

User Guide

AWS Transform



AWS Transform: User Guide

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Table of Contents

What is AWS Transform?	1
Terminology	1
Getting started	4
Setting up	4
Before you begin	4
Getting started with AWS Organizations	4
Getting started with AWS IAM Identity Center	5
Enable AWS Transform	5
Quick start: Trying AWS Transform	6
Understanding collaborator permissions	6
User roles	7
Role permissions	7
Human-in-the-loop (HITL) actions	8
Managing users	9
Adding users in IAM Identity Center	9
Adding users to AWS Transform	10
Onboarding	10
Accepting the invitation	10
Signing in to AWS Transform	11
Welcome experience	11
Workspaces	12
Creating a workspace	12
Adding team members to a workspace	12
Deleting a workspace	13
Deleting a job	13
Migration assessment	14
Creating and starting a job	14
Migration assessment workflow	15
Preparing DII files	15
Tracking the progress of a migration assessment job	16
VMware migration	17
Capabilities and key features	17
Limitations	17
VMware migration jobs	18

	Getting a workspace	. 18
	Job types	. 19
	Creating and starting a job	. 20
	VMware migration workflow	. 20
	Kick off migration	. 20
	Connect discovery account	. 21
	Discover source data	. 22
	Generate application groupings and waves	. 23
	Connect target account	. 24
	Migrate network	. 26
	Set up service permissions	. 31
	Prepare waves	. 32
	Migrate waves	. 32
	AWS account connectors	. 36
	Discovery account connector	36
	Target account connector	. 37
	Tracking the progress of a migration job	. 38
Ma	ninframe modernization	40
	Capabilities and key features	. 40
	High-level walkthrough	. 41
	Human in the loop (HITL)	. 41
	Supported file types for transformation of mainframe applications	. 42
	Supported Regions and quotas for AWS Transform mainframe	. 42
	Modernization journey	. 42
	Phase 1: Assess	43
	Phase 2: Mobilize	. 44
	Phase 3: Migrate and modernize	45
	Phase 4: Operate, optimize, and innovate	. 46
	Transformation of mainframe applications workflow	. 47
	Prerequisite: Prepare code in S3	. 47
	Step 1: Sign-in and onboarding	. 48
	Step 2: Create and start a job	. 49
	Step 3: Set up a connector	49
	Step 4: Tracking transformation progress	51
	Step 5: Analyze code	51
	Step 6: Generate technical documentation	55

Step 7: Extract business logic	59
Step 8: Decomposition	61
Step 9: Migration wave planning	64
Step 10: Refactor code	65
Step 11: Re-run the job	68
Step 12: Deployment capabilities in AWS Transform	69
Build and deploy modernized application post-refactoring	70
Prerequisites	70
Step 1: Retrieve the modernized code	71
Step 2: Build the modernized application	71
Step 3: Configure the test environment	72
Step 4: Deploy the modernized application	73
Step 5: Test the modernized application	73
Additional example	74
.NET	75
Capabilities and key features	75
Supported versions and project types	75
Limitations	76
Human intervention	76
More information	76
.NET web app	77
.NET Quick start guide	77
Create .NET job	82
Connect a source code repo	84
Confirm your repos	86
Use the plan generated by AWS Transform	88
Customize the transformation plan	89
Resolve missing dependencies	90
Review transformation plan	
Transform code	94
.NET IDE	97
Transform .NET in Visual Studio	98
How it works	100
Troubleshooting	102
Security	104
Data protection	104

Data encryption	106
Service improvement	112
Cross-region processing	112
Identity and access management	114
Audience	114
Authenticating with identities	115
Managing access using policies	118
How AWS Transform works with IAM	121
Identity-based policy examples	127
Troubleshooting	134
Using service-linked roles	136
AWS managed policies	139
AWS Transform permissions reference	141
Compliance validation	143
Resilience	144
VPC endpoints (AWS PrivateLink)	145
Considerations for AWS Transform VPC endpoints	145
Prerequisites	145
Creating an interface VPC endpoint for AWS Transform	146
Using an on-premises computer to connect to a AWS Transform endpoint	146
Monitoring	147
Monitoring with CloudWatch	147
Monitoring events	148
CloudTrail logs	148
AWS Transform information in CloudTrail	148
Understanding AWS Transform log file entries	149
Notifications	149
Email notifications	149
Quotas	151
Supported Regions	156
Supported AWS Regions (enabled by default)	156
Document history	157

What is AWS Transform?

AWS Transform is a service that helps you accelerate and simplify the transformation of infrastructure, applications, and code with an agentic AI experience. It provides specialized tools to streamline modernization and migration efforts across various workloads.

With AWS Transform, you can:

- Modernize IBM z/OS migrations to AWS
- Migrate VMware workloads to Amazon EC2
- Modernize .NET applications to Linux-ready cross-platform .NET
- Assess workloads for migration readiness

AWS Transform helps you offload labor-intensive, complex transformation tasks across the discovery, planning, and execution phases to AI agents with expertise in languages, frameworks, and infrastructure. This approach allows your teams to focus on innovation while it efficiently transforms complex, large-scale projects through a unified web experience.

There are no additional charge to use AWS Transform.

Terminology

Within this section, italics indicate an official term within the definition of a different term.

Account connection

Account in this context is a generalized reference to a customer-owned container or security boundary for resources in AWS or remote service, for example, an AWS account or GitHub account.

Artifact

An output deliverable produced by AWS Transform.

Administrator

Administrators can read and mutate everything in the workspace. They can begin chats with AWS Transform, start and stop jobs, and upload/download artifacts. Administrators can interact

Terminology 1

with running jobs for human-in-the-loop (HITL) actions, and can approve critical HITL actions such as merging to main, performing graph decomposition, or deploying code to production environments. Administrators can mutate workspaces, connectors, and users.

Agent

A task-specific service that executes a specific transformation type. For example, VMware migrations.

Approver

Approvers have permissions similar to Administrators except that they do not have permissions to mutate workspaces, connectors, or users. Approvers cannot mutate <u>workspaces</u>, <u>connectors</u>, or users.

Asset

Input for a transformation <u>job</u>. For example, customer's source code, server, database, network. Assets are accessed via a <u>connector</u>.

Collaborator request

A *task* in which AWS Transform is asking a human to do something.

Connector

Connectors are asset providers that allow access to customer-owned resources in a system external to AWS Transform.

When you set up a connector, the administrator of the account to which you are connecting must accept the connection. In order to accept the connection, they must have permissions given in the connector acceptor policy.

The following two accounts must either be identical, or in the same AWS Organizations organization:

- The account from which the AWS Transform administrator enables the service.
- The account that will be on the receiving end of your transformation. This account must be assigned an IAM role that allows it to use a connector.

Contributor

Contributors can read everything in the workspace. They can begin chats with AWS Transform, start and stop jobs, upload or download artifacts, and interact with running jobs for HITL actions. However, they cannot perform critical HITL actions such as merging to main,

Terminology 2

performing graph decomposition, or deploying code to production environments. Contributors also cannot mutate workspaces, connectors, or users.

Objective

A user-defined end state that AWS Transform works to reach. Users write this, and then AWS Transform converts the objective into a series of tasks that it perform in concert with users when required.

Job

A long-running process (weeks/months+) that AWS Transform is working on in order to fulfill an objective defined by a user. Made up of multiple tasks and collaborator requests.

Plan

A list of <u>tasks</u> that AWS Transform undertakes (with help from human users) in pursuit of an objective.

Reader

Readers can view the status and outcomes of AWS Transform jobs, but cannot make any changes. They can read everything in the <u>workspace</u>, download artifacts, view <u>jobs</u>, and view human-in-the-loop (HITL) actions. However, readers cannot perform mutating actions in AWS Transform.

Task

An individual unit of work that is part of a job.

Worklog

A log of what actions AWS Transform and users have performed as part of a job.

Workspace

A AWS Transform resource that contains other resources like <u>connectors</u> and <u>jobs</u>. A <u>workspace</u> serves as a permissions boundary.

Terminology

Getting started with AWS Transform

Topics

- Setting up AWS Transform
- Understanding collaborator permissions
- Managing users
- User onboarding
- Setting up your workspace

Setting up AWS Transform

Before you begin

Before you set up AWS Transform make sure you have an AWS account with administrator access



Note

If you want to try out AWS Transform as a proof-of-concept or for test environments see Quick start: Trying AWS Transform.

Getting started with AWS Organizations

If you're using AWS Organizations, follow these steps to set up AWS Transform:

- 1. Sign in to your AWS Organizations management account.
- 2. Navigate to the AWS Transform service.
- 3. Choose **Enable service** for your organization to use AWS Transform.
- Configure the necessary permissions for organizational member accounts. 4.
- 5. Access the AWS Transform web experience from your member accounts.

Now you're ready to set up your workspace.

Setting up

Getting started with AWS IAM Identity Center

Follow these steps to use IAM Identity Center for AWS Transform and to add users and groups.



Note

By default, no users have access to AWS Transform when you first enable it.

Set up IAM Identity Center following the instructions in To enable an instance of IAM Identity Center.

Configure IAM Identity Center to use an external enterprise identity provider, and replicate its user and group info into IAM Identity Center.

- In the AWS console, select AWS Transform and choose **Get started**. 2.
- 3. Choose **Enable service** for your organization to use AWS Transform.
- Select an encryption key. The default selection is an AWS managed key. To use a custom key: 4.
 - Under Encryption key, choose Customize encryption settings. a.
 - b. Select Use an AWS KMS key.
 - Choose an existing key or create a new one. c.
 - Choose **Submit** to apply your changes, and then choose **Enable AWS Transform**. d.

Click View profile to view the configuration. The Web application URL is used by your users to access the AWS Transform unified web experience.

- Select **Users** in the navigation pane and select **Assign users or groups**.
- Search for the name of the user or groups you want to authorize to use AWS Transform. The search references users and groups propagated from your identity provider.
- Select a group or user, select **Done**, and then, **Assign**. These users are authorized to use the AWS Transform unified web interface.

Enable AWS Transform

To enable AWS Transform:

Sign in to the AWS Management Console.

- 2. In the search bar at the top of the console, search for "AWS Transform".
- 3. Select **AWS Transform** from the search results.
- 4. On the AWS Transform homepage, choose **Get started**.
- 5. Review the service information, including benefits and features.
- 6. Choose **Get started** again to enable the service in your current Region.
- 7. The system displays "Enabling AWS Transform" while it creates the necessary resources.

After AWS Transform is enabled, the service configuration page displays the following information:

- Web application URL The URL for accessing the AWS Transform web application
- **Start URL for IDE** The URL for accessing AWS Transform in integrated development environments
- Region The AWS Region where AWS Transform is enabled

Quick start: Trying AWS Transform

The easiest way to try out AWS Transform is with a standalone AWS account. You may want to do this as a proof-of-concept or for test environments. Use this procedure:

- 1. Sign in to the AWS Management Console.
- Navigate to the AWS Transform service.
- 3. Choose **Get started** to enable the service.
- 4. After the service is enabled, you'll see the AWS Transform web application URL.
- 5. Open that URL in a new browser window to access the AWS Transform web experience.

Now you're ready to set up your workspace.

Understanding collaborator permissions

AWS Transform uses a workspace-based permission model to control access to resources and actions. Each user is assigned a specific role within a workspace, which determines what actions they can perform. A user can have different roles in different workspaces.

User roles

AWS Transform supports five user roles within each workspace. These roles apply within the context of a workspace, and a user will be assigned roles in each workspace they are a member of. The access permissions defined for each role are workspace agnostic, so user A with the Administrator role in workspace A has the same permissions as user B with the Administrator role in workspace B.

Role permissions

Detailed permissions for each role:

Action	ResourceType	Admin	Approver	Contribut or	ReadOnly
Create	Workspace	✓	✓	✓	✓
List	Workspace	✓	✓	✓	✓
Get	Workspace	✓	✓	✓	✓
Update	Workspace	✓	x	X	X
Delete	Workspace	✓	X	X	X
Create	ChatMessage	✓	✓	✓	✓
Read	ChatMessage	✓	✓	✓	✓
Create	RoleAssociation	✓	x	X	X
Read	RoleAssociation	✓	✓	✓	✓
Update	RoleAssociation	✓	x	X	X
Delete	RoleAssociation	✓	X	X	X
Read	CriticalHITLTask	√	✓	✓	✓
Update	CriticalHITLTask	✓	✓	X	X

User roles 7

Action	ResourceType	Admin	Approver	Contribut or	ReadOnly
Delete	CriticalHITLTask	✓	✓	X	X
Read	HITLTask	✓	✓	✓	✓
Update	HITLTask	✓	✓	✓	x
Delete	HITLTask	✓	✓	✓	x
Create	Job	✓	✓	✓	x
Read	Job	✓	✓	✓	✓
Update	Job	✓	✓	✓	x
Delete	Job	✓	✓	✓	x
Read	Worklog	✓	✓	✓	✓
Create	Artifact	✓	✓	✓	x
Read	Artifact	✓	✓	✓	✓
Update	Artifact	✓	✓	✓	x
Delete	Artifact	✓	✓	✓	x
Create	Connector	✓	✓	✓	x
Read	Connector	✓	✓	✓	✓
Update	Connector	✓	✓	✓	X
Delete	Connector	✓	✓	✓	X

Human-in-the-loop (HITL) actions

AWS Transform provides two types of HITL actions - standard and critical:

Standard HITL actions

These are routine actions that can be performed by users with Contributor, Approver, or Administrator roles.

Critical HITL actions

These are actions with significant impact, and thus require higher permission levels. Examples include:

- Merging code to main branches
- · Performing graph decomposition
- Deploying code to production environments

Critical HITL actions can only be performed by users with Approver or Administrator roles.

To ensure there's a differentiation between Standard HITL and Critical HITL actions in AuthZ policies, AWS Transform provides two separate HITL APIs, one for completing a standard HITL action, and one for completing a critical HITL action.

Managing users

AWS Transform integrates with IAM Identity Center for user management. This section describes how to add users to IAM Identity Center and grant them access to AWS Transform.

Adding users in IAM Identity Center

To add users in IAM Identity Center:

- 1. Navigate to the IAM Identity Center console.
- 2. In the navigation pane, choose **Users**.
- Choose Add user.
- 4. Enter the required information:
 - **Username** A unique identifier for the user (cannot be changed later)
 - Email address The user's email address
 - First name and Last name The user's name
 - **Display name** The name that appears in the user list

Managing users 9

- 5. For **Password**, choose how the user receives their password:
 - Send an email Send setup instructions via email
 - Generate a one-time password Create a password to share manually
- 6. Choose **Next** to review the user information.
- 7. Review the details and choose **Add user**.

After the user is added, they'll receive an email invitation to set up their IAM Identity Center account. The invitation link is valid for 7 days.

You can also learn about working with IAM Identity Center and AWS Transform in this video:

VideoTitle

Adding users to AWS Transform

After adding users to IAM Identity Center, you can grant them access to AWS Transform:

- 1. Return to the AWS Transform console.
- 2. In the navigation pane, choose **Users and groups**.
- 3. Select the **Users** tab or the **Groups** tab.
- 4. Search for and select the users or groups that you want to add from IAM Identity Center.
- 5. Choose **Assign users and groups** to grant the selected users or groups access to AWS Transform.

After adding users, they appear in the **Users** list with a status of "Pending" until they accept the invitation and sign in.

User onboarding

This section describes the experience for users who have been granted access to AWS Transform.

Accepting the invitation

When a user is added to AWS Transform, they receive an email invitation containing:

A greeting and information about the invitation

- The AWS Transform web application URL
- Their username
- A link to accept the invitation and set up their password

To set up their account:

- 1. The user clicks the "Accept invitation" link in the email.
- 2. On the "New user sign up" page, they enter and confirm a password.
- 3. The password must meet security requirements, including:
 - At least 8 characters
 - At least one uppercase letter
 - At least one lowercase letter
 - At least one number
 - At least one special character
- 4. After creating a password, they see a confirmation that their account was successfully created.

Signing in to AWS Transform

To sign in to AWS Transform:

- 1. Navigate to the AWS Transform web application URL provided in the invitation email.
- 2. Enter the username.
- Choose Next.
- 4. Enter the password.
- 5. Choose **Sign in**.

Welcome experience

Upon first login, users see the AWS Transform welcome page with:

- A personalized greeting
- Available transformation capabilities
- Option to create a workspace

Signing in to AWS Transform

The welcome page provides information about the transformation capabilities available in AWS Transform, including:

- Modernize IBM z/OS migrations to AWS
- Migrate VMware workloads to Amazon EC2
- Modernize .NET applications to Linux-ready cross-platform .NET
- · Assess workloads for migration readiness

Users can start by creating a workspace or asking their team to add them to an existing workspace.

Setting up your workspace

Workspaces in AWS Transform help you organize your transformation projects and collaborate with team members.

Creating a workspace

To create a new workspace:

- 1. From the AWS Transform welcome page, choose **Create a workspace**.
- 2. Enter a name for your workspace.
- 3. Provide a description that helps team members understand the workspace's purpose.
- 4. Select the appropriate settings for your transformation project.
- Choose Create.

Adding team members to a workspace

To add team members to your workspace:

- 1. Navigate to the workspace settings.
- 2. Choose **Team members**.
- 3. Select the users you want to add to the workspace.
- 4. Assign appropriate roles and permissions.
- 5. Choose **Add**.

Workspaces 12

Team members receive notifications about their access to the workspace.

Deleting a workspace

To delete a workspace:

- 1. Navigate to the workspace settings.
- 2. From the dropdown, select **Delete workspace**.
- 3. In the pop-up, enter **delete-workspace** to confirm.
- 4. Choose Confirm.

Deleting a job

In order to delete a job:

- · You must have a role other than reader.
- The job must be in a terminal state (completed, failed, or stopped).

To delete a job:

- 1. In the **Jobs** tab, identify the job that you want to delete.
- 2. On the panel displaying the job you want to delete, choose the three vertical dots.
- 3. From the dropdown menu, choose **Delete job**.
- 4. Choose Confirm.
- 5. In the pop-up, enter **delete-workspace** to confirm.
- 6. Choose **Confirm**.

