5 TPS	Maximum number of `DeleteIdNamespace` API calls per second.
Rate of `GetIdNamespace` requests	5 TPS	Maximum number of `GetIdNamespace` API calls per second.
Rate of `ListIdNamespaces` requests	5 TPS	Maximum number of `ListIdNamespaces` API calls per second.
Rate of `UpdateIdNamespace` requests	5 TPS	Maximum number of `UpdateIdNamespace` API calls per second.

Resource	Rate limit	Description
Rate of AddPolicyStatement requests	5 TPS	Maximum number of AddPolicyStatement API calls per second.
Rate of DeletePolicyStatement requests	5 TPS	Maximum number of DeletePolicyStatement API calls per second.
Rate of GetPolicy requests	5 TPS	Maximum number of GetPolicy API calls per second.
Rate of PutPolicy requests	5 TPS	Maximum number of PutPolicy API calls per second.
Rate of GetMatchingJob requests	10 TPS	Maximum number of GetMatchingJob API calls per second.
Rate of ListMatchingJobs requests	5 TPS	Maximum number of ListMatchingJobs API calls per second.
Rate of StartMatchingJob requests	5 TPS	Maximum number of StartMatchingJob API calls per second.
Rate of GetMatchId requests	50 TPS	Maximum number of GetMatchId API calls per second.
Rate of GetIdMappingJob requests	10 TPS	Maximum number of GetIdMappingJob API calls per second.

Resource	Rate limit	Description
Rate of `ListIdMappingJobs` requests	5 TPS	Maximum number of `ListIdMappingJobs` API calls per second.
Rate of `StartIdMappingJob` requests	5 TPS	Maximum number of `StartIdMappingJob` API calls per second.
Rate of `BatchDeleteUniqueId` requests	5 TPS	Maximum number of `BatchDeleteUniqueId` API calls per second.

Document history for the AWS Entity Resolution User Guide

The following table describes the documentation releases for AWS Entity Resolution.

For notification about updates to this documentation, you can subscribe to the RSS feed. To subscribe to RSS updates, you must have an RSS plug-in enabled for the browser you are using.

Change	Description	Date
Support for enhanced rule conditions and incremental deletions	Customers can now use rule conditions with Boolean operators and new matching functions like ExactMany ToMany, allowing for more precise matching criteria with combinations of exact and fuzzy matching. Additionally, customers can delete records incrementally in advanced matching workflows using an Amazon S3 file.	July 30, 2025
Match ID processing clarification	Added clarification that the **Modify or generate match ID** and the **Look up match ID** options require **Automatic** processing cadence in matching workflows.	July 17, 2025
Generate a new match ID	Customers can now look up and modify an existing match ID or generate a new match ID when using a rule-based matching workflow.	June 2, 2025

Provider service-based matching workflow – update	Customers can now use Digital Identifiers such as IPV4, IPV6, and MAID when using the TransUnion provider service-based matching workflow.	April 21, 2025
Amazon CloudWatch Logs	AWS Entity Resolution now supports CloudWatch Logs integration, allowing you to enable detailed workflow logging that captures job execution metrics, timing, and processing statistics which can be delivered to CloudWatch Logs, Amazon S3, or Amazon Data Firehose destinations.	April 14, 2025
ID mapping workflow – update	Customers can now set up AWS Glue partitioning when using an ID mapping workflow.	March 25, 2025
Quotas – update	Documentation-only update. Rule-based matching workflows can process up to 100M records while machine learning-based matching workflows can process up to 250M records. Customers needing higher limits are directed to contact the service team.	February 7, 2025

Schema mapping – update	Documentation-only update to clarify that normalization is supported for Full name, Full address, and Full phone attribute types.	January 17, 2025
Provider integration	Documentation-only update. Customers can learn how to integrate as a provider service with AWS Entity Resolution.	August 8, 2024
ID mapping workflow – update	Customers can now use matching rules to translate first-party data in an ID mapping workflow.	July 23, 2024
Matching workflow – update	Customers can now delete the records from either a rule-based or ML-based matching workflow to help comply with data management regulatio ns.	April 8, 2024
ID mapping workflow – update	Customers can now use an ID mapping workflow across multiple AWS accounts.	April 2, 2024