Deleting a workspace 13

Migration assessment

The assessment job type in AWS Transform is designed to provide cost estimates and savings for migrating your workloads to AWS.

This job type uses an advanced AI-powered language model specifically trained for AWS infrastructure analysis. The specialized LLM evaluates Amazon EC2 instance recommendations, performs BYOL (Bring Your Own License) analysis, and determines optimal dedicated host mappings by incorporating real-time Amazon EC2 pricing data and host configuration specifications.

The assessment job type currently supports compute-based workloads. While it includes servers deployed to run specialized workloads, it cannot assess the specialized workloads themselves.

When planning to migrate to AWS you can use this assessment to:

- Get cost estimates, including Amazon EC2 and Amazon EBS and storage cost estimates, for your migration
- · Identify potential savings opportunities
- Receive Amazon EC2 instance recommendations
- Analyze licensing options (BYOL)
- Determine optimal dedicated host mappings

Creating and starting a job

The first step of a migration project is to create an AWS Transform job.

To create and start a new migration assessment job

- 1. On your workspace landing page, choose **Create a job with AWS Transform**.
- 2. Choose the migration assessment option.
- 3. Review the job details that AWS Transform proposes. You can specify a different name for the job if you'd like.
- 4. Choose Create and start a job.

Creating and starting a job 14

Migration assessment workflow

- 1. In the left navigation pane, choose **Share on-premises server data**.
- 2. Upload data files that can be used by AWS Transform for the assessment. Make sure the file includes all the servers that you want to assess for migration to AWS. You can include 30,000 servers per assessment job. The maximum supported file size is 10 MB. These types of files are supported:
 - RVTools: You can upload either a ZIP of .csv files or an excel file that RVTools produces when you choose **Export all to Excel** from the RVTools **File** menu.
 - Migration Portfolio Assessment (MPA) import file
 - Export for vCenter
 - Migration Evaluator Quick Insights. You can download the Migration Evaluator Quick Insights file from the Migration Evaluator Console.
 - A Microsoft Excel file created from the AWS Transform Assessment Data template. You can download the AWS Transform Assessment data template from AWS Transform.
- 3. To include a storage assessment, upload a NetApp Data Infrastructure Insights (DII) file or an Excel file created from the AWS Transform Assessment data template. See Preparing DII files to learn more about DII files. You can download the AWS Transform Assessment Data template from AWS Transform.
- 4. Specify the AWS Region where you want to host your migrated workloads. All commercial AWS Regions are supported.
- 5. Choose **Generate business case**. The right pane automatically switches to the **Worklog** tab where you can track the progress of the job.
- 6. Download and review the business case.
- 7. (Optional) Use the chat pane to ask AWS Transform questions about your business case.

Preparing DII files

To provide AWS Transform with the necessary data for a storage assessment you have to run NetApp queries and save the results in CSV files. Zip the CSV files and upload the zip to AWS Transform.

For NAS volumes, run these queries:

- Disk
- Internal Volumes
- Storage
- Share

For SAN volumes, run these queries:

- Virtual Machines
- Volume

Follow these steps to run the queries and save the resulting files:

- From the home page of your NetApp DII tenant choose Explore and then choose New Metric Query.
- 2. Select the query type.
- 3. Select a period of 30 days.
- 4. Open the settings from the gear icon and choose **Select all the columns**.
- 5. Save as a CSV file by selecting the **Export to csv** icon.

Tracking the progress of a migration assessment job

The **Worklog** tab provides a detailed log of the actions that AWS Transform takes, along with human input requests and your responses to those requests. AWS Transform adds entries to the worklog to show the progress of the assessments, including processing uploaded files, analyzing files, generating the assessment report, and any errors that occur.

VMware migration

AWS Transform can help you migrate your VMware environment to Amazon EC2 by using generative AI. This document provides an overview of AWS Transform and of the workflow of the migration process.

Capabilities and key features

AWS Transform offers the following capabilities and key features for migrating your VMware environment to AWS.

- Three discovery options:
 - Assisted discovery of your VMware environment by using collectors from AWS Application Discovery Service.
 - Use the open-source Export for vCenter tool.
 - Importing independently collected discovery data.
- Al-driven conversion of your source VMware network configuration to an Amazon VPC network architecture.
- Al-driven generation of migration plans, including application grouping and suggested migration waves.
- Rehosting your servers to run natively on Amazon EC2.

AWS Transform supports migrating Windows and Linux servers of supported operating systems. For the full list of supported operating systems, see <u>Supported operating systems</u> in the AWS Application Migration Service User Guide.

Limitations

AWS Transform has the following limitations:

 AWS Transform does not support using Amazon Elastic Compute Cloud Dedicated Hosts or the Bring Your Own License (BYOL) licensing model. You may decide to switch tenancy and use BYOL licenses after you launch Amazon EC2 instances. For information, see <u>License type conversions in</u> <u>License Manager</u> in the *License Manager User Guide*.

Capabilities and key features 17

• If you stop a running migration job, and then ask the agent to restart it, the job will start again from the beginning and you will lose any progress you have made in the job. However, artifacts created in the job before restarting it will still be available.

- You can specify one target AWS account and one target AWS Region per VMware migration
 job. To migrate waves to different target accounts or different Regions, create multiple VMware
 migration jobs, and use the same source account connector for your inventory. For information
 about the two types of account connectors, see the section called "AWS account connectors".
- To create a target account connector, the target AWS account must have a default VPC in your target AWS Region. The default VPC is used by AWS Transform to stage data from your source servers for replication to AWS. For information about how to create a default VPC if the account doesn't have one, see Create a default VPC in the Amazon VPC User Guide.
- NSX imports are only supported for end-to-end migration jobs.
- AWS Transform can execute a single network migration job for a given target AWS account and AWS Region at a time. When you generate a VPC configuration during the network migration step of your job, if other jobs are concurrently performing a network migration for the same account and Region, AWS Transform will delete the metadata associated with those other network migrations, and that might cause those other jobs to fail. For NSX environment migrations, this means AWS Transform cannot automatically associate security groups with migrated servers. However, any network resources (such as VPCs, Subnets, and security groups) already deployed to your target account and Region won't be deleted. We recommend you complete the first job before generating a VPC configuration in another job for the same target account and Region.

VMware migration jobs

To use AWS Transform for VMware migrations, you first need a workspace, which is a logical container in which you can create one or more transformation jobs. The sections in this topic describe how to get a workspace and how to create and start a VMware migration job in it.

Getting a workspace

For information about getting a workspace, see <u>Getting started</u>.

The workspace that you use determines the AWS Region where you can create transformation jobs. That is the AWS Region where your jobs will reside. Your discovery data and AWS Transform recommendations will also reside in this AWS Region. To create workspaces and jobs in a different

VMware migration jobs 18

AWS Region, ask your administrator to create a different workspace for you. For information about supported AWS Regions, see *Supported Regions*.

Even though the AWS Region where you can create jobs and store discovery data and recommendations is determined by your AWS Transform administrator, you can specify a different AWS Region as your target for the migration. In other words, you can run discovery and receive AWS Transform recommendations in one AWS Region, but then create your target environment in a different AWS Region. If you do that, you will be transferring your data across AWS Regions. For more information, see the section called "Target account connector".

Job types

AWS Transform offers the following types of VMware migration jobs that you can choose from depending on your migration needs.

End-to-end migration

- 1. Perform discovery
- 2. Generate wave plan
- 3. Generate VPC configuration
- 4. (Optional) Deploy VPC networks
- 5. Migrate servers

Network migration only

- 1. Generate VPC configuration
- 2. (Optional) Deploy VPC networks

Network-and-server migration

- 1. Generate VPC configuration
- 2. (Optional) Deploy VPC networks
- 3. Migrate servers

Job types 19

Discovery and server migration

- 1. Perform discovery
- 2. Generate wave plan
- 3. Migrate servers

Creating and starting a job

The first step of a migration project is to create an AWS Transform job. For VMware migration projects, you can choose different job types, depending on your goals. The following procedure describes how to create and start a new VMware migration job of any type. For information about the different job types, see the section called "Job types".

To create and start a new VMware migration job

- 1. On your workspace landing page, choose **Create a job**.
- Choose the VMware migration option, and then specify the type of VMware migration job
 that you want to create. For information about the steps included in each of the four VMware
 migration job types, see the section called "Job types".
- 3. After you answer all the chat questions, choose **Create job**.

VMware migration workflow

The type of VMware migration job you choose determines the workflow. For a description of the different types of VMware migration jobs, see the section called "Job types".

Kick off migration

Here you can optionally invite collaborators to work with you on this migration job. For more information about the different collaborator roles and the permissions they have, see <u>the section</u> called "Understanding collaborator permissions".

After you (optionally) invite collaborators, choose Send to AWS Transform.

Creating and starting a job 20

Connect discovery account

In this step, you create a connector to an AWS account and AWS Region that AWS Transform can use for storing and analyzing discovery data. You can either use an existing discovery connector if your workspace has one, or you can create a new discovery connector. For information about the role of the discovery account in this migration process, and for discovery Region considerations, see the section called "Discovery account connector".

AWS Transform will create an Amazon S3 bucket on your behalf in this discovery AWS account. This bucket won't have SecureTransport enabled by default. If you want the bucket policy to include secure transport, you must update the policy yourself. For more information, see Security best practices for Amazon S3.

To use an existing discovery connector

- In the Job Plan pane expand Connect discovery account, and then choose Create or select connectors.
- In the **Collaboration** tab, select an existing connector from the list of available connectors, and then choose Use connector. If a connector is grayed out, that means its version isn't compatible with the job type that you selected earlier.
- Choose Send to AWS Transform.

To create a new connector

- In the Job Plan pane expand Connect discovery account, and then choose Create or select connectors.
- 2. If you have existing connectors in the current workspace, go to the **Collaboration** tab, and choose **Create new connector**.
- In the **Collaboration** tab, enter the ID of the AWS account that you want to use for discovery, and then choose Next.
- Choose whether you want to use Amazon S3 managed keys for encryption. If you specify your own KMS key, you can use the default key policy. However, if you want a less permissive key policy, the following is an example. For information about how to create a KMS key, see Create a KMS key in the AWS Key Management Service Developer Guide.

Connect discovery account 21

AWS Transform uses the kms:DescribeKey permission to make sure the key exists. It uses the kms:GenerateDataKey and kms:Decrypt permissions to encrypt and decrypt the transformation job data in the Amazon S3 bucket.

AWS Transform uses default Amazon S3 encryption. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys

- 5. Choose Continue.
- Copy the verification link, share it with an administrator of the discovery AWS account, and ask them to approve the connection request.
- After the administrator of the AWS account approves the request, choose **Send to AWS** Transform.

Discover source data

In the Job Plan pane expand Discover source data, and then choose Perform discovery.

To collect source data, set up one of the two types of AWS collectors. These collectors gather data that AWS Transform can use to generate Amazon EC2 recommendations and wave plans for you. If you don't use these AWS collectors, you can upload an RVTools file in which you specify application groupings and waves.

AWS Transform uses all of the data collected in the source AWS account for its analysis. As a result, you will see all files that were imported with previous jobs that used the same source account. Importing the same file again will not create conflicts because AWS Transform automatically handles the de-duplication of servers in its analysis.



Marning

The official RVTools site is https://www.dell.com/en-us/shop/vmware/sl/rvtools, which is the site that this guide links to in steps that mention RVTools. Beware of the scam site (rvtools)(dot)(org).

Discover source data 22



Note

There is a known issue with RVTools 4.7.1. Learn more in RVTools 4.7.1 bug with VMware Cloud on AWS.

To get network translation, you must upload an NSX export or an RVTools export.

RVTools file: This type of file is used for networks that use vSphere constructs. RVTools is a utility that exports detailed information about your VMware environment, including vSwitches, port groups, and VLANs. You can upload either a ZIP of .csv files or an excel file that RVTools produces when you choose **Export all to Excel** from the RVTools **File** menu.

Import/Export for NSX file: This type of file is required if your network uses VMware NSX (Network Virtualization and Security Platform). This file is generated using an open-source tool provided by AWS called Import/Export for NSX. It exports all software-defined network (SDN) resources in JSON format.

Export for vCenter file: You may prefer this type of file if your administrative workstation does not run Windows, RVTools is not approved in your organization, or you want more granular control over the export file's contents.

After you upload a data file, set up collectors, or do both, choose **Continue**. The next step is to review discovery data.

To review discovery data

- 1. In the **Job Plan** pane, choose **Review source data for discovery**.
- 2. If AWS Transform states that more data is needed, choose **Set up collectors**, and follow the instructions for setting ups collectors.
- After you set up collectors, we recommend that you let them collect data for at least one week. When you are satisfied with the collected data, choose Continue with existing data, and then choose **Generate waves**.

Generate application groupings and waves

AWS Transform uses the discovery data to generate application groupings and waves. If you didn't set up collection, AWS Transform can only generate a pre-populated template of the servers. In this step you can download a file that contains the groupings and waves that AWS Transform

generated. You can then work with your stakeholders to review and adjust these groupings and waves if necessary. Only servers with an application and application wave provided will be included in the migration.

- In the Job Plan pane, expand Generate application groupings and waves, and choose Plan and review migration waves.
- 2. In the Collaboration tab, choose Download file.
- 3. Review the application groupings and waves and adjust them if necessary. Do not add or remove columns or change the titles of the existing columns.
- 4. Under **Upload waves to AWS Transform**, upload the waves file that you reviewed (and possibly adjusted).
- 5. Choose **Continue**.

Connect target account

The target account is where your network will be deployed and where your migrated servers and applications will reside in AWS. For more information, see the section called "Target account connector".

To create a target account connector, the target AWS account must have a default VPC in your target AWS Region. The default VPC is used by AWS Transform to stage data from your source servers for replication to AWS. For information about how to create a default VPC if the account doesn't have one, see Create a default VPC in the Amazon VPC User Guide.

▲ Important

- To create a target account connector, the target AWS account must have a default VPC in your target AWS Region. The default VPC is used by AWS Transform to stage data from your source servers for replication to AWS. For information about how to create a default VPC if the account doesn't have one, see Create a default VPC in the Amazon VPC User Guide.
- AWS Transform will create an Amazon S3 bucket on your behalf in this target AWS
 account. This bucket won't have SecureTransport enabled by default. If you want the
 bucket policy to include secure transport, you must update the policy yourself. For more
 information, see Security best practices for Amazon S3.

Connect target account 24

To use an existing target account connector

In the Job Plan pane, expand Choose target account, and then choose Create or select connectors.

In the **Collaboration** tab, select an existing connector if your workspace already has connectors, and then choose **Use connector**. In the list of available connectors, if a connector is grayed out, that means its version isn't compatible with the job type that you selected earlier.



Important

If you specify a connector with a target AWS Region that is different from the discovery AWS Region, that means AWS Transform will be transferring your data across AWS Regions.

Choose Continue. 3.

To create a new connector

- In the Job Plan pane, expand Connect target account, and then choose Create or select connectors.
- Specify the AWS account and AWS Region that you want to use as your target, and then choose Next.



Important

If you specify a target AWS Region that is different from the discovery AWS Region, that means AWS Transform will be transferring your data across AWS Regions.

Choose whether you want to use Amazon S3 managed keys for encryption. If you specify your own KMS key, you can use the default key policy. However, if you want a less permissive key policy, the following is an example. For information about how to create a KMS key, see Create a KMS key in the AWS Key Management Service Developer Guide.

AWS Transform uses the kms: DescribeKey permission to make sure the key exists. It uses the kms:GenerateDataKey and kms:Decrypt permissions to encrypt and decrypt the transformation job data in the Amazon S3 bucket.

Connect target account 25

AWS Transform uses default Amazon S3 encryption. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys

- 4. Choose Continue.
- 5. Copy the verification link, share it with an administrator of the target AWS account, and ask them to approve the connection request.
- 6. After the administrator of the AWS account approves the request, select the newly created connector from the list of connectors in the **Collaboration** tab, and then choose **Use connector**.
- 7. Choose **Send to AWS Transform**.

If you plan to modify the AWS Application Migration Service template to enable post-launch actions, add the following permission to the target connector role. You can find the name of that role in the **Collaboration** tab after the connector is created. For information about how to add permissions to a role, see Update permissions for a role in the *IAM User Guide*.

```
{
    "Sid": "MGNPostLaunchActions",
    "Effect": "Allow",
    "Action": [
        "iam:PassRole"
],
    "Resource": "arn:aws:iam::target-account-ID:role/service-role/
AWSApplicationMigrationLaunchInstanceWithSsmRole"
}
```

Migrate network

AWS Transform can migrate your VMware networks to AWS. The migration process translates your source environment configuration to an AWS-equivalent network. AWS Transform analyzes your source network data and translates your source network to these AWS networking resources as needed: VPCs, subnets, security groups, network access control lists (NACLs), NAT gateways, transit gateways, internet gateways, elastic IPs, routes, and route tables. AWS Transform then creates AWS CloudFormation templates and AWS Cloud Development Kit (AWS CDK) templates. You'll be able to review the generated network configuration, and then either deploy it on your own or ask AWS Transform to deploy it for you.

The translation process requires you to upload a configuration file from your source environment. You can use RVTools, Export for vCenter, or Import/Export for NSX to capture on-premisesnetwork data, and then import that file to AWS Transform. The choice of tool depends on the type of source network that you have:

- If you have an NSX-defined network, upload an NSX configuration file using Import/Export for NSX.
- If you have a vSphere-constructs-defined network, you can upload an RVTools file. AWS Transform will use that data to generate Amazon VPC configurations for you to review and deploy in your target AWS account. If you upload an RVTools file, AWS Transform won't create security groups because RVTools files don't include the information required to generate security groups.

If your administrative workstation does not run Windows, RVTools is not approved in your organization, or you want more granular control over the export file's contents, use an Export for vCenter file, which generates a file in the same format as RVTools.

Ensure that the target account has the required quotas.