AWS CloudFormation Resources - New and updated resources	AWS Entity Resolution has added the following resources: `AWS::EntityResolution::IdNamespace` and `AWS::EntityResolution::PolicyStatement` and updated the following resource: `AWS::EntityResolution::IdMappingWorkflow` .	April 2, 2024
Find Match ID	Customers can now find the corresponding Match ID and associated rule for a processed rule-based workflow.	March 25, 2024
Matching workflow – update	AWS Entity Resolution now supports PII-based RAMPID assignment in the LiveRamp provider service-based matching workflow.	February 12, 2024
AWS PrivateLink	AWS Entity Resolution now supports additional data security with AWS PrivateLink that helps customers to privately access services hosted on AWS.	October 20, 2023

AWS CloudFormation Resources – New and updated resources	AWS Entity Resolution has added the following resource: `AWS::EntityResolution:IdMappingWorkflow` and updated the following resources: `AWS::EntityResolution::MatchingWorkflow` and `AWS::EntityResolution::Schemamapping` .	October 19, 2023
Update to existing policy	The following new permissions have been added to the `AWSEntityResolutionConsoleFullAccess` managed policy: `ADXReadAccess` and `ManageEventBridgeRules` .	October 16, 2023
Schema mapping – update	Customers now have the ability to edit and update an existing data schema.	October 16, 2023
Matching workflow – update	Customers can now select a preferred data provider service to help match and link their data.	October 16, 2023
ID mapping workflow	Customers can use this new workflow to specify ID mapping details, choose your desired ID mapping method, and specify data input and output fields.	October 16, 2023

AWS CloudFormation integration	AWS Entity Resolution now integrates with AWS CloudFormation.	August 24, 2023
AWS managed policy update - New policies	AWS Entity Resolution added two new managed policies.	August 18, 2023
Initial release	Initial release of the AWS Entity Resolution User Guide	July 26, 2023

AWS Entity Resolution Glossary

Amazon Resource Name (ARN)

A unique identifier for AWS resources. ARNs are required when you need to specify a resource unambiguously across all of AWS Entity Resolution, such as in AWS Entity Resolution policies, Amazon Relational Database Service (Amazon RDS) tags, and API calls.

Attribute type

The type of the attribute for the input field. When you create a schema mapping, you select the **Attribute type** from a pre-configured list of values such as **Name**, **Address**, **Phone number**, or **Email address**. Attribute type tells AWS Entity Resolution what kind of data that you're presenting it, allowing it to be classified and normalized properly.

Automatic processing

A processing cadence option for a matching workflow job that enables it to be run on automatically when your data input changes.

This option is available for rule-based matching only.

By default, the processing cadence for a matching workflow job is set to **Manual**, which enables it to be run on demand. You can set up **Automatic** processing to run your matching workflow job automatically when your data input changes. This keeps your matching workflow output up-to-date.

AWS KMS key ARN

This is your AWS KMS Amazon Resource Name (ARN) for encryption at rest. If not provided, system will use an AWS Entity Resolution managed KMS key.

Cleartext

Data that isn't cryptographically protected.

Confidence level (ConfidenceLevel)

For ML matching, this is the confidence level applied by AWS Entity Resolution when ML identifies a matched record set. This is part of the matching workflow metadata that will be included in output.

Decryption

The process of transforming encrypted data back to its original form. Decryption can only be performed if you have access to the secret key.

Encryption

The process of encoding data into a form that appears random using a secret value called a key. It's impossible to determine the original plaintext without access to the key.

Group name

The **Group name** references the entire group of input fields and can help you to group parsed data together for matching purposes.

For example, if there are three input fields: **`first_name`**, **`middle_name`**, and **`last_name`**, you can group them together by entering in the **Group name** as **`full_name`** for matching and output.

Hash

Hashing means applying a cryptographic algorithm that produces an irreversible and unique string of characters of a fixed size—called a hash. AWS Entity Resolution uses Secure Hash Algorithm 256-bit (SHA256) hash protocol and will output a 32-byte character string. In AWS Entity Resolution, you can choose whether to hash data values in your output.

Hash protocol (HashingProtocol)

AWS Entity Resolution uses Secure Hash Algorithm 256-bit (SHA256) hash protocol and will output a 32-byte character string. This is part of the matching workflow metadata that will be included in output.

ID mapping method

How you want the ID mapping to be performed.

There are two ID mapping methods:

- Rule-based – The method by which you use matching rules to translate first-party data from a source to a target in an ID mapping workflow.
- Provider services – The method by which you use a provider service to translate third party-encoded data from a source to a target in an ID mapping workflow.