Marning

The official RVTools site is https://www.dell.com/en-us/shop/vmware/sl/rvtools, which is the site that this guide links to in steps that mention RVTools. Beware of the scam site (rvtools)(dot)(org).

AWS Transform analyzes your source network data and translates your source network to these AWS networking resources as needed: VPCs, subnets, security groups, network access control lists (NACLs), NAT gateways, transit gateways, internet gateways, elastic IPs, routes, and route tables.

If you want the target network to contain different IP ranges than your source network you can modify your VPC CIDR ranges during migration. AWS Transform automatically propagates changes to subnets, route tables, and security groups, and then assigns static IPs based on the CIDR. This is best for applications requiring predictable network behavior, DNS management, or IP-based access control. IP addresses persist across instance restarts using Elastic Network Interfaces (ENIs).

For NSX environments, you can choose to have AWS Transform automatically convert security policies and gateway policies to Security Groups, associating them with network interfaces based

on VM external IDs and IP addresses. In this case you will not have the option to use dynamic IP assignment (DHCP) for your target instances because IP addresses aren't known until instance launch.

AWS Transform then creates AWS CloudFormation templates and AWS Cloud Development Kit (AWS CDK) templates.

Once the target network is generated, review the generated network configuration. You can either deploy it on your own or ask AWS Transform to deploy it for you. If you choose to let AWS Transform deploy the network for you, it also uses tools such as <u>Reachability Analyzer</u> to run an analysis to check connectivity between subnets across multiple VPCs and under the same VPC. If you choose to make changes to the generated configuration, you must deploy the modified configuration yourself.

Network topologies

During the migration, you can choose one of two network topologies for the target network:

- Isolated VPCs Each VPC serves as its own unit with no intercommunication among VPCs.
- **Hub and Spoke** AWS Transform creates a transit gateway and connects all VPCs using route tables. It also creates 3 additional VPCs:
 - Inspection VPC A placeholder for the firewall to inspect the traffic.
 - Inbound VPC For all traffic from the public internet (North-South). It has an internet gateway.
 - Outbound VPC For all traffic to the public internet. It has a NAT gateway and an elastic IP.

By default AWS Transform doesn't open the communication to the internet for either topology. Once you are ready to use your new network and have configured security precautions according to your company policies, you can manually open the network to the internet

If you want fine-grained control over the communication between the VPCs, choose the **Isolated VPCs** option and modify the generated network to create the specific communication paths your require.

IP migration approaches

AWS Transform offers three IP addressing strategies for your migration:

• IP Address Retention: Keep original IP addresses during migration. Ideal for lift-and-shift scenarios with legacy applications that have hard-coded IP dependencies or existing firewall rules.

- Static IP Assignment: You can modify VPC CIDR ranges during migration, and AWS Transform automatically propagates changes to subnets, route tables, and security groups, and then assigns static IPs based on the CIDR. This is best for applications requiring predictable network behavior, DNS management, or IP-based access control. IPs persist across instance restarts using Elastic Network Interfaces (ENIs).
- Dynamic IP Assignment (AWS DHCP): Automatically assign IPs from subnet pools at instance launch. Optimal for cloud-native applications, auto-scaling workloads, and container orchestration (ECS, EKS, ASGs). Reduces operational overhead but requires applications to use DNS or service discovery.

Note

IP strategy is set at the wave level. You can assign different strategies to specific servers by customizing the wave file. For example if you chose a static IP address approach for the wave, but want to assign a dynamic approach to a specific server, you would use [RESET_VALUE] as described in Editing your configuration In the Application Migration Service user guide.

Generate VPC configuration

To import network data

- 1. In the **Job Plan** pane, choose **Migrate network**.
- 2. Expand **Generate VPC configuration**.
- 3. Choose **Import network data**.
- 4. In the **Network source data** section, either select an existing import, or choose **Upload file** to import a new file to add to the list, and then select the file that you uploaded.
- 5. In the **Network topology** section, the topology that you want AWS Transform to generate.
- 6. Choose **Generate VPC configuration**.
- 7. After AWS Transform generates a Amazon VPC configuration, choose **Review generated VPC configuration**.

In this step you can choose to either **use the current configuation** or **Modify your VPC CIDRs**. You cannot modify the prefix length, which comes after the "/".

To modify your VPC CIDRs:

- a. In the **Generated VPCs** list provide your modified CIDRs.
- b. Choose the **Regenerate network** button.

Review the results and then choose to continue with network deployment or to repeat the **Modify your VPC CIDRs** and **Regenerate network** steps.

8. Review the generated network configuration, and then choose **Deploy using AWS Transform** or **Deploy on my own**. If you make changes to the generated configuration you have to deploy the modified configuration yourself. If you choose to let AWS Transform deploy the network for you, it uses tools such as <u>Reachability Analyzer</u> to run an analysis to check connectivity between subnets across multiple VPCs and under the same VPC.

Tag network resources

For AWS Transform to launch Amazon EC2 instances within your existing AWS network resources which were not created by AWS Transform, the target network resources must be tagged. You can ask AWS Transform to do the tagging for you, in which case it will tag all network resources that it finds in the target AWS account and AWS Region. You can also manually tag target network resources that you've created on your own with the following tags:

Key: CreatedFor Value: AWSTransform

Key: ATWorkspace Value: AWS Transform workspace the connector will be used in

Security group association

When you migrate networks from an NSX source environment, AWS Transform creates security groups based on your source environment configurations. AWS Transform converts the following NSX configurations to security groups:

- Security policies and security policy rules
- Gateway policies and gateway policy rules

The association is based on the source groups for outbound communication, and on the destination_groups for inbound communication. During server migration (import inventory), AWS Transform uses the following logic to associate these security groups with the appropriate network interfaces:

- Rules associated to a VM external ID are attached to all of the elastic network interfaces of the given VM.
- Rules associated to an IP are attached to the network interface with the specific IP.
- Exclude rules are ignored. However, you can manually create replacements for them.

The association maintains the security rules specified in the inventory import. The migration process creates encoded mapping files, one per VPC in your Amazon S3 bucket, that link VM external identifiers and IP addresses to the security groups they should be bound to. Follow AWS best practices to secure these files because they contain sensitive information about mappings. For guidance, see the section called "Data protection".

Important

Do not modify these mapping files, as they are essential for proper security group association. Modifying these files will cause a failure during the import inventory phase.

Security groups removed from the VPC after network deployment will not be associated with migrated servers.

During import, AWS Transform automatically deduplicates security groups based on their ID so as to remove redundant assignments.

Set up service permissions

In this step, you initialize the AWS Application Migration Service (Application Migration Service) if you haven't already. To learn more about this requirement, see Initializing Application Migration Service with the console or Initializing AWS Application Migration Service with the API.

After you initiate Application Migration Service, AWS Transform helps you add MAP tags to your resources (if you have a MAP 2.0 agreement in place) so that you can get MAP credit. For information about MAP, see AWS Migration Acceleration Program.

31 Set up service permissions

Prepare waves

In this step, AWS Transform processes the wave plan and then populates the job plan in the left navigation pane with a node for every migration wave. Under every node, there will be steps for migrating the servers in that wave.

Migrate waves

At this stage, you will see migration waves in the **Job Plan** pane. For each wave, perform the following steps. In some of these steps you will have the option of importing an updated inventory file. AWS Transform allows one import to a given target AWS account and target AWS Region at a time. This means that if you work on more than one wave simultaneously, or if there is more than one migration job running with the same target account, you must wait for an import to finish before you can perform another import in a different wave or job.

Set up migration wave

- In the Job Plan pane, expand the step Set up migration waves, and then choose Set EC2
 recommendation preferences. Follow the instructions in the right pane, and then choose
 Continue.
- 2. In the Job Plan pane, choose Confirm inventory for wave-name. Download the inventory file and review the list of servers and Amazon EC2 configurations. Modify the file if necessary, but do not add or remove columns or change the titles of the existing columns. After you choose whether to continue with the file you downloaded or to upload a version of the file that you updated, choose Continue.
- 3. For each wave, choose your IP assigment approach:
 - Use source IP or the converted IP from the new CIDR
 - Use new IP using DHCP



AWS Transform provides Amazon EC2 recommendations based on the utilization specification of your source VMs. You can modify the suggested Amazon EC2 instance types to include recommendations from the <u>Migration Evaluator</u>, <u>AWS Optimization and Licensing Assessment</u> (OLA), or a *Migration assessment* job.

Prepare waves 32

Deploy replication agents

 In the Job Plan pane, expand Deploy replication agents, and then choose Start replication agent deployment. You have two options:

• Use AWS Transform to automate deployment: To automate the deployment of the agents on the source servers in this wave, AWS Transform uses an MGN connector already deployed in your account. For information about how to deploy an MGN connector in your account, see Set up the MGN Connector in the Application Migration Service User Guide.

To use this option, perform the following steps:

- 1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
- 2. In the left navigation pane, under **Node Tools**, choose **Fleet Manager**.
- 3. Choose the name of the managed instance of the MGN connector that you want AWS Transform to use for this wave.
- 4. Tag the managed instance with the following key-value pairs.

Key: CreatedFor Value: AWSTransform

Key: ATWorkspace Value: AWS Transform workspace the connector will be used in>

- 5. In AWS Transform, choose **Use AWS Transform to automate deployment**.
- 6. Specify the MGN connector that you tagged and the AWS Secrets Manager secret that you want AWS Transform to use for this wave. You must create a single set of credentials for the MGN connector to use for deploying replication agents on all servers in a particular wave. For information about setting up the secret, see Register server credentials.
- 7. If AWS Transform encounters errors during the deployment of the agent, you will see those errors in the **Job Plan** pane. Choose each error in the **Job Plan** pane to view its details in the **Collaboration** tab.
- 8. After you resolve all errors, you can track the replication status for the wave by choosing **Review replication status** in the **Job Plan** pane.
- Deploy replication agents on your own: You can deploy the replication agents on the source servers manually. Alternatively, you can use the MGN connector or another automation framework to deploy them on your own. For information about how to set up

Migrate waves 33

the MGN connector, see Set up the MGN Connector in the Application Migration Service User Guide.

To deploy the replication agents manually, or use an automation framework other than the MGN Connector to deploy them, perform the following steps.

- 1. Go to the AWS Application Migration Service console, and export a list of your servers. For instructions, see Exporting your data inventory.
- 2. Filter the list by wave to obtain a list of the servers in the current wave.
- 3. Follow the instructions under Installing the AWS Replication Agent. Specify the user-provided-id parameter, and for every server set its value to the server's mgn:server:user-provided-id as it appears in the .csv file that you exported from AWS Application Migration Service. AWS Transform connects the replication agent with the imported server using this parameter. If it's not provided, MGN will create a separate instance of source server for each agent that is installed.

To see the replication agent installation status, check the AWS Systems Manager run command history at agent installation time. For information, see Understanding command statuses in the AWS Systems Manager User Guide.

To see the replication status in real-time, go to the AWS Application Migration Service console. Status updates in the AWS Transform web app are delayed.

For quotas related to replication, see AWS Application Migration Service service quota limits in the Application Migration Service User Guide.



Note

AWS Transform does not support MGN agentless replication. For information about agentless replication, see Agentless replication overview in the Application Migration Service User Guide.

When replication is complete, expand **Review the replication status** in the **Job Plan** pane. In 2. the right pane you can see the status of the replication and resolve replication alerts.

Migrate waves



Note

To proceed, you must install the MGN replication agent on all servers in a wave. Disconnect and archive servers on which you don't install the replication agent. You can use the disconnect-from-service command to disconnect servers. To archive disconnected servers, use the mark-as-archived command. The archiving command only works for source servers whose lifecycle state is DISCONNECTED.

Launch test instances

- In the Job Plan pane, under Launch test instances, choose Confirm instance launch. 1.
- 2. Download the inventory file, review it, and choose whether to continue with the current file or upload a modified one, then choose **Launch test instances**. You can change the launch settings within the inventory file, but don't modify the list of source servers and applications.

Mark applications as ready for cutover

- In the Job Plan pane, expand Mark applications as ready for cutover, and choose Mark applications as ready for cutover.
- In the **Collaboration** tab, review the replication status of each application, and resolve replication alerts.
- 3. Choose Mark for cutover.

Launch cutover instances

- 1. In the **Job Plan** pane, under **Launch cutover instances**, choose **Confirm instance launch**.
- Download and open the inventory file, review the inventory, and choose whether to continue 2. with the current inventory or upload a modified one. At this step, don't modify the list of source servers and applications listed in the inventory file. You can only change the launch settings within the inventory file.
- Choose whether to continue with the current inventory or upload a modified one, and then choose Continue.
- Choose Launch cutover instances. 4.

Migrate waves 35

Finalize cutover

1. (Optional) Review the launched Amazon EC2 cutover instances, validate connectivity and run acceptance tests. If you want to fix anything because there's a connectivity issue or a problem in the testing, you need to revert the cutover. This is the time to revert it.

- 2. In the **Job Plan** pane, expand **Finalize cutover**, and then choose **Start finalizing cutover**. Finalizing the cutover removes the replication agents. After you finalize the cutover you cannot make any changes, you cannot fix any connectivity issues, or anything else, and you cannot revert the cutover.
- 3. Choose Finalize cutover.

AWS account connectors for VMware migrations

To perform a VMware migration, you need two types of AWS account connectors.

Discovery account connector

This AWS account connector is for discovery and planning purposes. It gives AWS Transform permissions to perform discovery-related actions within the AWS account that you specify. AWS Transform performs these actions in the AWS Region of the workspace that has the VMware migration job. To use another AWS Region for discovery, ask your administrator to provide you a workspace in that AWS Region. A workspace can be in one of the following AWS Regions:

- US East (N. Virginia)
- Europe (Frankfurt)

After you create this connector, AWS Transform creates an Amazon S3 bucket for you in the discovery account and Region. It uses that bucket for storing data that is discovered from your source VMware environment. This discovery data is crucial for planning and performing the migration. This data includes information about your source servers, applications, networks, and dependencies.

AWS Transform uses the discovery data for the following purposes:

- To analyze your source environment, which is essential for planning the migration strategy.
- To understand your current network setup, which is crucial for planning the network configuration in AWS.

AWS account connectors 36

- To assess security requirements and compliance needs based on your current setup.
- To understand application dependencies, which is critical for planning the migration waves and ensuring all necessary components are moved together.

• To determine the appropriate Amazon EC2 instance types and sizes for your migrated VMs based on the discovery data.

Target account connector

This AWS account connector connects your migration job to your new AWS environment where your workloads will reside after the migration. It's important to ensure that the target AWS account that you specify for this connector is properly set up with the necessary permissions, quotas, and configurations to support your migrated infrastructure.

When you create your target AWS account connector, AWS Transform will ask you to specify a target AWS Region. That is the AWS Region where your target environment with all your servers will reside. You can specify any one of the following AWS Regions for the target account connector:.

- US East (N. Virginia)
- US East (Ohio)
- US West (Oregon)
- Asia Pacific (Mumbai)
- Asia Pacific (Tokyo)
- Asia Pacific (Seoul)
- Asia Pacific (Sydney)
- Asia Pacific (Singapore)
- Canada (Central)
- Europe (Frankfurt)
- Europe (London)
- Europe (Paris)
- Europe (Ireland)
- Europe (Stockholm)
- South America (São Paulo)

Target account connector 37

Important

If you specify a target AWS Region that is different from the discovery AWS Region, that means AWS Transform will be transferring your data across AWS Regions.

The target connector connects your migration job to the target AWS account and target AWS Region for the following purposes:

- Network-infrastructure setup The target account is where you will create new Amazon VPCs and associated network resources to host your migrated applications in the target AWS Region that you specify when you create the target connector.
- Amazon EC2-instance setup The target AWS account is where you will migrate your VMware virtual machines and run them as Amazon EC2 instances in the target AWS Region.
- **Testing and validation:** Before final cutover, you will use the target AWS account for testing the migrated servers and ensuring they function correctly in the AWS environment.
- Cost management The target AWS account will be where the costs for running your migrated infrastructure are incurred and where you can track those costs.
- Long-term operations Post-migration, this target AWS account becomes your primary account for operating and managing your former source workloads in AWS.



Note

AWS Transform may update connector types when introducing features requiring permission changes within your AWS accounts. You can use a connector version that is compatible with your VMware migration job. New connectors are created with the latest version for that connector type. The current version for the discovery account connector type is 1.0. The current version for target account connector type is 2.0.

Tracking the progress of a migration job

You can track the progress of the transformation in two ways:

• Worklog – Provides a detailed log of the actions that AWS Transform takes, along with human input requests, and your responses to those requests.

• **Dashboard** – Provides a high-level summary of the VMware migration.

Modernization of mainframe applications

AWS Transform is designed to accelerate the modernization of legacy mainframe applications. It orchestrates the analysis of mainframe codebases, generate documentation, extract business logic, decompose monolithic structures, transform legacy code, and manage the overall journey with human inputs (HITL) when needed. The transformation capabilities of AWS Transform for modernizing and migrating mainframe applications empower you to modernize your critical mainframe application faster, while preserving your business-critical logic throughout the transformation process.

Topics

- · Capabilities and key features
- High-level walkthrough
- Human in the loop (HITL)
- Supported file types for transformation of mainframe applications
- Supported Regions and quotas for AWS Transform mainframe
- High-level overview of mainframe modernization journey
- Transformation of mainframe applications
- Build and deploy your modernized application post-refactoring

Capabilities and key features

AWS Transform provides the following capabilities for mainframe modernization:

- Supports modernization of mainframe applications written in COBOL (Common Business-Oriented Language) with associated JCL (Job Control Language), CICS (Customer Information Control System) transactions, BMS (Basic Mapping Support) screens, Db2 databases, and VSAM (Virtual Storage Access Method) data files.
- Performs goal-driven reasoning, analysis, decomposition, planning, documentation generation, and code refactoring.
- Automatically refactors COBOL-based mainframe workloads into modern, cloud-optimized Java applications.
- Orchestrates and integrates seamlessly with underlying tools executing analysis, documentation, decomposition, planning, and code refactoring.

Capabilities and key features 40

• Helps you set up cloud environments for modernized mainframe applications by providing ready-to-use Infrastructure as Code (IaC) templates.

High-level walkthrough

The following steps provide a high-level walkthrough of AWS Transform for modernizing and migrating mainframe applications.

- 1. Start a chat with AWS Transform, and enter an objective.
- 2. Based on your objective, AWS Transform proposes a modernization plan—breaking down the high-level goal into intermediate steps.
- 3. Depending on the goal you provided, AWS Transform can:
 - Analyze the codebase
 - Generate technical documentation
 - Extract business logic from your mainframe applications
 - Decompose the monolithic application into functional domains
 - Plan waves for code modernization
 - Refactor the application assets, including transforming the COBOL codebase to Java-based architecture, and optionally Reforge to improve the quality of refactored code
 - · Re-run your jobs as needed
- 4. Along the way, AWS Transform might request information from you to execute the tasks.

Human in the loop (HITL)

Throughout the transformation of mainframe applications, you can monitor the progress and status of the transformation tasks through the AWS Transform web experience.

AWS Transform will gather additional information from you in the following scenarios:

- When additional information is needed to execute tasks.
- When approval is required for intermediate artifacts (for example, domains decomposition or wave planning).
- When issues arise that AWS Transform cannot automatically resolve.

High-level walkthrough 41

Supported file types for transformation of mainframe applications

The supported file types include:

- COBOL artifacts and related CPY (Copybooks)
- JCL (Job Control Language) and JCL Procedure (PROC)
- CICS System Definition (CSD)
- BMS (Basic Mapping Support)
- Db2 databases
- VSAM (Virtual Storage Access Method)

Supported Regions and quotas for AWS Transform mainframe

For a list for supported Regions, see Supported Regions for AWS Transform.



Note

Your data might be processed in a different Region from the Region where you use AWS Transform. For information on cross-region processing, see the section called "Cross-region" processing".

For the quota limits, see *Quotas*.

High-level overview of mainframe modernization journey

Modernizing mainframe applications to AWS is typically achieved in four phases: (a) assess, (b) mobilize, (c) migrate and modernize, and (d) operate, optimize, and innovate. Each of these phases are further described in detail with goals and associated activities to support your modernization journey.

Phases

Phase 1: Assess

- Phase 2: Mobilize
- Phase 3: Migrate and modernize
- Phase 4: Operate, optimize, and innovate

Phase 1: Assess

Goal: To understand your modernization readiness and develop initial modernization plans.

With AWS Transform you can evaluate your mainframe applications, infrastructure, and operations to determine the right modernization approach for your business needs.



Note

Each phase contains a list of all possible activities you can do when modernizing your mainframe application. You can skip some of the activities that don't align with your usecase or goals.

Activities

- · Start with an informal conversation about the modernization and work on opportunity qualification by getting the right people in the room
- Determine opportunity qualification
- Do a first-call presentation with application dependency mapping plan and application architecture
- Determine ROM (Rough order of magnitude) that includes infrastructure cost during migration, migration labor, post-production costs, project duration
- Rapid portfolio detection:
 - Execute pattern & tool guidance assessment
 - Do a source code analysis
 - Identify proof of concept (POC) candidates
- Create technical enablement training:
 - Immersion days
 - Workshops with general topics

Phase 1: Assess

• Create a Migration readiness assessment (MRA) report for large mainframe modernization projects or new AWS customers

Create a statement of work for Mobilize phase

Phase 2: Mobilize

Goal: To build foundation and validate approach through successful proof of concept or pilot modernizations.

With the mobilize phase, you can establish the necessary frameworks, tools, and processes to support mainframe modernization at scale.

Activities

- Kick off the modernization plan with a pilot and the identified POC:
 - Analyze the pilot and set up the code for modernization
 - · Build and test the pilot modernized data
 - Deliver results and review the pilot to create a detailed solution for larger data modernization
- For platform:
 - Create a security, compliance, and monitoring plan
 - Set up landing zone environments with AWS mainframe landing zone and account structure
 - Determine architecture infrastructure, application, integration, and more
- Think about people and processes:
 - Cloud Center of Excellence is a good resource
 - Operation model
- Create a comprehensive portfolio analysis for different application scenarios:
 - Comprehensive discovery that includes high-level documentation
 - Decomposition plan (for all applications)
 - Pattern and tool guidance for all applications
 - Come up with an initial modernization and migration plan
- Create a detailed solution considering pilot, platform, people and processes, and comprehensive portfolio analysis
- Determine the business case
- Complete successful pilot with solution for the statement of work for next phase

Phase 2: Mobilize 44

Phase 3: Migrate and modernize

Goal: To complete modernization and migration within planned timeline and budget.

During this phase, you can migrate and modernize your mainframe applications and data to AWS.

Activities

- Allocate resources to the production-ready application code
- Review and analyze the design using existing documentation or by interviewing subject matter experts (SMEs)
- · Define test scenarios
- Set up modernization environments with appropriate runtime, databases, and third-party software
- Migrate and modernize source code and data:
 - You can modernize your application iteratively based on business domains in the application codebase
 - Scale modernization environments as needed to support your modernization workloads
- Establish CI/CD (continuous integration, continuous delivery) pipeline
- Migrate selected workloads to initiate your modernization.
- Transfer application programs and adjust environment scaling as needed:
 - You can scale the environments up or down depending on your need
 - Modernize work packages by running code sanity tests
 - Resynchronize with source code as necessary
- Run comprehensive testing and verify your migration:
 - · Collect initial state data
 - Record test scenarios on mainframe
 - Stage environment, application, and data
 - Execute test scenarios on the cloud
 - Compare source and target test results
 - Build automated test scenario execution pipelines
 - Generate KPIs and reports
- Validate your data for production setup

- Validate your progress by:
 - · Running integration and system testing
 - Running performance testing
 - Running user acceptance testing
 - · Running HA/DR testing
- Define a roll-out plan with application owner and create a plan for training and knowledge transfer
- Consider the cutover and execute production cutovers
- Establish sustained support for post-production activities such as operations, monitoring, capability planning, and warranty for application functions

Phase 4: Operate, optimize, and innovate

Goal: To support operational excellence and continuous improvement of modernized applications.

After modernization, you can optimize your applications' performance, security, and costs while leveraging AWS services for innovation.

Activities

- Monitor application for:
 - · Performance metrics
 - Security practices
 - · Compliance guidelines
 - Costs
- Logging details
- Manage application operations:
 - Infrastructure management
 - Transactions and job management
 - Runtime environment support
- Application optimization and modernization
- Application development and maintenance:
 - Coding IDE

- Build
- Integration
- Testing
- Deployment

Transformation of mainframe applications

AWS Transform accelerates the transformation of your mainframe modernization applications from COBOL to Java. The following document guides you through the process of leveraging generative AI and the automated transformation capabilities of AWS Transform for analyzing codebases, planning transformation, and executing the refactored code in an accelerated manner. All of this while preserving your mission-critical business logic.

Topics

- Prerequisite: Prepare code in S3
- Step 1: Sign-in and onboarding
- Step 2: Create and start a job
- Step 3: Set up a connector
- Step 4: Tracking transformation progress
- Step 5: Analyze code
- Step 6: Generate technical documentation
- Step 7: Extract business logic
- Step 8: Decomposition
- Step 9: Migration wave planning
- Step 10: Refactor code
- Step 11: Re-run the job
- Step 12: Deployment capabilities in AWS Transform

Prerequisite: Prepare code in S3

AWS Transform is capable of handling complex mainframe codebases. To use this codebase, make sure you have all the assets in your S3 location.

• **Source code:** You must upload your mainframe source code files to S3. This includes COBOL programs, JCL scripts, copybooks, and any other relevant source files.

- **Data files**: If you have any VSAM files or other data files that your mainframe applications use, these need to be uploaded to S3.
- **Configuration files**: Any configuration files specific to your mainframe environment should be included.
- **Documentation**: If you have any existing documentation about your mainframe applications or systems, it's helpful to upload them to S3.

Note

- For technical documentation generation, you can leverage an optional configuration file to generate PDF documents which aligns with your required formats and standards, including headers, footers, logos, and customized information.
- AWS Transform leverages automation with Generative AI for documentation generation and business rule extraction. Including a glossary CSV file with information about important abbreviations and terminologies in the root directory of your zip file will help improve the generated documentation quality.
- **Test data**: If available, upload any test data sets that can be used to validate the modernized application.

Step 1: Sign-in and onboarding

To sign into the AWS Transform web experience, follow all the instructions in <u>Getting started with AWS Transform</u> section of the documentation.

When setting up your workspace for mainframe transformation, you can optionally set up an Amazon S3 bucket to be used with the S3 connector. After creating the bucket and uploading the desired input files into the bucket, save that S3 bucket ARN for use later. Or you can set up the S3 bucket when setting up the connector as well. For more information, see the section called "Step 3">the section called "Step 3". Set up a connector".

Important

AWS Transform will refuse operations from you if you don't have the proper permissions. For example, a contributor cannot cancel a job transformation of mainframe applications or delete a job. Only an administrator can perform these functions.

Step 2: Create and start a job

Follow these steps to start a new job in your workspace.

- On your workspace landing page, choose **Ask AWS Transform to create a job**. 1.
- 2. Next, choose **Mainframe Modernization** as the type of job.
- 3. In the chat window, AWS Transform will ask you to confirm the job details, such as, the job type, job name, and what steps you want this job to perform.



Note

You can ask AWS Transform to perform any combination of the capabilities mentioned in the section called "High-level walkthrough". But you always need to finish the **Analyze code** step.

4. Once confirmed, choose **Create job**.

AWS Transform then kicks off the modernization for your job.

Step 3: Set up a connector

In this step, you set up a connector with your Amazon S3 bucket, which allows AWS Transform to access resources, and perform consecutive transformation functions.

Under job plan, expand Kick off modernization, and choose Connect to AWS account.



Note

You directly skip to Specify resource location page if you have already created a connector and added S3 bucket when creating your workspace.

Enter the AWS account ID you would like to use to perform the mainframe modernization 2. capabilities.

- Choose Next. 3.
- Enter the Amazon S3 bucket ARN from earlier where your resources are stored for transformation of your mainframe applications.
- Choose **Create connector**. 5.
- Once you add the Amazon S3 bucket ARN, you will get a verification link. You must share 6. this link with your AWS administrator, and ask them to approve the request in the AWS Management Console. After the request is approved, you will see connection details with Amazon S3 as the connector type.



Note

If you need to create a different connector, you can choose to restart the set up connector process.

- When your connector is set to active, on the **Specify asset location** page, enter the Amazon S3 bucket path for the input resources you would like to transform for your mainframe applications.
- (Optional) You can also choose to enable AWS Transform chat to learn from the progress you make on this job. This will allow AWS Transform to assist you with better guidance and result generation in each step. This data will only be stored within your workspace and will not be used for any other purposes beyond this job. If you disable this experience, AWS Transform chat will guide you based in the publicly available information in AWS Documentation.
- Then, choose **Continue** to move to the next step.



Important

Your data will be stored and persisted in the AWS Transform's artifact store in your workspace and will only be used for running the job.

Step 3: Set up a connector

S3 bucket CORS permissions

When setting up your S3 bucket to view artifacts in AWS Transform, you need to add this policy to the S3 bucket's CORS permission. If this policy is not set up correctly, you may not be able to use the inline viewing or file comparison functionalities of AWS Transform.

Step 4: Tracking transformation progress

You can track the progress of the transformation throughout the process in two ways:

- **Worklog** This provides a detailed log of the actions AWS Transform takes, along with human input requests, and your responses to those requests.
- **Dashboard** This provides high-level summary of the mainframe application transformation. It shows metrics on number of jobs transformed, transformation applied, and estimated time to complete the transformation of mainframe applications. You can also see details of each step including, lines of code by file types, generated documentation by each file type, the decomposed code, migration plan, and the refactored code.

Step 5: Analyze code

After you share the Amazon S3 bucket path with AWS Transform, it will analyze the code for each file with details such as file name, file type, lines of code, and their paths.



Note

You can download the Analyze code results using the **Download** link in the left navigation pane. This will download a zip file that contains the classification file for manual classification workflow, assets, dependencies JSON file, and list of missing files.

Under Analyze code in the left navigation pane, choose View code analysis results.

You can view your code analysis results in multiple ways:

- List view All files in the Amazon S3 bucket you want to transform for mainframe
- File type view All files in the Amazon S3 bucket displayed per file type. For a list of supported file types, see Supported files.
- Folder view All files in the Amazon S3 bucket displayed in folder structure.

Within the file results, AWS Transform provides the following information depending on what file view you choose:

- Name
- File type
- Total lines of code
- File path
- Comment lines
- Empty lines
- Effective lines of code
- Number of files
- Cyclomatic Complexity Cyclomatic complexity represents the number of linearly independent paths through a program's source code. AWS Transform will show a cyclomatic complexity for each of the files. With this metric, you can evaluate code maintainability and identify areas that need refactoring.

Missing files – Missing files from the mainframe modernization code analysis. These files ideally, should be added as a part of the source input in Amazon S3 bucket, and the analysis step should be re-run for better and cohesive results.

Step 5: Analyze code 52

Identically named – AWS Transform gives you a list of files that share the same name, and possibly the same characteristics (e.g., number of lines of code). It will not have the ability to compare the difference between the contents of any two files at one time.

Duplicated IDs – With Cobol program, the Program ID field serves as the unique identifier of the file. This ID must be unique because it's used to call the program throughout your project. However, some projects might have COBOL files with different names but the same Program ID. Getting the list of those files during the assessment can help understand the dependencies among all programs.



Note

This is specific to COBOL code and files.

When you have programs with duplicated IDs, it's suggested to change the Program IDs of these files to have a unique identifier for each of these in the COBOL code. You can then re-run your job to get more accurate and comprehensive code analysis results.

By resolving duplicate Program IDs, you can:

- Improve code clarity and maintainability
- Reduce potential conflicts in program calls
- Enhance the accuracy of dependency mapping
- Simplify future modernization efforts

Update classification – With manual reclassification, you can reclassify files using the bulk update feature by uploading the JSON file with the new classification.



This is only available for UNKNOWN and TXT files.

After reclassification, AWS Transform will:

- 1. Updates the classification results
- 2. Re-runs dependency analysis with the new file types

Step 5: Analyze code

3. Refreshes all affected analysis results



Note

You can reclassify files only after the initial analysis loop completes.

Inline viewer and file comparison

The Inline viewer is a feature in the AWS Transform for mainframe capabilities that provides two key visualization capabilities:

- File view: View content of selected legacy files from jobs
- File comparison: Compare content of two legacy files side-by-side

Input file viewing

To view your files in the Analyze code step

Under View code analysis results, select a file using the check box in the list.

Choose the **View** action button (enabled when 1 item is selected).

File content will be rendered on screen in the **File View** component.

File comparison

To compare files in the Analyze code step

- Under View code analysis results, select two files using the check boxes in the list. 1.
- 2. Choose the **Compare** action button (enabled only when 2 items are selected).
- Files will be displayed side-by-side in the **File comparison** component. 3.



Note

You can't select more than two files to compare files.

Step 5: Analyze code

Important

If you're having issues with inline viewer or file comparison make sure that the S3 bucket is set up correctly. For more information on S3 bucket's CORS policy, see the section called "S3 bucket CORS permissions".

Step 6: Generate technical documentation

In this step, you can generate technical documentation for your mainframe applications undergoing modernization. By analyzing your code, AWS Transform can automatically create detailed documentation of your application programs, including descriptions of the program logic, flows, integrations, and dependencies present in your legacy systems. This documentation capability helps bridge the knowledge gap, enabling you to make informed decisions as you transition your applications to modern cloud architectures.

To generate technical documentation

- In the left navigation pane, under Generate technical documentation, choose Select files and configure settings.
- 2. Select the files in the Amazon S3 bucket that you want to generate documentation for, and configure the settings in the Collaboration tab.



Note

Selected files should have the same encoding type (that is, all in the same CCSID -UTF8 or ASCII). Otherwise, generated technical documentation might have empty fields or sections.

- Choose the documentation detail level: 3.
 - Summary Provides a high-level overview of each file in the scope. Also, gives a one-line summary of each file.
 - **Detailed functional specification** Provides comprehensive details for each file in the mainframe application transformation scope. Some details include logic and flow, dependencies, input and output processing, and various transaction details.



Note

Documentation can be generated only for COBOL and JCL files.

- Choose Continue. 4.
- 5. Once AWS Transform generates documentation, review the documentation results by following the Amazon S3 bucket path in the console, where the results are generated and stored.
- Once the documentation is generated, you can also use AWS Transform chat to ask questions about the generated documentation and decide the next steps.

Add user information into the documentation with user's glossary file, a pdf configuration file and user logo files

```
ARTIFACT_ID.zip
### app/
    ### File1.CBL
    ### File2.JCL
    ### subFolder/
        # File3.CBL
### glossary.csv
### pdf_config.json
### header-logo.png
### footer-logo.png
# ...
```

Optional files can be added in the zip file to help improve the generated documentation quality and provide customized PDF cover page. Some of these can be:

• glossary.csv file: You can choose to provide and upload an optional glossary in the zip file in the S3 bucket. The glossary is in CSV format. This glossary helps create documentation with relevant descriptions in line with the customer vocabulary. A sample glossary.csv file looks like:

```
LOL, Laugh out loud
ASAP, As soon as possible
WIP, Work in progress
SWOT, "Strengths, Weaknesses, Opportunities and Threats"
```

• **pdf_config.json**: You can leverage this optional configuration file to generate PDF documents which align with their company's formats and standards, including headers, footers, logos, and customized information. A sample pdf_config.json looks like:

```
{
    "header": {
    "text": "Acme Corporation Documentation",
    "logo": "header-logo.png"
  },
  "customSection": {
    "variables": [
      {
        "key": "business Unit",
        "value": "XYZ"
      },
      {
        "key": "application Name",
        "value": "ABC"
      },
        "key": "xxxxxxxxxx",
        "value": "yyyyyyyyyyyy"
      },
      {
        "key": "urls",
        "value": [
          {
            "text": "Product Intranet Site",
            "url": "https://example.com/intranet"
          },
          {
            "text": "Compliance Policies",
            "url": "https://example.com/policies"
          }
        ]
      }
    ]
  },
  "footer": {
    "text": "This document is intended for internal use only. Do not distribute
 without permission.",
    "logo": "footer-logo.png",
    "pageNumber": true
```

}

Header:

- For the cover page PDF file, the default text will be the project name.
- For each program PDF file, the default text will be the program name.
- There is no default logo. If a header logo is not configured, no logo will be displayed.
- The font size and logo size shall be dynamically changed based on the number of words or logo file size.