  AWS Entity Resolution currently supports LiveRamp as the provider services-based ID mapping method. You must have a subscription to LiveRamp through AWS Data Exchange to use this method. For more information, see Step 1: Subscribe to a provider service on AWS Data Exchange.

ID mapping workflow

A data processing job that maps data from an input data source to an input data target based on the specified ID mapping method. It produces an ID mapping table. This workflow requires you to specify the ID mapping method and the input data you want to translate from a source to a target.

You can set up an ID mapping workflow to run in your own AWS account or across two AWS accounts.

ID namespace

A resource in AWS Entity Resolution that contains metadata explaining datasets across multiple AWS accounts and how to use these datasets in an ID mapping workflow.

There are two types of ID namespaces: `SOURCE` and `TARGET`. The `SOURCE` contains configurations for the source data that will be processed in an ID mapping workflow. The `TARGET` contains a configuration of the target data to which all sources will resolve to. To define the input data that you want to resolve across two AWS accounts, create an ID namespace source and an ID namespace target to translate your data from one set (`SOURCE`) to another (`TARGET`).

After you and another member create ID namespaces and run an ID mapping workflow, you can join a collaboration in AWS Clean Rooms to run a multi table join on the ID mapping table, and analyze the data.

For more information, see the AWS Clean Rooms User Guide.

Input field

An input field corresponds to a column name from your AWS Glue input data table.

Input Source ARN (InputSourceARN)

The Amazon Resource Name (ARN) that was generated for an AWS Glue table input. This is part of matching workflow metadata that will be included in output.

Machine learning-based matching

Machine learning-based matching (ML matching) finds matches across your data that might be incomplete or might not look exactly the same. ML matching is a preset process that will attempt to match records across all of the data you input. ML matching returns a match ID and a confidence level for each matched set of data.

Manual processing

A processing cadence option for a matching workflow job that enables it to be run on demand.

This option is set by default and is available for both rule-based matching and machine learning - based matching.

Many-to-Many matching

Many-to-many matching compares multiple instances of similar data. Values in input fields that have been assigned the same match key will be matched against each other, regardless of whether they are in the same input field or different input fields.

For example, you might have multiple phone number input fields like `mobile_phone` and home_phone that have the same match key “Phone”. Use many-to-many matching to compare data in the `mobile_phone` input field with data in the `mobile_phone` input field and data in the home_phone input field.

Matching rules evaluate data in multiple input fields with the same match key with an (or) operation, and one-to-many matching compares values across multiple input fields. This means that if any combination of `mobile_phone` or `home_phone` matches between two records, the "Phone" match key will return a match. For match key "Phone" to find a match, `Record One mobile_phone = Record Two mobile_phone` OR `Record One mobile_phone = Record Two home_phone` OR `Record One home_phone = Record Two home_phone` OR `Record One home_phone = Record Two mobile_phone`.

Match ID (MatchID)

For rule-based matching and ML matching, this is the ID generated by AWS Entity Resolution and applied to each matched record set. This is part of the matching workflow metadata that will be included in output.

Match key (MatchKey)

Match key instructs AWS Entity Resolution which input fields to consider as similar data and which to consider as different data. This helps AWS Entity Resolution automatically configure rule-based matching rules and compare similar data stored in different input fields.

If there are multiple types of phone number information like a `mobile_phone` input field and a `home_phone` input field in your data that you would like compared together, you could give them both the match key "Phone". Then rule-based matching can be configured to compare data using "or" statements in all input fields with the "Phone" match key (see One-to-One Matching and Many-to-Many Matching definitions in Matching Workflow section).

If you want rule-based matching to consider different types of phone number information completely separately, you can create more specific match keys like "Mobile_Phone" and "Home_Phone". Then, when setting up a matching workflow, you can specify how each phone match key will be used in rule-based matching.

If no MatchKey is specified for a particular input field, it can't be used in matching but can be carried through the matching workflow process and can be output if desired.

Match key name

The name assigned to a Match key.

Match rule (MatchRule)

For rule-based matching, this is the rule number applied that generated a matched record set. This is part of the matching workflow metadata that will be included in output.

Matching

The process of combining and comparing data from different input fields, tables, or databases and determining which of it is alike – or “matches” – based upon satisfying certain matching criteria (for example, either through matching rules or models).