Custom section:

- If the custom section is not configured, it will be omitted from the PDF.
- The link has to be click able.

Footer:

- There is no default text or logo for the footer.
- The page number will be displayed in the footer by default, unless explicitly configured otherwise.
- The font size and logo size shall be dynamically changed based on the number of words or logo file size.

Generate documentation inline viewer

You can view the PDF files in the generate technical documentation step.

To view the PDF files

- 1. Navigate to the **Review documentation results** tab.
- 2. Locate the PDF in the table listing generated PDFs.
- 3. Select the external link element next to the PDF.

The PDF will open in a new browser tab for you to access and read.



Note

AWS Transform also gives you the ability to download either an XML of PDF version of the generated technical documentation.

Important

If you're having issues with documentation inline viewer, make sure that the S3 bucket is set up correctly. For more information on S3 bucket's CORS policy, see the section called "S3 bucket CORS permissions".

Step 7: Extract business logic

In this step, you can extract essential business logic from your mainframe applications that are undergoing modernization. AWS Transform automatically analyzes your code to identify and document critical business elements, including detailed process flows, and business logic that is embedded in your applications. This capability serves multiple stakeholders in your modernization journey. Business analysts can leverage extracted logic to create precise business requirements and identify gaps or inconsistencies in current implementations. Developers gain the ability to quickly comprehend complex legacy system functionality without extensive mainframe expertise.

To extract business logic

- 1. In the left navigation pane, under **Extract business logic**, choose **Configure settings**.
- 2. In the **Collaboration tab** select how you want to extract business logic:
 - Application level: Generates business documents for all business functions, transactions, batch jobs, and files. This selects all of the files in the application.
 - **File level**: Generates business documents only for files you select from the file table.



Note

• For either option you can select Include detailed functional specification so that AWS Transform includes control flow and comprehensive business rules for the selected files.

 Selected files should have the same encoding type (that is, all in the same CCSID -UTF8 or ASCII). Otherwise, generated documentation might have empty fields or sections.

- Documentation can be generated only for COBOL and JCL files.
- For application level, programs used by CICS transactions and batch jobs are grouped together, while all other programs are categorized as *Unassigned*.

3. Choose Continue.

4. Once AWS Transform extracts business logic it stores the results in an Amazon S3 bucket in JSON format so that you can view them online.

Note

The number of generated business rule files might be larger than your initial selection. Some selected files may trigger business rule extraction to include additional dependent files, which will also appear in the results table.

Add user information into the documentation with user's glossary file

```
ARTIFACT_ID.zip

### app/

### File1.CBL

### File2.JCL

### subFolder/

# # File3.CBL

### glossary.csv

# ...
```

glossary.csv file: You can choose to provide and upload an optional glossary in the zip file in the source S3 bucket. Save the glossary as a CSV file named *glossary.csv*. The glossary helps creating documentation with relevant descriptions using the customer's vocabulary. A sample glossary.csv file looks like:

```
LOL, Laugh out loud
ASAP, As soon as possible
WIP, Work in progress
```

SWOT, "Strengths, Weaknesses, Opportunities and Threats"

View the extracted business documentation inline

You can view the business logic in the Extract business rule step. To do this,

- 1. Navigate to **Review business logic extraction results**.
- 2. Select the document you want to view from the table, and then click the **View result** button.

The business documentation page opens in a new browser tab.

Step 8: Decomposition

In this step, you decompose your code into domains that account for dependencies between programs and components. This helps the related files and programs to be grouped appropriately within the same domain. It also helps maintain the integrity of the application logic during the decomposition process.

- 1. Expand **Decompose code** from the left navigation pane.
- 2. Choose **Decompose into domains**.



Note

Two domains (unassigned and disconnected) are automatically created initially by the application. Unassigned domain is strictly under decomposition control and cannot be edited.

- Create a new domain by choosing Create domain from the AWS Transform prompt (for first domain only), or from under **Actions** menu.
- In **Create domain**, provide domain name, optional description, and mark some files as seeds. Seeds are elements that are labeled with business features or functions for AWS Transform to group related components into domains. For more information about seeds, see Seeds.
 - CICS configured files (CSD) and scheduler configured files (SCL) can be used for automatic seed detection.

Step 8: Decomposition



Note

You can also set one domain only as a common component. The files in this domain are common to multiple domains.

5. Choose Create.



Note

You can create multiple domains with different files as seeds.

- After confirming all domains and seeds, choose **Decompose**. 6.
- 7. AWS Transform will check the source code files and then decompose into domains with programs and data sets with similar use cases and high programming dependencies.

AWS Transform gives you a tabular and graph view of decomposed domains as dependencies. Graph view has two options:

- Domain view Can view how different domains are related to each other in visual format.
- **Dependency view** Can view all files in each domain as a complex dependency graph. If a node that was added to a domain didn't receive information from a seed in the same domain, then this node will either be predicted into unassigned (node didn't receive any information), disconnected (in a sub graph that didn't receive seed information) or into another domain (node received information from at least that domain).

Repeat these steps to add more domains or to reconfigure your already created domains with a different set of seeds if you don't like current domain structure.

When completed, choose **Continue**.

Seeds

Seeds are the foundational inputs for the decompose code phase. Each component or file (e.g., JCL, COBOL, Db2 tables, CSD, and scheduler files) can be assigned as a seed to only one domain, ensuring clear boundaries and alignment during the decomposition process.

The identification of the seeds depends on the structure of the application or portfolio. In the case of a typical mainframe legacy application, seeds can often be determined by adhering to

Step 8: Decomposition 62

established naming conventions, batch-level grouping in the scheduler, and transaction-level grouping defined in the CICS system. Additionally, database tables can also serve as seeds, providing another layer of structure for decomposition.

Import and/or update dependencies files

During decomposition, you can upload a JSON file for the dependencies that replaces the existing files generated by the dependencies analysis AWS Transform performs.

Export dependencies function allows you to download the dependencies json file generated in the decomposition step. After downloading, you can modify the file per your requirement. Then, you can import dependencies using the AWS Transform's upload functionality which allows you to upload the new JSON file of the dependencies that replaces the file generated by the dependencies analysis. After that, the graph in the decomposition step will be updated.

To export, modify, and import dependencies

- On the View decomposition results page, choose Actions. 1.
- 2. In the dropdown list, choose **Update dependencies file** option under **Other actions**.
- 3. In the **Update dependencies file** modal,
 - Download the dependency file AWS Transform created from the existing analysis results. a.
 - In the downloaded file, modify the dependencies based on what you want to achieve. b.
 - After modifying, save and upload this file using the **Upload dependency file** button. c.



Note

The only accepted file format is JSON file.

Next, choose **Import**.

AWS Transform will import the dependency file and create a new dependencies graph based on your input.

Parent/child/neighbor files

In a dependencies graph, programs relate to each other through different types of connections. Understanding these relationships helps you analyze program dependencies during transformation

63 Step 8: Decomposition

of your mainframe applications. It also helps with understanding the boundaries of a domain. For example, if you select a domain, and then select parent one level, it will show you the connected nodes.

Parent relationships – A parent file calls or controls other programs. Parents sit above their dependent programs in the hierarchy. You can select parent at one level or at all levels.

Children relationships – A child file is called or controlled by the parent program. Children sit below their parent in the file hierarchy.

Neighbor relationships – Neighbors are files at the same hierarchical level. They share the same parent program and might interact with each other directly.

Step 9: Migration wave planning

Based on the domains you created in the previous step, AWS Transform generates a migration wave plan with recommended modernization order.

- To view the planning results, choose **Plan Migration Wave**, and then choose **Review Planning** Results.
- 2. Review the domain wave plan (either in a table view or a chart view).
- You can either choose to go with the recommended migration wave plan generated by AWS Transform or add your preference manually by importing a JSON file.



Note

You can choose to migrate multiple domains in a single wave.

- 4. (Optional) If you decide to manually adjust migration wave plan, AWS Transform generates a new migration wave plan per your preference. You can also adjust the domains in each wave as required by choosing **Add preference** and then, **Add and regenerate**.
- After verifying, choose **Continue**.

If you're satisfied with this migration plan, you can move next step for refactoring the code. If you need to adjust the preference, you can follow these steps again.

Step 10: Refactor code

In this step, AWS Transform refactors the code in all or selected domain files into Java code. The goal of this step is to preserve the critical business logic of your application while refactoring it to a modernized cloud-optimized Java application.

- 1. Navigate to **Refactor code** in the left navigation pane, and choose **Domains to migrate**.
- 2. Select the domains you want to refactor.
- Choose Continue. You can track the status of refactoring domains (and files in it) using the worklog. AWS Transform will do the transformation of the mainframe code, and generate results without any manual input.
- 4. After refactoring completes, it will change the status to Completed in the worklog. You can view the results of refactored code by going to the Amazon S3 bucket where the results are stored. Each domain will provide a status for **Transform** (with each file), and **Generate** and will be marked as Done.



Along with the refactored code, your S3 bucket will also have the AWS Blu Age Runtime to

You might also see certain domains that have a Done with issues status. Expand those to see files showing a Warning status or an Error status. You can view the issues for the Warning and Error files, and choose to fix them for better refactoring results. Additional guidance for fixing these errors and warnings can be found in the console by viewing each of these files.

File transformation status

be compiled.

After your refactoring completes, AWS Transform will give you transformation status for all your files. These may include:

Ignored – AWS Transform will also give you the Ignored files after the code refactor. These are the files that are ignored during refactoring and haven't been included in the transformation.

Missing – Missing files are not included during the refactoring and transformation. These should be added again as a part of the source input in Amazon S3 bucket for better and cohesive results. AWS Transform will give you the number and information of missing files in the console.

Step 10: Refactor code 65

Pass through – Pass through files are not modified during the refactoring step, and do not go through any transformation. This status is useful for the Refactoring action which may not have changed the file depending on the configured refactoring.

Fatal – An unexpected error occurred during the transformation of this file.

Error – An error occurred during the transformation of this file and these files need to go through refactoring again.

Warning – The transformation generated all expected outputs for this file, but some elements might be missing or need additional input. Fixing these and running the refactoring steps again would give you better transformation results.

Success – The transformation generated all expected outputs for this file and it has detected nothing suspicious.

Custom transformation configuration

Refactor transformation allows you to change and/or modify configuration to improve the results of transformation.

To customize your transformation configuration

- 1. In **Refactor code** section, go to **Configure transformation** under Select domains.
- 2. In **Configure refactor** modal, specify the **Refactor engine version** (e.g. 4.6.0) which will be used to compile and run the generated application. For more information on available engine versions, see AWS Blu Age release notes.
- Add your project name, root package, and target database. The target database is target RDMS for the project.
- 4. Under **Legacy encoding**, define the default encoding for your files (e.g., CP1047). And mark the check boxes next to **Export Blusam masks** and **Specify generate file format**. You can also choose to specify conversion table encoding file format.
- 5. Review all you changes. Then, choose **Save and close**.

This will allow you to reconfigure your code with the new specified properties.

Step 10: Refactor code 66

Reforge code

Reforge uses Large Language Models (LLMs) to improve the quality of refactored code. The initial COBOL-to-Java transformation preserves functional equivalence while retaining COBOL-influenced data structures and variable names from the legacy system. Reforge restructures this code to follow modern Java practices and idioms, replacing COBOL-style constructs with native Java collections and naming conventions. This makes the code more readable and maintainable for Java developers.



Note

Quotas for reforge are:

- 3,000,000 lines of code per job
- 50,000,000 lines of code per user per month

Reforge your code after refactoring by following these steps:

- Choose **Reforge java code** in the left navigation pane and then select **Configure code reforge**. 1.
- 2. Provide the S3 location to your zipped buildable source project and choose **Continue**. Use this zip structure:

```
input.zip
        ### PROJECT-pom
        ### PROJECT-entities
        ### PROJECT-service
        ### PROJECT-tools
        ### PROJECT-web (optional)
        ### pom.xml
```

AWS Transform analyzes your zip package to locate files within the PROJECT-service directory so that it can provide a selectable list of classes that you can reforge. These classes have the suffix ProcessImpl.java.

- Complete the **Select classes to reforge** page and choose **Continue**. Track the reforge status on the Worklog tab.
- View the results of your completed reforge on the View results page, which displays the reforge status per class. It also specifies where to find the Reforge result in your S3 bucket.

Step 10: Refactor code 67

Once AWS Transform gets this input from you it gives you a downloadable file with the **Reforge** results.

This is the zip structure resulting from a successful reforge:

```
reforge.zip
### maven_project
### reforge.log
###tokenizer_map.json
```

- maven_project contains the reforged source code.
 - · Files that have been refactored but whose compilation was not successfully finalized are located at /src/main/resources/reforge/originalClassName.java.incomplete and are named originalClassName.java.incomplete. Compare these to the original versions of the files to choose reforged functions you want to save.
 - Source files provided to AWS Transform that were refactored successfully are backed up to src/main/resources/reforge/originalClassName.java.original and are named originalClassName.java.original. The refactored versions of the files replace the source files provided to AWS Transform.

Note

The originalClassName. java files are replaced with the reforged files only if the reforging process is successful. Otherwise, they retain the original content.

- reforge.log contains logs that you can use to diagnose job failures or provide to AWS support in case of an issue.
- tokenizer_map.json contains a mapping of token IDs to your data, such as file paths and class/ method names, that are tokenized in the logs for privacy protection. You can provide this file to AWS support in case of an issue.

Step 11: Re-run the job

With re-run capabilities you can restart an in-progress job, re-run a previously completed job, or modify job objectives. When you initiate a re-run through either the **re-run** button or the chat interface, you can choose to restart the entire job plan or select specific steps to re-run. AWS

Step 11: Re-run the job

Transform automatically retains the results of completed steps from the previous run, except those for dependent steps, which also run again. For example:

- If you choose to re-run **Analyze code**, all of the other steps are also re-run.
- If you only re-run Generate technical documentation or Extract business logic, other steps are not re-run.
- If you re-run **Decomposition**, then **Migration wave planning** and **Refactor code** are also rerun.



Note

When a rerun has been initiated for any step, AWS Transform compares the source code used during the analysis step in the original job with the source code currently available in the resource location specified when the job was created. If AWS Transform finds that the source code has been updated then it requires the **Analyze code** step, which necessitates a re-run of all of the subsequent steps.

You can download certain assets to preserve progress. For example, you can download the classification file from the Analyze code step to retain manual classifications. In the Decomposition step, you can download dependency updates, and domain and seed files, to retain previously created domains. This allows for a more iterative approach, enabling you to refine your work as needed throughout the transformation process. Be careful about using artifacts from previous jobs if the source code has been updated as this can introduce inconsistencies to the job.

When all the steps are successfully completed the left navigation pane displays each job task completed in green.

Step 12: Deployment capabilities in AWS Transform

AWS Transform helps you set up cloud environments for modernized mainframe applications by providing ready-to-use Infrastructure as Code (IaC) templates. Through the AWS Transform chat interface, you can access pre-built templates that create essential components like compute resources, databases, storage, and security controls. The templates are available in popular formats including AWS CloudFormation (CFN), AWS Cloud Development Kit (AWS CDK), and Terraform, giving you flexibility to deploy your infrastructure.

These templates serve as building blocks that reduce the time and expertise needed to configure environments for your modernized mainframe applications. You can customize these templates to fit your needs, giving you a foundation to build your deployment environment.

To retrieve the IaC templates, ask in the AWS Transform chat for the Infrastructure-as-Code templates clarifying your preferred modernization pattern (such as AWS Blu Age Refactor), your preferred topology (standalone vs high availability), and your preferred format (CloudFormation vs Cloud Development Kit vs Terraform).

Build and deploy your modernized application post-refactoring

After you complete the refactoring process with AWS Transform, you can build and deploy your modernized Java application. This guide walks you through retrieving your modernized code, configuring your environment, and deploying and testing your application.



Note

In addition to the guidance provided here, the AWS Transform generated code package will include an Set up the AWS Automated Refactor Development Environment document which provides instructions to set up a IDE (Integrated Development Environment) with Developer Runtime.

Topics

- Prerequisites
- Step 1: Retrieve the modernized code
- Step 2: Build the modernized application
- Step 3: Configure the test environment
- Step 4: Deploy the modernized application
- Step 5: Test the modernized application
- Additional example

Prerequisites

Before you begin, make sure you have:

- Successfully completed a refactoring job with AWS Transform.
- Access to the Amazon S3 bucket containing your modernized code. You can find this path on the console under Refactor code → View results or see Step 1: Retrieve the modernized code.
- Installed and configured build software tool stack on your development machine, such as <u>Apache Maven</u> or <u>Apache Tomcat</u>. For more information on Runtime versioning, see <u>AWS Blu Age release</u> notes.
- Installed and configured Amazon Corretto or a version of Java runtime. For more information on installing Amazon Corretto, see Amazon Corretto 24.
- Access to create and configure Amazon Aurora PostgreSQL databases for Runtime components, if necessary. For more information on Creating the Aurora PostgreSQL database, see <u>Working</u> with Amazon Aurora PostgreSQL.
- Administrative access to deploy applications to your runtime environment.
- Reviewed the <u>AWS Blu Age Runtime concepts</u> for fundamental concepts on applications modernized with AWS automated refactoring solution.

Step 1: Retrieve the modernized code

To retrieve the modernized code

- Navigate to your Refactor code → View results page on the console and locate the S3 path containing your generated code.
- 2. Download and extract the generated code package.
- 3. Open codebase/app-pom/pom.xml and note the required runtime engine version. For example <gapwalk.version>4.6.0</gapwalk.version>.
- 4. Locate the *Set up the AWS Automated Refactor Development Environment* document from the downloaded code package for reference.

Step 2: Build the modernized application

To build your modernized application

- 1. Access the runtime version from a dedicated Amazon S3 bucket on the AWS account used with AWS Transform:
- 2. Download and install the appropriate runtime version (identified in the section called "Step 1: Retrieve the modernized code") on your local development machine.



Note

Additional information for installing the runtime dependencies on your local machine is available in section 3.1 of the Set up the AWS Automated Refactor Development Environment document.

- Open the command prompt and navigate to your application's root directory. 3.
- To build deployable packages for the modernized application run the Maven build command: 4.

```
mvn package
```

Refer to the Application Organization page for details on the basic organization of the modernized code. For instance, for modernized application containing a front-end web application, you may expect at-least the following deployable .war aggregates in addition to the runtime components:

• Service project: Contains legacy business logic modernization elements

```
<business-app>-service.*.war
```

• Web project: Contains the modernization of user interface-related elements

```
<business-app>-web.*.war
```

Step 3: Configure the test environment

To configure your test environment

1. Configure your modernized application runtime. For more information, see the Set up configuration for Runtime section in the AWS Mainframe Modernization user guide.



Note

Refer section 5 of the Set up the AWS Automated Refactor Development Environment guide for runtime component specific configuration examples.

2. Prepare input and output (I/O) data sets for modernized applications. Modernized applications may process sequential I/O data sets, VSAM data sets, or others.



(i) Note

Refer section 6 of the Set up the AWS Automated Refactor Development Environment for examples.

- 3. A runtime environment You can use your existing runtime environment or create a new runtime environment.
 - To configure a non-managed runtime environment, see Set up a non-managed application.
 - To configure a managed runtime environment, see Set up a managed application.

After configuring the test environment, you move to the next step of deploying the modernized application.

Step 4: Deploy the modernized application

Deploy the application artifacts in the runtime you created and/or configured in the the section called "Step 2: Build the modernized application" and the section called "Step 3: Configure the test environment" sections.

Additional guidance for deploying the modernized application can be found using these links:

- Deploy on Amazon EC2
- Deploy on containers on Amazon ECS and Amazon EKS
- Create an AWS Mainframe Modernization application

Step 5: Test the modernized application

After deployment,

- 1. Review the available Runtime APIs for ways to interact with the modernized applications.
- 2. Test your application to align its functional equivalence with legacy application. For example, see Test a sample application in the AWS Mainframe Modernization user guide.

Additional example

For a specific example of modernizing mainframe application with AWS Transform, see <u>Modernize</u> the CardDemo mainframe application.

Additional example 74

Modernizing .NET with AWS Transform

The AWS Transform agent for .NET can help you modernize your .NET applications to be compatible with cross-platform .NET. This capability is called .NET modernization. After <u>Setting up</u> your workspace in AWS Transform, you can create a .NET modernization transformation job.

Capabilities and key features

- Analyze .NET Framework codebases from your source control systems, which includes private NuGet support, identifying cross repository dependencies, and providing an analysis report.
- Automated transformation of legacy .NET Framework applications to cross-platform .NET, email notifications, and a transformation summary report.
- Seamless integration with the source control platforms (BitBucket, GitHub, and GitLab) to ingest existing code and commit transformed code to a new branch.
- Validation of transformed code through unit tests.

Supported versions and project types

AWS Transform supports transformation for these versions of .NET:

- .NET Framework 3.5+
- .NET Core 3.1
- .NET 5.x+
- .NET 8.0

AWS Transform supports transformation of these types of projects:

- Libraries
- Applications written in C#
- Console applications
- Web API (ASP.NET) Web API
- Business Logic Layers of SPA (Single Page Application) backends
- Model View Controller (MVC) applications including front-end Razor views

Capabilities and key features 75

- Windows Communication Foundation (WCF) services
- Unit test projects (NUnit, xUnit, and MSTest)
- Projects with provided cross-platform versions for third-party or private NuGet packages. If a cross-platform equivalent is missing or unavailable, AWS Transform .NET will attempt a besteffort conversion.

Limitations

For more information on quotas and limitations for AWS Transform, see <u>Quotas for AWS</u> Transform.

AWS Transform does not transform the following:

- · WebForms (.aspx), WinForms, Blazor UI components
- Win32 DLLs that don't have core compatible libraries
- Applications written in F# or VB.NET
- Applications already in .NET 8.0+
- Repositories that do not contain any solutions.
- AWS Transform will not modify the original repo branches, and can only write to a separate target branch specified in your transformation plan.

Human intervention

During the porting of .NET Framework applications to cross-platform .NET, you may be requested to provide input or approvals in the following scenarios:

- Set up a connector to your source code and permissions
- Validate the proposed modernization plan
- Upload missing package dependencies as NuGets
- Review and accept the transformed code

More information

You can modernize your .NET code by using either the AWS Transform web application or the AWS Toolkit for Visual Studio.

Limitations 76

- Modernizing your .NET code by using the AWS Transform web application
- · Modernizing .NET in the IDE

Modernizing your .NET code by using the AWS Transform web application

AWS Transform for .NET is a new generative AI-powered agent designed to modernize legacy .NET applications. You can modernize your legacy .NET code by using the AWS Transform web application or the Visual Studio AWS Toolkit extension.

To modernize your .NET code using the AWS Transform web application, navigate to the web application, then see our <u>Modernizing .NET with AWS Transform quick start guide</u> or follow these detailed steps:

- 1. Creating the AWS Transform .NET job plan
- 2. Creating a source code repository connector
- 3. Confirming your repositories to prepare for transformation
- 4. Resolving package dependencies to prepare for transformation
- 5. Reviewing your plan to prepare for transformation
- 6. Transforming your .NET code

Modernizing .NET with AWS Transform quick start guide

.NET step 1: Sign-in and onboarding

- 1. Follow the steps under Quick start: Getting started with AWS Transform.
- 2. Follow the steps under Setting up your workspace or Getting started with AWS Organizations.

.NET step 2: Job creation

- 1. On your workspace landing page, choose to create a .NET job.
- 2. In the chat window, AWS Transform will ask you to confirm job details.

For more information, see Creating the AWS Transform .NET job plan.

.NET web app 77

.NET step 3: Set up a connector

In order for AWS Transform to assess your code and identify the jobs that can be transformed automatically, you must set up a connector to your repositories.

For .NET transformation, AWS Transform supports connectors to repositories of the following type:

- Bitbucket
- GitHub
- GitLab

AWS Transform also needs access to a writable branch in the same repository for submitting the transformed code.

If necessary, chat with AWS Transform in the left pane. It will walk you through setting up your connectors step by step.

This may involve the following steps:

- Creating a separate AWS account for importing your codebase.
- Identifying that AWS account.
- (Required) Adding the Bitbucket, GitHub, or GitLab app to your instance of AWS CodeConnections.
- (Required) Creating an AWS CodeConnections connection with your data source.
- Identifying that connection.
- Asking your AWS administrator to validate your connection in the AWS CodeConnections Console.
- Asking your AWS account administrator to assign an IAM role to the workspace, allowing it to use the connection.
- Confirming to AWS Transform that you are ready to begin the data transfer.

For more information about AWS CodeConnections, see <u>What are connections?</u> in the *Developer Tools Console User Guide*.

For more information about IAM roles, see <u>IAM roles</u> in the *AWS Identity and Access Management User Guide*.

Limits:

- AWS Transform does not support connectors to AWS CodePipeline
- AWS Transform can only connect to source control using an App ID. AWS Transform cannot connect to a source with a username and password.
- You cannot upload your source code files directly to AWS Transform. You must put them in a supported repository for AWS Transform to access.

When you set up a connector, the administrator of the AWS account to which you are connecting must accept the connection. In order to accept the connection, they must have the permissions given in the connector acceptance policy.

For more information, see Creating a source code repository connector.

.NET step 4: Analysis

In this step, AWS Transform analyzes the code and proposes a modernization plan, outlining the intermediate steps and tasks required to transform the application to .NET 8.0+.

Once the connector is set up, AWS Transform begins to automatically analyze the source code repositories (repos) to identify a list of repos that have supported project types for porting. Each repo may contain multiple .NET projects. By assessing all the repos and projects, the transformation agents for .NET can identify dependencies between .NET projects across multiple repos to help transform your code successfully.

When the analysis is completed, AWS Transform will provide you with a list of repositories, the number of .NET projects within each of these repos, the default branch to select for the transformation, and the last commit date and time.

By default, AWS Transform selects all .NET projects that are supported within a repo, and you have the option to select specific .NET projects, solutions, and branches to include or exclude from the transformation.

Also, as a part of your transformation job, the AWS Transform .NET agent has the following **Default settings**.

Exclude .NET Standard projects from the transformation plan

This setting is selected by default. When selected, AWS Transform excludes any .NET Standard projects from the transformation plan. If you deselect this setting, .NET Standard projects will be transformed, which will make them no longer compatible with the .NET Framework.

• Transform Model-View Controller (MVC) Razor Views to ASP.NET Core Razor

This setting is selected by default. When selected, AWS Transform handles the transformation of any MVC Razor View UI layers into ASP.NET Core Razor Views. If you deselect this option, you must transform these UI layers manually.

For the UI layer, only the transformation of MVC Razor Views to ASP.NET core is supported. UI Layers for WebForms (.aspx), Windows Presentation Foundation (WPF), WinForms, and Blazor UI components must be transformed manually.

Once the repo and .NET projects are selected, AWS Transform automatically begins the transformation process.

Legacy versions of .NET supported for transformation to .NET 8.0+:

- .NET Framework versions 3.5+
- .NET Core 3.1, .NET 5
- .NET 6
- .NET 7

For information about the .NET project types that AWS Transform supports, see <u>AWS Transform supported project types</u>. For information about unsupported project types, see <u>AWS Transform limitations for .NET modernization</u>.

For more information see the following:

- Confirming your repositories to prepare for transformation.
- Resolving package dependencies to prepare for transformation.
- Reviewing your plan to prepare for transformation.

.NET step 5: Bulk transformation

Once you have selected the repo and projects to be transformed, AWS Transform will automatically begin the transformation of the related .NET applications. AWS Transform downloads the source code into a Managed Development Environment (MDE), and encrypts it using your managed KMS keys. Then, AWS Transform builds a dependency tree for the jobs across the repos being modernized. Based on the dependency tree, the agents will start the transformation in parallel across the repo. Along the way, AWS Transform will ask you for input when it needs information, or when it needs you to take some action.

You can track the progress of the transformation in two ways:

- Worklog This provides a detailed log of the actions AWS Transform takes, along with human input requests, and your responses to those requests.
- *Dashboard* This provides high level summary of the transformation. It shows metrics on number of jobs transformed, transformation applied, and estimated time to complete the transformation.

Safeguards:

AWS Transform will refuse questions from users who don't have the proper permissions. For example, a read-only user cannot cancel a job transformation or delete a job.

For more information see Transforming your .NET code.

.NET Step 6: Code review and completion

At this point, either your jobs have been transformed successfully, or they have been partially transformed, with build errors.

In this step, you transition from the AWS Transform web experience to AWS Transform in the Visual Studio IDE. You can use AWS Transform in Visual Studio to verify the transformation of the projects, and to make modifications if required.

For information about setting up the AWS Transform extension with Visual Studio, see Modernizing .NET using AWS Transform in Visual Studio.

There are two possible scenarios for review, and user input varies depending on the scenario:

• The job is fully transformed – AWS Transform has fully transformed a job. The customer can review this transformed code, and if they are satisfied with the change, they can then proceed to Complete the transformation. This prompts an input response required action for the Code approver or the Administrator persona to review this action. Once the administrator approves, AWS Transform marks the job transformation status as Completed.

• The job is partially transformed – AWS Transform has partially transformed a job, and the job has build errors that require a human in the loop (HITL) action. For this scenario, you can review the build errors and manually address any issues. After the Administrator has reviewed and approved the code, AWS Transform will continue the transformation and update the build errors for the job. You can continue to track this progress and take further action as required until all build errors are resolved.

Creating the AWS Transform .NET job plan

After you create your workspace, on the **Jobs** tab, select **Create a job with AWS Transform**. Then follow the prompts from AWS Transform in the chat pane, using natural language. The following are the typical steps to creating a .NET modernization job.

- 1. AWS Transform will ask you which type of transformation job you would like to create. In the chat, enter .NET modernization.
- 2. AWS Transform will suggest a job name and ask you if you want to change the job name. If you would like to change the job name, let AWS Transform know using natural language, such as, *change the job name to ExampleCorpDotNet1*. Otherwise, in the chat, you can accept the suggested job name.
- 3. AWS Transform creates the transformation job.

After you accept the job name, AWS Transform notifies you in the chat window that it is creating the job.

Components of the AWS Transform .NET job plan

The AWS Transform .NET job plan in the web app has a left side bar that lists the phases of the job plan. It also has a right pane that shows the details. These phases include:

Left pane

Create .NET job 82

1. **Get resources to be transformed** In this phase, you create a connector to your code repository using AWS CodeConnections. Depending on your repository permissions, an admin of the code repository may need to approve the connector and give AWS Transform access to the repository.

- 2. **Discover resources for transformation** In this phase, AWS Transform assesses your repository.
- 3. **Prepare for transformation** In this phase, AWS Transform notifies you if any dependencies are missing from your repositories. You can upload the missing dependencies or ignore them. If you are not an admin for the repo, an admin may need to approve the final transformation plan.
- 4. **Transform** In this phase, AWS Transforms your repo and provides you the ongoing status during the transformation until it's completed.

Right pane

In the top right section, you can select the Region of the transformation job. Make sure that the job Region has the same Region as your AWS CodeConnections connector to your third party repository. For a list of supported Regions, see AWS Transform Regions.

You can also see the job status:

- Awaiting user input
- Time elapsed
- Running

You can also see the following icons:

- A stop transformation icon
- A refresh icon
- A settings icon

The right pane contains the following tabs:

Dashboard

The *Dashboard* provides high level summary of the transformation. It shows metrics on number of jobs transformed, transformation applied, and estimated time to complete the transformation.

Collaboration

Create .NET job

Use the *Collaboration* tab to establish a connection to your source code repository. AWS Transform supports Bitbucket, GitHub, and GitLab repositories for .NET modernization. You can create only one code repository connector per transformation plan. You must specify the AWS account that you would like to use to access your code repository by using AWS CodeConnections.

Worklog

AWS Transform logs its actions in the Worklog tab. The Worklog provides a detailed log of the actions AWS Transform takes, along with human input requests, and your responses to those requests.

Creating a source code repository connector

After Creating the AWS Transform .NET job plan, the left pane of the AWS Transform window lists the phases of the transformation job. The first phase is Get resources to be transformed. In this phase, you can *Invite collaborators* and *Connect a source code repository*. The right pane of the AWS Transform window shows the Collaboration tab.

Each transformation job is required to have only one source code repository connector. The AWS Transform agent uses this connector to download your .NET codebase from GitHub, GitLab, or Bitbucket by using AWS CodeConnections. You must set up AWS CodeConnections in the same Region as your AWS Transform job.



Note

If you have an existing source code repository connector, AWS Transform will notify you. Select **Use existing connector** to use this connector. If you do not wish to use the existing connector, see Deleting an existing connector before Adding a new connector. You can only have one source repository connector per job.

Adding a new connector

To create a new source code repository connector:

- 1. Enter the AWS account number that you would like to use for the AWS CodeConnections connector.
- 2. If you have not set up AWS CodeConnections for the same AWS account, Set up AWS CodeConnections.

Connect a source code repo

3. Use the AWS Developer Tools Console to create a connection to your Bitbucket, GitHub, GitHub Enterprise Server, GitLab, or AzureDevOps repository.

- 4. Copy the Amazon Resource Name (ARN) of the connection you created.
- 5. Return to AWS Transform, and paste the ARN of the connection you created.
- 6. Enter a name for the connector.



Note

Do not enter personal information as part of the connector name.

- 7. Select Initiate connector creation.
- 8. AWS Transform makes the request to create a connector for AWS account you entered, in the same Region as the current transformation job, and notifies you that an approval request is ready to be approved by your AWS administrator in the AWS Management Console. Select Copy the verification link.
- 9. If you are the AWS administrator, sign into the AWS account and go to the verification link to approve the connection request. If you are not the AWS administrator, provide the verification link to your AWS administrator.
- 10After the AWS administrator approves the connector request, select **Finalize connector** to complete the connector creation process. Should you wish to use a different connector, select **Restart**, to restart the connector creation process.

The Collaboration tab also includes details about your connector, which include:

- Status -- Approved or Pending approval
- AWS account ID
- AWS Region
- Connection ARN

Connect a source code repo

Deleting an existing connector



Note

If you have an existing source code repository connector, AWS Transform will notify you. Select either Use existing connector or Delete and create a new connector.

If you choose to delete an existing source code repository connect, AWS Transform will warn you before actually deleting the connector.

- 1. You can delete an existing connector a couple of ways:
 - a. If AWS Transform prompts you to, you can select **Delete and create a new connector**.
 - b. You can also select **restart** in the prompt, **To modify this connector, you must restart**.
- 2. AWS Transform warns you that restarting will delete the connector. To delete the connector, select **Restart** again.
- 3. AWS Transform warns you again that deleting the connector will remove the connection to your third party repository, such as Bitbucket, GitHub, and GitLab. Also, any AWS Transform jobs that are using this connector will fail if the connector is deleted. To confirm deletion, type delete and select **Delete**.
- 4. To add a new source code repository connector, see Adding a new connector.