Matching workflow

The process that you set up to specify the input data to match together and how the matching should be performed.

Matching workflow description

An optional description of the matching workflow that you might choose to enter. Descriptions help you differentiate between matching workflows if you create more than one.

Matching workflow name

The name for the matching workflow that you specify.

> **Note**
>
> Matching workflow names must be unique. They can't have the same name or an error will be returned.

Matching workflow metadata

Information generated and output by AWS Entity Resolution during a matching workflow job. This information is required on output.

Normalization (ApplyNormalization)

Choose whether to normalize input data as defined in the schema. Normalization standardizes data by removing extra spaces and special characters and standardizing to lowercase format.

For example, if an input field has an attribute type of Full phone, and the values in the input table are formatted as `(123) 456-7890`, AWS Entity Resolution will normalize the values to `1234567890`.

> **Note**
>
> Normalization is only supported the group type for Name, Address, Phone, and Email.

The following sections describe our standard normalization rules.

For ML-based matching specifically, see Normalization (ApplyNormalization) – ML-based only.

Topics

Name

> **Note**
>
> Normalization is only supported for the **Name** group type.
> The **Name** group type appears as **Full name** in the console and as `NAME` in the API.
> If you want to normalize the sub-types of the **Name** group type:
>
> - In the console, assign the following subtypes to the **Full name** group: **First name**, **Middle name**, and **Last name**.

- In the CreateSchemaMapping API, assign the following **Types** to the `NAME` **groupName**: `NAME_FIRST`, `NAME_MIDDLE`, and `NAME_LAST`.

- **TRIM** = Trims leading and trailing whitespace
- **LOWERCASE** = Lowercases all alpha characters
- **CONVERT_ACCENT** = Covert accented letter to regular letter
- **REMOVE_ALL_NON_ALPHA** = Removes all non-alpha characters [a-zA-Z]

Email

> **Note**
>
> Normalization is supported for the **Email** group type.
> The **Email** group type appears as **Email address** in the console and as `EMAIL_ADDRESS` in the API.

- **TRIM** = Trims leading and trailing whitespace
- **LOWERCASE** = Lowercases all alpha characters
- **CONVERT_ACCENT** = Covert accented letter to regular letter
- **EMAIL_ADDRESS_UTIL_NORM** = Removes any dots (.) from the username, removes anything after a plus sign (+) in the username, and standardizes common domain variations
- **REMOVE_ALL_NON_EMAIL_CHARS** = Removes all non-alpha-numeric characters [a-zA-Z0-9] and [.@-]

Phone

> **Note**
>
> Normalization only supported for the **Phone** group type.
> The **Phone** group type appears as **Full phone** in the console and as `PHONE` in the API.
> If you want to normalize the sub-types of the **Phone** group type:

- In the console, assign the following sub-types to the **Full phone** group: **Phone number**, and **Phone country code**.
- In the CreateSchemaMapping API, assign the following **Types** to the `PHONE` **groupName**: `PHONE_NUMBER` and `PHONE_COUNTRYCODE`.

- **TRIM** = Trims leading and trailing whitespace
- **REMOVE_ALL_NON_NUMERIC** = Removes all non-numeric characters [0-9]
- **REMOVE_ALL_LEADING_ZEROES** = Removes all leading zeroes
- **ENSURE_PREFIX_WITH_MAP, "phonePrefixMap"** = Examines each phone number and tries to match it against patterns in the phonePrefixMap. If a match is found, the rule will add or modify the prefix of the phone number to ensure it conforms to the standardized format specified in the map.

Address

Note

Normalization only supported for the **Address** group type.
The **Address** group type appears as **Full address** in the console and as `ADDRESS` in the API.
If you want to normalize the sub-types of the **Address** group type:

- In the console, assign the following sub-types to the **Full address** group: **Street address 1**, **Street address 2**: **Street address 3 name**, **City name**, **State**, **Country**, and **Postal code** t
- In the CreateSchemaMapping API, assign the following **Types** to the `ADDRESS` **groupName**: `ADDRESS_STREET1`, `ADDRESS_STREET2`, `ADDRESS_STREET3`, `ADDRESS_CITY`, `ADDRESS_STATE`, `ADDRESS_COUNTRY`, and `ADDRESS_POSTALCODE`.