Confirming your repositories to prepare for transformation

After Creating a source code repository connector, confirm your repositories as the first step of preparing for transformation. You can:

- Review and change Default Settings that affect the plan contents.
- Review discovered repositories and select which ones to assess for transformation.
- Review the repository assessment report and chat with AWS Transform about its contents.
- Use the plan generated by AWS Transform.
- Customize the transformation plan using the web app or by downloading a JSON file, modifying it, and uploading it to AWS Transform.

Confirm your repos

Default Settings

In the **Job details** section of the **Collaboration** tab, you can set the following:

Exclude .NET Standard projects from the transformation plan

This setting is selected by default. When selected, AWS Transform excludes any .NET Standard projects from the transformation plan. If you deselect this setting, .NET Standard projects will be transformed, which will make them no longer compatible with the .NET Framework.

Transform UI Model-View Controller (MVC) Razor Views to ASP.NET Core Razor

This setting is selected by default. When selected, AWS Transform handles the transformation of any MVC Razor View UI layers into ASP.NET Core Razor Views. If you deselect this option, you must transform these UI layers manually.

For the UI layer, only the transformation of MVC Razor Views to ASP.NET core is supported. UI Layers for WebForms (.aspx), MVC Views (.cshtml), Windows Presentation Foundation (WPF), WinForms, and Blazor UI components must be transformed manually.

Review the repository assessment report

The repository assessment report provides a tabular view of information about each repository detected by AWS Transform. This includes:

- · Repository Owner
- Assessed Branch
- · No. of solutions
- No. of projects
- Total Lines of Code
- Detected .NET Versions
- Detected Project Types
- No. of Private Nuget Dependencies
- No. of Public Nuget Dependencies
- Transformation Complexity

To download the file:

Confirm your repos 87

- 1. In your workspace, select the **Collaboration** tab.
- In the Choose repositories pane, select Download, and choose Download as HTML or Download as JSON

Chat about the repository summary report

Your repository summary may be very long. Use chat to better understand its contents. Your report is available for chat when you see this message on the **Worklog** tab: *Assessment report is now available in chat for queries*. To open the chat click the purple hexagonal icon in the lower right corner of the web console. Here are some example prompts:

- List all repositories discovered or assessed.
- How many .NET 6 projects were discovered in the assessment?
- Which projects have the highest complexity?

Use the plan generated by AWS Transform

You can choose to use the AWS Transform plan created from discovering repositories with a higher probability of transformation success, which includes the last modified, root, and cross-dependent repositories.

AWS Transform lists some high level details about the transformation plan, which include:

- *Total number of repositories discovered* This number of legacy .NET repositories that AWS Transform discovered using your source code repository connector. AWS Transform can scan a maximum of 1,000 repositories in each job. If you have more repositories to scan or transform, you can create multiple transformation jobs.
- Selected repositories The number of legacy .NET repositories that AWS Transform selected for transformation.
- Selected dependencies The number of dependencies that the selected repositories require, which AWS Transform auto-detected and added to the transformation plan.

The *Selected repositories* table lists the selected repositories. You can search repositories by name, navigate to additional pages of repositories, and download a JSON formatted file that lists the selected repositories and their dependencies. You can make any changes in this view.

The *Selected repositories* table lists the following details about the repositories that AWS Transform selected for transformation:

- Repository name
- Source branch
- · Last modified date and time
- Lines of code This allows you to see if you're approaching your quota limits. For more information, see Quotas for AWS Transform.
- Dependent repositories You can click on the number of dependent repositories to view more details about these dependencies.

If you are happy with the AWS Transform generated transformation plan, select **Confirm repositories**.

If you would like to make changes to the AWS Transform generated transformation plan, select Customize the transformation plan.

Customize the transformation plan

If you choose not to <u>Use the plan generated by AWS Transform</u>, you can customize the list of repositories to transform using one of the following options:

- 1. Select repositories in the **Selected repositories** table.
 - a. When you add a repository to the plan, AWS Transform will select the repository's dependencies for you and automatically add them to the transformation plan.
 - b. You can also deselect repositories in the table.
- 2. Download the AWS Transform generated JSON list of repos and branches.
 - a. Select **Download** and then select **Download as JSON**.
 - b. Review and modify the JSON. You can remove repositories, or solutions within repositories, that you don't want to transform.
 - c. Upload the revised JSON by selecting **Upload customized plan**.
 - d. If the JSON uploads successfully, the **Selected repositories** table displays the selections you made.
- 3. You can select **Show dependent repos** to display the list of repositories that AWS Transform discovered are dependencies for the repositories you selected.

4. You can repeat these steps if needed. When you are satisfied with your customized plan, select **Confirm repositories**.

Resolving package dependencies to prepare for transformation

After <u>Confirming your repositories to prepare for transformation</u>, if AWS Transform finds missing package dependencies, you must complete this step. You can run a Windows PowerShell script to get the missing package dependencies from the same device as your Visual Studio development environment, or you can retrieve the missing packages manually. Then, upload the missing packages.

AWS Transform lists the missing packages in the *Missing package dependencies* table. You can search for a missing package by name in the search box. This table includes the following details about the missing packages:

- Name
- Associated repositories
- Framework version status
- Core version status

To resolve the missing package dependencies, Upload the missing packages.

Upload the missing packages

1. If you choose, you can download a Windows PowerShell helper script to retrieve the missing package dependencies from within your Visual Studio development environment. Or you can find the missing packages manually.

To use the Windows PowerShell script:

- a. Select **Download Windows PowerShell script**.
- b. Run the script locally with an active connection to the repositories that contain the missing package dependency files.
- c. This script allows you to download the missing package dependencies to your local environment.
- 2. The script will create a single zip file for you to upload which includes all of the dependencies in one archive. You can also upload individual . nupkg or zip files for each dependency.

- 3. Select Upload package files.
- 4. In the **Upload dependency files** modal, select **Choose files** and browse to the location of the compressed missing package files on your device.
- 5. Select Upload.
- 6. AWS Transform validates the files you uploaded. During validation, you cannot make any updates. AWS Transform reports the validation status above the *Missing package dependencies* table.
- 7. AWS Transform also updates the status columns in the *Missing package dependencies* table from *Missing* to *Resolved*. If a package fails validation, its status becomes *Invalid*. For invalid files do the following:
 - a. In the Missing package dependencies table, select the invalid package using the check box.
 - b. Select Remove uploaded file.
 - c. This changes its status back to Missing.
- 8. After you have uploaded the missing packages and resolved the package dependencies, select **Proceed to review**.

If you select **Proceed to review** without resolving the missing package dependencies, AWS Transform asks if you would like to start the transformation job without the missing packages. If you select **Ignore the missing package dependencies**, AWS Transform will use assembly references to transform the code. Proceeding with this action can affect related resources.

Folder Structure Requirements

For NuGet uploads AWS Transform requires either:

- The .nuspec file to be placed at the root level for manually uploaded packages, or
- Use of the provided PowerShell script to generate the correct structure

Version matching requirements

- Framework status: AWS Transform attempts to match the version of the dependency specified
 in the .csproj file (source code) with the version of the uploaded dependency. An exact match
 results in a Success status.
- Core status: AWS Transform expects a core version specification in the dependencies section of the uploaded .nuspec file. If one is found then the core status is Success.

Example format:

```
<dependencies>
    <group targetFramework="net8.0" />
</dependencies>
```

Common versions include:

- .NET 5.0 (net5.0)
- .NET 6.0 (net6.0)
- .NET 7.0 (net7.0)
- .NET 8.0 (net8.0)

Reviewing your plan to prepare for transformation

After Confirming your repositories to prepare for transformation, and Resolving package dependencies to prepare for transformation, the AWS account administrator must review the transformation plan and approve it in AWS Transform.

AWS Transform displays the job's list of repositories, dependent repositories, and dependent packages that were selected for transformation.

Reviewing the transformation plan

AWS Transform displays the job's list of repositories, dependent repositories, and dependent packages that were selected for transformation.



Note

AWS Transform can transform a maximum of 100 dependencies and repositories per transformation plan.

- 1. If you are not the AWS account administrator, review the job plan, and if you accept it, select Send for approval.
- 2. If you are the AWS account administrator, you must review the plan, and when ready, approve the plan to start the transformation. After you review the job plan, select either:

92 Review transformation plan

a. *Reject* If the job was created by a user who is not the AWS account administrator, we suggest you notify the job creator to restart the job.

b. Approve and start transformation.

The job review includes the following details:

- 1. *Job summary* This includes:
 - a. The target branch where AWS Transform will place the transformed code.
 - b. The target .NET version, currently .NET 8.0.
 - c. The job settings:
 - i. Exclude .NET standard projects
 - ii. Transform MVC Razor Views to ASP.NET Core Razor Views
 - d. Number of repositories selected for transformation
 - e. Number of dependent repositories
 - f. Number of private NuGet packages
 - g. Total lines of code for the job
- 2. *Repositories selected* These are the repositories selected for transformation. They must be either MVC, Web, Windows Communication Foundation (WCF), Console, class library, UI framework Razor pages, or unit test packages. This table includes the following information:
 - a. Name
 - b. Source branch
 - c. Supported projects
 - d. Lines of code
 - e. Projects detected
 - f. Projects skipped
 - g. Dependencies detected
- 3. Dependent repositories added These are the dependent repositories added for transformation. They must be either MVC, Web, Windows Communication Foundation (WCF), Console, class library, UI framework Razor pages, or unit test packages. This table includes the following information:
 - a. Name

- c. Source branch
- d. Supported projects
- e. Lines of code
- f. Projects detected
- g. Projects skipped
- 4. Dependent packages These are the dependent packages added for transformation. They must be either MVC, Web, Windows Communication Foundation (WCF), Console, class library, UI framework - Razor pages, or unit test packages. This table includes the following information:
 - a. Name
 - b. Associated repositories
 - c. Framework version status
 - d. Core version status

Transforming your .NET code

After the AWS account administrator finishes Reviewing your plan to prepare for transformation, you can view the transformation progress on the *Dashboard* tab of your transformation job.



Note

In addition to transforming repositories and dependencies, AWS Transform can execute fully transformed (zero build errors) unit test projects. AWS Transform doesn't have access to unit test results prior to the transformation and can only share post-transformation results. You can then compare your baseline data, prior to transformation, with the posttransformation results to understand any potential gaps.

In the top right corner, you can see the job status, which has one of the following values:

- Awaiting user input
- Time elapsed
- Running

You can also see the following icons:

Transform code

- · A stop transformation icon
- A refresh icon
- A settings icon

The *Dashboard* provides a high level summary of the transformation. It shows metrics for the number of jobs transformed and transformation applied, and the estimated time to complete the transformation.

The Dashboard includes:

- 1. The transformation *Job details* section, which lists the default settings and details of the transformation job, including:
 - a. *Target branch destination* To transform your code, AWS Transform creates a new branch for the transformed code in your code repo.
 - b. Target .NET version, which is .NET 8.0
 - c. The AWS Transform job ID.
 - d. The job settings:
 - i. Exclude .NET standard projects
 - ii. Transform MVC Razor Views to ASP.NET Core Razor Views
- 2. The *Transformation summary* section contains:
 - a. The number of repositories selected for transformation
 - b. The number of projects to be transformed
 - c. The total lines of codes in these repositories and projects

After the transformation starts, pie charts appear in the *Repository status*, *Package status*, and *Unit test status* sections displaying progress in real time. The *Unit test status* shows the status of unit tests located in your repositories that AWS Transforms runs after transformation to test the transformed code. AWS Transform shares the executed test results, along with individual test name for customers to review the list of unit tests passed and failed.

Transform code 95



Note

The Repositories section lists the repositories that AWS Transform recommends or that you selected for transformation. Select **Download JSON** to download a list of repositories, dependencies, and packages in your transformation plan.

Review the transformation summary report

After a transformation finishes for a repository, or when the transformation job completes, you can download a transformation summary report.

Download the transformation summary report from the **Dashboard** tab. Select **Download**, and choose **Download as HTML** or **Download as JSON**. This report provides an overview of the transformation, and information specific to each repository that was transformed, including:

- Job Details
 - Transformation job overview
 - Transformation results overview
 - Files overview
- Repositories list, and for each repository:
 - repository_name overview
 - Solution Transformation Summary
 - Project Details, including Status, Type, .NET Version, Total Lines of Code, Transformed Lines of Code, and Files Changed

Chat with AWS Transform about the transformation report

You can chat with AWS Transform about the transformation report after a repository has been completely processed or after the transformation job is completed. The **Worklog** tab displays this message for each repository that is available for chat: Repository repository_name transformation details are now available in chat for queries. It displays this message when the transformation job is completed: AWS Transform transformation job is completed.

To open the chat click the purple hexagonal icon in the lower right corner of the web console.

Here are some example prompts:

Transform code

- Which projects were successfully transformed?
- Which projects were partially ported?
- What changes were made to the repo_name repository?
- What packages were upgraded in the project_name project?

Modernizing .NET in the IDE

AWS Transform for .NET is a new generative AI-powered agent designed to modernize legacy .NET applications. For more information, see Modernizing .NET with AWS Transform. You can modernize your legacy .NET code by using the AWS Transform web application or the Visual Studio AWS Toolkit extension. Use AWS Transform in integrated development environments (IDEs) to get assistance with your software development needs. In IDEs, AWS Transform includes capabilities to provide guidance and support across various aspects of .NET code modernization.

To transform a .NET solution or project, the AWS Transform agent analyzes your codebase, determines the necessary updates to port your application, and generates a transformation plan before the transformation begins. During this analysis, the AWS Transform agent divides your .NET solution or project into code groups that you can view in the transformation plan. A code group is a project and all its dependencies that together generate a buildable unit of code such as a dynamic link library (DLL) or an executable.

During the transformation, the AWS Transform agent provides step-by-step updates in the AWS **Transformation Hub** window where you can monitor progress. After transforming your application, AWS Transform generates a summary with the proposed changes in a diff view for you to optionally verify the changes before you accept them. When you accept the changes, AWS Transform makes in-place updates to your .NET solution or project.

AWS Transform performs four keys tasks to port .NET applications to Linux:

- Upgrades language version Replaces outdated C# versions of code with Linux-compatible C# versions.
- Migrates from .NET Framework to cross-platform .NET Migrates projects and packages from Windows dependent .NET Framework to cross-platform .NET compatible with Linux.
- Rewrites code for Linux compatibility Refactors and rewrites deprecated and inefficient code components.

.NET IDE 97

• Generates a Linux compatibility readiness report – For open-ended tasks where user intervention is needed to make the code build and run on Linux, AWS Transform provides a detailed report of actions needed to configure your application after transformation.

For more information about how AWS Transform performs .NET transformations, see <u>How AWS</u> Transform modernizes .NET applications.

Quotas

For AWS Transform .NET transformation quotas in the IDE, see Quotas for AWS Transform.

To modernize your .NET code using the AWS Transform, part of the Visual Studio AWS Toolkit extension, see the following:

- Modernizing .NET using AWS Transform in Visual Studio
- How AWS Transform modernizes .NET applications
- Troubleshooting issues with .NET transformations in the IDE

Modernizing .NET using AWS Transform in Visual Studio

Complete these steps to port a Windows-based .NET application to a Linux-compatible cross-platform .NET application with AWS Transform in Visual Studio.

Step 1: Prerequisites

Before you continue, make sure you've <u>downloaded the AWS toolkit extension in Visual Studio</u> and completed the steps to <u>Authenticate in Visual Studio</u>. You can log in with an AWS Transform credential or an Amazon Q Developer credential.

Also, see the AWS Transform <u>capabilities</u> and <u>limitations</u> for .NET modernization to make sure that your code can be transformed.

Step 2: Transform your application

To transform your .NET solution or project, complete the following procedure:

- 1. Open any C# based solution or project in Visual Studio that you want to transform.
- 2. Open any C# code file in the editor.
- 3. Choose **Solution Explorer**.

Transform .NET in Visual Studio 98

4. From the **Solution Explorer**, right click a solution or project you want to transform, and then choose to port or transform the solution or project with AWS Transform.

5. The AWS Transform window appears.

The solution or project you selected will be chosen in the **Choose a solution or project to transform** dropdown menu. You can expand the menu to choose a different solution or project to transform.

In the **Choose a .NET target** dropdown menu, choose the .NET version you want to upgrade to.

- 6. You can update the Default Settings, or leave them as-is. These include:
 - Exclude .NET Standard projects from the transformation plan

This setting is selected by default. When selected, AWS Transform excludes any .NET Standard projects from the transformation plan. If you deselect this setting, .NET Standard projects will be transformed, which will make them no longer compatible with the .NET Framework.

• Transform UI Model-View Controller (MVC) Razor Views to ASP.NET Core Razor

This setting is selected by default. When selected, AWS Transform handles the transformation of any MVC Razor View UI layers into ASP.NET Core Razor Views. If you deselect this option, you must transform these UI layers manually.

For the UI layer, only the transformation of MVC Razor Views to ASP.NET core is supported. UI Layers for WebForms (.aspx), MVC Views (.cshtml), Windows Presentation Foundation (WPF), WinForms, and Blazor UI components must be transformed manually.

Check the NuGet sources and get .NET compatible package versions

Allow AWS Transform to search for and get .NET compatible package versions for the code that you would like to transform.

- 7. Choose **Confirm** to begin the transformation.
- 8. AWS Transform begins transforming your code. You can view the transformation plan it generates for details about how it will transform your application.

A **Transformation Hub** opens where you can monitor progress for the duration of the transformation. After AWS Transform has completed the **Awaiting job transformation startup** step, you can navigate away from the project or solution for the duration of the transformation.

Transform .NET in Visual Studio 99

9. After the transformation is complete, navigate to the **Transformation Hub** and choose **View diffs** to review the proposed changes in a diff view.

10. Choose **View code transformation summary** for details about the changes AWS Transform made. You can also download the transformation summary by choosing **Download summary** as .md.

If any of the items in the **Code groups** table require input under the Linux porting status, you must manually update some files to run your application on Linux.