- **TRIM** = Trims leading and trailing whitespace
- **LOWERCASE** = Lowercases all alpha characters
- **CONVERT_ACCENT** = Covert accented letter to regular letter

- **REMOVE_ALL_NON_ALPHA** = Removes all non-alpha characters [a-zA-Z]
- **RENAME_WORDS using ADDRESS_RENAME_WORD_MAP** = replace words in Address string with words from ADDRESS_RENAME_WORD_MAP
- **RENAME_DELIMITERS using ADDRESS_RENAME_DELIMITER_MAP** = replace delimiters in Address string with string from ADDRESS_RENAME_DELIMITER_MAP
- **RENAME_DIRECTIONS using ADDRESS_RENAME_DIRECTION_MAP**= replace delimiters in Address string with string from ADDRESS_RENAME_DIRECTION_MAP
- **RENAME_NUMBERS using ADDRESS_RENAME_NUMBER_MAP** = replace numbers in Address string with string from ADDRESS_RENAME_NUMBER_MAP
- **RENAME_SPECIAL_CHARS using ADDRESS_RENAME_SPECIAL_CHAR_MAP** = replace special characters in Address string with string from ADDRESS_RENAME_SPECIAL_CHAR_MAP

ADDRESS_RENAME_WORD_MAP

These are the words that will be renamed when normalizing the address string.

```
"avenue": "ave",
 "bouled": "blvd",
 "circle": "cir",
 "circles": "cirs",
 "court": "ct",
 "centre": "ctr",
 "center": "ctr",
 "drive": "dr",
 "freeway": "fwy",
 "frwy": "fwy",
 "highway": "hwy",
 "lane": "ln",
 "parks": "park",
 "parkways": "pkwy",
 "pky": "pkwy",
 "pkway": "pkwy",
 "pkwys": "pkwy",
 "parkway": "pkwy",
 "parkwy": "pkwy",
 "place": "pl",
 "plaza": "plz",
 "plza": "plz",
 "road": "rd",
```

```
 "square": "sq",
 "squ": "sq",
 "sqr": "sq",
 "street": "st",
 "str": "st",
 "str.": "strasse"
```

ADDRESS_RENAME_DELIMITER_MAP

These are the delimiters that will be renamed when normalizing the address string.

```
",": " ",
".": " ",
"[": " ",
"]": " ",
"/": " ",
"-": " ",
"#": " number "
```

ADDRESS_RENAME_DIRECTION_MAP

These are the direction identifiers that will be renamed when normalizing the address string.

```
"east": "e",
"north": "n",
"south": "s",
"west": "w",
"northeast": "ne",
"northwest": "nw",
"southeast": "se",
"southwest": "sw"
```

ADDRESS_RENAME_NUMBER_MAP

These are the number strings that will be renamed when normalizing the address string.

```
"número": "number",
 "numero": "number",
 "no": "number",
 "núm": "number",
 "num": "number"
```

ADDRESS_RENAME_SPECIAL_CHAR_MAP

These are the special characters string that will be renamed when normalizing the address string.

```
"ß": "ss",
 "ä": "ae",
 "ö": "oe",
 "ü": "ue",
 "ø": "o",
 "æ": "ae"
```

Hashed

- **TRIM** = Trims leading and trailing whitespace

Source_ID

- **TRIM** = Trims leading and trailing whitespace

Normalization (ApplyNormalization) – ML-based only

Choose whether to normalize input data as defined in the schema. Normalization standardizes data by removing extra spaces and special characters and standardizing to lowercase format.

For example, if an input field has an attribute type of `NAME`, and the values in the input table are formatted as `Johns Smith`, AWS Entity Resolution will normalize the values to `john smith`.

The following sections describe the normalization rules for machine learning-based matching workflows.

Topics

- Name
- Email
- Phone

Name

- **TRIM** = Trims leading and trailing whitespace
- **LOWERCASE** = Lowercases all alpha characters

Email

- **LOWERCASE** = Lowercases all alpha characters
- Replaces only (at)(case sensitive) with an @ symbol
- Removes all whitespace, anywhere in the value
- Removes everything that's outside of the first "< >" if it exists

Phone

- **TRIM** = Trims leading and trailing whitespace
- **REMOVE_ALL_NON_NUMERIC** = Removes all non-numeric characters [0-9]
- **REMOVE_ALL_LEADING_ZEROES** = Removes all leading zeroes
- **ENSURE_PREFIX_WITH_MAP, "phonePrefixMap"** = Examines each phone number and tries to match it against patterns in the phonePrefixMap. If a match is found, the rule will add or modify the prefix of the phone number to ensure it conforms to the standardized format specified in the map.

One-to-One matching

One-to-one matching compares single instances of similar data. Input fields with the same match key and values in the same input field will be matched against each other.

For example, you might have multiple phone number input fields like `mobile_phone` and `home_phone` that have the same match key “Phone”. Use one-to-one matching to compare data in the `mobile_phone` input field with data in the `mobile_phone` input field and to compare data in the `home_phone` input field with data in the `home_phone` input field. Data in the `mobile_phone` input field won't be compared with data in the `home_phone` input field.

Matching rules evaluate data in multiple input fields with the same match key with an (or) operation, and one-to-many matching compares values within a single input field. This means

that if mobile_phone or home_phone matches between two records, the “Phone” match key will return a match. For match key “Phone” to find a match, Record One mobile_phone = Record Two mobile_phone OR Record One home_phone = Record Two home_phone.

Matching rules evaluate data in input fields with different match keys with an (and) operation. If you want rule-based matching to consider different types of phone number information completely separately, you can create more specific match keys like “mobile_phone” and “home_phone”. If you want to use both match keys in a rule to find matches, Record One mobile_phone = Record Two mobile_phone AND Record One home_phone = Record Two home_phone.

Output

A list of **OutputAttribute** objects, each of which have the fields **Name** and **Hashed**. Each of these objects represent a column to be included in the AWS Glue output table and whether you want the values in the column to be hashed.

OutputS3Path

The S3 destination to which AWS Entity Resolution will write the output table.

OutputSourceConfig

A list of OutputSource objects, each of which have the fields **OutputS3Path**, **ApplyNormalization**, and **Output**.

Provider service-based matching

Provider service-based matching is process designed to match, link, and enhance your records with preferred data service providers and licensed data sets. You must have a subscription through AWS Data Exchange with the provider service to use this matching technique.

AWS Entity Resolution currently integrates with the following data service providers:

- LiveRamp
- TransUnion

- UID 2.0

Rule-based matching

Rule-based matching is process designed to find exact matches. Rule-based matching is a hierarchical set of waterfall matching rules, suggested by AWS Entity Resolution, based upon the data that you input and completely configurable by you. All match keys provided within rule criteria must match exactly for compared data to be declared a match and for associated metadata to be output. Rule-based matching returns a Match ID and a rule number for each matched set of data.

We recommend defining rules that can uniquely identify an entity. Order your rules to find more precise matches first.

For example, let's say you have two rules, **Rule 1** and **Rule 2**.

These rules have the following match keys:

- **Rule 1** includes Full name and Address
- **Rule 2** includes Full name, Address, and Phone

Because **Rule 1** runs first, no matches will be found by **Rule 2** because they would have all been found by **Rule 1**.

To find matches that are differentiated by Phone, reorder the rules, like this:

- **Rule 2** includes Full name, Address, and Phone
- **Rule 1** includes Full name and Address

Schema

The term used for a structure or layout defining how a set of data is organized and connected.

Schema description

An optional description of the schema that you can choose to enter. Descriptions help you differentiate between schema mappings if you create more than one.

Schema name

The name of the schema.

> **Note**
>
> Schema names must be unique. They can't have the same name or an error will be returned.

Schema mapping

Schema mapping in AWS Entity Resolution is the process by which you tell AWS Entity Resolution how to interpret your data for matching. You define the schema of the input data table that you want AWS Entity Resolution to read into a matching workflow.

Schema mapping ARN

The Amazon Resource Name (ARN) generated for the schema mapping.

Unique ID

A unique identifier that you designate and that must be assigned to each row of input data that AWS Entity Resolution reads.

Example

For example: **`Primary_key`**, **`Row_ID`**, or **`Record_ID`**.

The **Unique ID** column is required.

The **Unique ID** must be a unique identifier within a single table.

The **Unique ID** must satisfy this pattern: `[a-zA-Z0-9_-]`

Across different tables, the **Unique ID** can have duplicate values.

The maximum **Unique ID** length is 38 for a matching workflow

The maximum **Unique ID** length 257 characters for a ID mapping workflow

When the matching workflow is run, the record will be rejected if the **Unique ID**:

- isn't specified
- isn't unique within the same table
- overlaps in terms of attribute name across sources
- exceeds 38 characters (rule-based matching workflows only)