- a. From the **Actions** dropdown menu, choose **Download Linux readiness report**.
- b. A .csv file opens with any changes to your project or solution that you must complete before your application is Linux compatible. It includes the project and file that need to be updated, a description of the item to be updated, and an explanation of the issue. Use the **Recommendation** column for ideas on how to address a Linux readiness issue.
- 11. To update your files in place, choose **Accept changes** from the **Actions** dropdown menu.

How AWS Transform modernizes .NET applications

Review the following sections for details about how .NET transformation with AWS Transform works.

Analyzing your application and generating a transformation plan

Before a transformation begins, AWS Transform builds your code locally to verify if it is buildable and configured correctly for transformation. AWS Transform then uploads your code to a secure and encrypted build environment on AWS, analyzes your codebase, and determines the necessary updates to port your application.

During this analysis, AWS Transform divides your .NET solution or project into code groups. A code group is a project and all its dependencies that together generate a buildable unit of code such as a dynamic link library (DLL) or an executable. Even if you didn't select all project dependencies to be transformed, AWS Transform determines the dependencies needed to build your selected projects and transforms them too, so that your transformed application will be buildable and ready for use.

After analyzing your code, AWS Transform generates a transformation plan that outlines the proposed changes that it will make, including a list of code groups and their dependencies that will be transformed.

How it works 100

Transforming your application

To start the transformation, AWS Transform builds your code again in the secure build environment to verify if it is buildable remotely. AWS Transform then begins porting your application. It works from the bottom up, starting with the lowest level dependency. If AWS Transform runs into an issue with porting a dependency, it stops the transformation and provides information about what caused the error.

The transformation includes the following updates to your application:

- Replacing outdated C# versions of code with Linux-compatible C# versions
- Upgrading .NET Framework to cross-platform .NET, including:
 - Identifying and iteratively replacing packages, libraries, and APIs
 - Upgrading and replacing NuGet packages and APIs
 - · Transitioning to cross-platform runtime
 - Setting up middleware and updating runtime configurations
 - Replacing private or third-party packages
 - Handling IIS and WCF components
 - Debugging build errors
- Rewriting code for Linux compatibility, including refactoring and rewriting deprecated and inefficient code to port existing code

Reviewing transformation summary and accepting changes

After the transformation is complete, AWS Transform provides a transformation summary with information about the proposed updates it made to your application, including the number of files changed, packages updated, and APIs changed. It flags any unsuccessful transformations, including affected files or portions of files and the errors encountered during an attempted build. You can also view a build summary with build logs to learn more about what changes were made.

The transformation summary also provides a Linux porting status, which indicates whether or not additional user input is needed to make the application Linux compatible. If any of the items in a code group require input from you, you download a Linux readiness report that contains Windowsspecific considerations that AWS Transform could not address at build time. If input is needed for any code groups or files, review the report for details about what type of change still needs to be

How it works 101

made and, if applicable, for recommendations for how to update your code. These changes must be made manually before your application can be run on Linux.

You can review the proposed changes AWS Transform made in a diff view before accepting them as in-place updates to your files. After updating your files and addressing any items in the Linux readiness report, your application is ready to run on cross-platform .NET.

Troubleshooting issues with .NET transformations in the IDE

Use the following sections to troubleshoot common issues with .NET transformations in the IDE with AWS Transform.

How do I know if a job is progressing?

If AWS Transform appears to be spending a long time on a step in the **AWS Transformation Hub**, you can check whether the job is still active in the output logs. If diagnostic messages are being generated, the job is still active.

To check the outputs, choose the **Output** tab in Visual Studio. In the **Show output from:** menu, choose **Amazon Q Language Client**.

The following screenshot shows an example of the outputs AWS Transform generates during a transformation.

```
Output from: Amazon Q Language Client | Amazon Q
```

Why are some projects not selected for transformation?

AWS Transform can only transform supported project types in the C# language. Currently, AWS Transform does not support porting all UI layer components or projects written in the VB.NET or F# languages. For a list of supported project types and other prerequisites for transforming your .NET projects, see Step 1: Prerequisites.

How can I get support if my project or solution isn't transforming?

If you aren't able to troubleshoot issues on your own, you can reach out to Support or your AWS account team to submit a support case.

Troubleshooting 102

To get support, provide the transformation job ID so AWS can investigate a failed job. To find a transformation job ID, choose the **Output** tab in Visual Studio. In the **Show output from:** menu, choose **Amazon Q Language Client**.

How can I prevent my firewall from interfering with transformation jobs?

If your organization uses a firewall, it might interfere with transformations in Visual Studio. You can temporarily disable security checks in Node.js to troubleshoot or test what is preventing the transformation from running.

The environment variable NODE_TLS_REJECT_UNAUTHORIZED controls important security checks. Setting NODE_TLS_REJECT_UNAUTHORIZED to "O" disables Node.js's rejection of unauthorized TLS/SSL certificates. This means:

- Self-signed certificates will be accepted
- Expired certificates will be allowed
- Certificates with mismatched hostnames will be permitted
- Any other certificate validation errors will be ignored

If your proxy uses a self-certificate, you can set the following environment variables instead of disabling NODE_TLS_REJECT_UNAUTHORIZED:

```
NODE_OPTIONS = -use-openssl-ca
NODE_EXTRA_CA_CERTS = Path/To/Corporate/Certs
```

Otherwise, you must specify the CA certs used by the proxy to disable NODE_TLS_REJECT_UNAUTHORIZED.

To disbale NODE_TLS_REJECT_UNAUTHORIZED on Windows:

- 1. Open the Start menu and search for **Environment Variables**.
- 2. Choose Edit the system environment variables.
- 3. In the **System Properties** window, choose **Environment Variables**.
- 4. Under **System variables**, choose **New**.
- 5. Set Variable name to NODE_TLS_REJECT_UNAUTHORIZED and Variable value to 0.
- 6. Choose **OK** to save the changes.
- 7. Restart Visual Studio.

Troubleshooting 103

Security in AWS Transform

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 <u>shared responsibility model</u> 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 <u>AWS</u> <u>Compliance Programs</u>. To learn about the compliance programs that apply to AWS Transform, 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 Transform. The following topics show you how to configure AWS Transform 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 Transform resources.

Topics

- Data protection in AWS Transform
- Identity and access management for AWS Transform
- Compliance validation for AWS Transform
- · Resilience in AWS Transform
- AWS Transform and interface endpoints (AWS PrivateLink)

Data protection in AWS Transform

The AWS <u>shared responsibility model</u> applies to data protection in AWS Transform. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the

Data protection 104

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 Security Blog.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into <u>tags</u> or free-form text fields such as a **Name** field. This includes when you work with AWS Transform or other AWS services using the AWS Management Console, API, AWS Command Line Interface (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. For more information about how AWS Transform uses content, see <u>AWS Transform service improvement</u>.

AWS Transform stores your questions, its responses, and additional context, such as console metadata and code in your IDE, to generate responses to your questions. For information about how data is encrypted, see Data encryption in AWS Transform. For information about how AWS may use some questions that you ask AWS Transform and its responses to improve our services, see AWS Transform service improvement.

In AWS Transform, your data is stored in the AWS Region where your AWS Transform profile was created.

Data protection 105

With cross- inferencing, your requests to AWS Transform may be processed in a different Region within the geography where your data is stored. For more information, see Cross-Region inference.

Topics

- Data encryption in AWS Transform
- AWS Transform service improvement
- Cross-region processing in AWS Transform

Data encryption in AWS Transform

This topic provides information specific to AWS Transform about encryption in transit and encryption at rest.

AWS Transform provides encryption by default to protect sensitive customer data at rest with encryption using AWS owned keys.

Encryption types

AWS Owned Keys (Default)



Note

AWS owned keys — AWS Transform 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 or change any programs to protect the keys that encrypt your data. For more information, see AWS owned keys in the AWSKey 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, it enables you to build secure applications that meet strict encryption compliance and regulatory requirements.

While you can't disable this layer of encryption or select an alternate encryption type, you can add a second layer of encryption over the existing AWS owned encryption keys by choosing a customer managed key when you create your transformation:

Customer managed keys (Optional)

Customer managed keys — AWS Transform supports the use of a symmetric customer managed key that you create, own, and manage to add a second layer of encryption over the existing encryption using AWS owned keys. 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
- Rotating key cryptographic material
- Adding tags
- Creating key aliases
- · Scheduling keys for deletion
- For more information, see <u>customer managed key</u> in the AWS Key Management Service Developer Guide.

Note

AWS Transform automatically enables encryption at rest using AWS owned keys to protect personally identifiable data at no charge. However, AWS KMS charges apply for using a customer managed key. For more information about pricing, see the <u>AWS Key Management Service pricing</u>.

For more information on AWS KMS, see What is AWS Key Management Service?

How AWS Transform uses grants in AWS Key Management Service

AWS Transform requires a grant to use your customer managed key.

When you create a [Resource Name] encrypted with a customer managed key, AWS Transform creates a grant on your behalf by sending a CreateGrant request to AWS Key Management Service.

Grants in AWS Key Management Service are used to give AWS Transform access to a KMS key in a customer account.

AWS Transform requires the grant to use your customer managed key for the following internal operations:

- Send [KMS API] requests to AWS KMS to verify that the symmetric customer managed KMS key ID entered when creating [Resource Name] is valid.
- Send [KMS API] requests to AWS KMS to generate data keys encrypted by your customer managed key.
- Send [KMS API] 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 Transform 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 attempt to get [Resource Name] from an encrypted [Resource Name] that AWS Transform can't access, then the operation would return an AccessDeniedException error.

Create a customer managed key

You can create a symmetric customer managed key by using the AWS Management Console, or the AWS Key Management Service APIs.

To create a symmetric customer managed key, follow the steps for <u>Creating symmetric customer</u> managed key in the AWS Key Management Service Developer Guide.

Key policy

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 Management Service Developer Guide.

To use your customer managed key with your AWS Transform resources, the following API operations must be permitted in the key policy:

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 Transform requires. For more information about Using Grants, see the AWS Key Management Service Developer Guide.

This allows AWS Transform to do the following:

- Call [KMS API (GenerateDataKeyWithoutPlainText/GenerateDatakey) to generate an encrypted data key and store it, because the data key isn't immediately used to encrypt.
- Call [KMS API (Decrypt)] to use the stored encrypted data key to access encrypted data.
- Set up a retiring principal to allow the service to RetireGrant.
- kms:DescribeKey Provides the customer managed key details to allow [Service Name] to validate the key.

Encryption in transit

Communication between customers and AWS Transform and between AWS Transform and its downstream dependencies is protected using TLS 1.2 or higher connections.

Important

When you Connect a discovery account, AWS Transform creates an Amazon S3 bucket on your behalf in that account. That bucket does not have SecureTransport enabled by default. If you want the bucket policy to include secure transport you must update the policy. For more information, see Security best practices for Amazon S3.

Encryption at rest

AWS Transform stores data at rest using Amazon DynamoDB and Amazon Simple Storage Service (Amazon S3). The data at rest is encrypted using AWS encryption solutions by default. AWS Transform encrypts your data with encryption using AWS owned keys from AWS Key Management Service (AWS KMS). You don't have to take any action to protect the AWS managed keys that encrypt your data. For more information, see AWS owned keys in the AWS Key Management Service Developer Guide.

Data type	AWS-owned key encryption	Customer managed key encryption (Optional)
Customer bucket data	Enabled	Enabled
Customer inputs and outputs such as code and documenta tion stored in an Amazon S3 bucket		
Artifact Store	Enabled	Enabled
Intermediate artifacts as part of code transformation stored in an S3 bucket		
Job Objective	Enabled	Enabled
The customer's intent for the job stored in an Amazon S3 bucket		
Chat messages	Enabled	Enabled
Messages between the customer and AWS Transform stored in an Amazon S3 bucket		
Chat Knowledge Base	Enabled	Enabled
Indexed data relevant to AWS Transform and customer chat stored in Amazon OpenSearc h and processed via AWS Bedrock		

Note: The customer can register their own Customer Managed Key (CMK) to be used for encrypting all of the above data types.

Customer managed keys are KMS keys in your AWS account that you create, own, and manage to directly control access to your data by controlling access to the KMS key. Only symmetric keys are supported. For information on creating your own KMS key, see Creating keys in the AWS Key Management Service Developer Guide.

When you use a customer managed key, AWS Transform makes use of KMS grants, allowing authorized users, roles, or applications to use a KMS key. When an AWS Transform administrator chooses to use a customer managed key for encryption during configuration, a grant is created for them. This grant is what allows the end user to use the encryption key for data encryption at rest. For more information on grants, see Grants in AWS KMS.

Using customer managed KMS keys

After creating a customer managed KMS key, an AWS Transform administrator must provide the key in the AWS Transform console to use it to encrypt data.

To set up a customer managed key to encrypt data in AWS Transform, administrators need permissions to use AWS KMS.

To use features that are encrypted with a customer managed key, users need permissions to allow AWS Transform to access the customer managed key.

If you see an error related to KMS grants while using AWS Transform, you likely need to update your permissions to allow AWS Transform to create grants. To automatically configure the needed permissions, go to the AWS Transform console and choose **Update permissions** in the banner at the top of the page. In order to allow AWS Transform to create grants, you need to update the permissions on the AWS Transform console page.

Allow AWS Transform access to customer managed keys

The following example policy grants users permissions to access features encrypted with a customer managed key by allowing AWS Transform access to the key. This policy is required to use AWS Transform if an administrator has set up a customer managed key for encryption.

AWS Transform service improvement

To help AWS Transform provide the most relevant information, we may use certain content from AWS Transform, such as questions that you ask AWS Transform and its responses, for service improvement. This page explains what content we use and how to opt out.

AWS Transform content used for service improvement

We may use certain content from AWS Transform for service improvement. AWS Transform may use this content, for example, to provide better responses to common questions, fix AWS Transform operational issues, for de-bugging, or for model training.

Content that AWS may use for service improvement includes, for example, your questions to AWS Transform and the responses and code that AWS Transform generates.

How to opt out

The way you opt out of AWS Transform using content for service improvement depends on the environment where you use AWS Transform.

For the AWS Transform web console experience, configure an AI services opt-out policy in AWS Organizations. For more information, see <u>AI services opt-out policies</u> in the *AWS Organizations User Guide*.

To opt out of sharing your content in Visual Studio IDE, use the following procedure.

Go to Tools -> Options -> AWS Toolkit -> Amazon Q

Toggle Share Amazon Q Content with AWS to True or False.

To opt out of sharing your telemetry data in the AWS Toolkit for Visual Studio, use this procedure:

- 1. Under **Tools**, choose **Options**.
- 2. In the Options pane, choose AWS Toolkit, and then choose General.
- 3. Deselect Allow AWS Toolkit to collect usage information.

Cross-region processing in AWS Transform

The following sections describe how cross-region inference and cross-region calls are used to provide the AWS Transform service.

Service improvement 112

Cross-region inference

AWS Transform is powered by Amazon Bedrock, and uses cross-region inference to distribute traffic across different AWS Regions to enhance large language model (LLM) inference performance and reliability. With cross-region inference, you get:

- Increased throughput and resilience during high demand periods
- Improved performance
- Access to newly launched AWS Transform capabilities and features that rely on the most powerful LLMs hosted on Amazon Bedrock

Cross-region inference requests are kept within the AWS Regions that are part of the geography where the data originally resides. For example, a request made from a AWS Transform configuration in the US is kept within the AWS Regions in the US. Although cross-region inferencing doesn't change where your data is stored, your requests and output results may move outside of the Region where the data originally resides. All data will be encrypted while transmitted across Amazon's secure network. There's no additional cost for using cross-region inference.

Cross region inference doesn't affect where your data is stored. For information on where data is stored when you use AWS Transform, see <u>Data protection in AWS Transform</u>.

Supported regions for AWS Transform cross-region inference

The following table describes what Regions your requests may be routed to depending on the geography where the request originated.

Source Region	Destination Regions
US East (N. Virginia) (us-east-1)	US East (N. Virginia) (us-east-1)
	US East (Ohio) (us-east-2)
	US West (Oregon) (us-west-2)
Europe (Frankfurt) Region (eu-central-1)	Europe (Frankfurt) (eu-central-1)
	Europe (Stockholm) (eu-north-1)

Cross-region processing 113

Source Region	Destination Regions
	Europe (Ireland) (eu-west-1)
	Europe (Paris) (eu-west-3)

For a complete list of Regions where you can use AWS Transform, see <u>Supported Regions for AWS</u> Transform.

Identity and access management for AWS Transform

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 Transform resources. IAM is an AWS service that you can use with no additional charge.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- · How AWS Transform works with IAM
- Identity-based policy examples for AWS Transform
- Troubleshooting AWS Transform identity and access
- Using service-linked roles for AWS Transform
- AWS managed policies for AWS Transform
- AWS Transform permissions reference

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in AWS Transform.

Service user – If you use the AWS Transform service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS Transform 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 Transform, see Troubleshooting AWS Transform identity and access.

Service administrator – If you're in charge of AWS Transform resources at your company, you probably have full access to AWS Transform. It's your job to determine which AWS Transform 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 Transform, see How AWS Transform 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 Transform. To view example AWS Transform identity-based policies that you can use in IAM, see Identity-based policy examples for AWS Transform.

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 <u>Tasks that require root user credentials</u> 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 <u>IAM user</u> 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 <u>Rotate access keys regularly for use cases that require long-term credentials</u> in the <u>IAM User Guide</u>.

An <u>IAM group</u> 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

Authenticating with identities 116

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 <u>Use cases for IAM users</u> in the *IAM User Guide*.

IAM roles

An <u>IAM role</u> 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 <u>switch from a user to an IAM role (console)</u>. 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 <u>Methods to assume a role</u> in the <u>IAM User Guide</u>.

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 <u>IAM role</u> 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 <u>Create a role to delegate permissions to an AWS service</u> 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 <u>Use an IAM role to grant permissions to applications running on Amazon EC2 instances</u> 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 <u>specify a principal</u> 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 <u>Access control list (ACL) overview</u> 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 Transform works with IAM

Before you use IAM to manage access to AWS Transform, learn what IAM features are available to use with AWS Transform.

IAM feature	AWS Transform support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys	Yes
ACLs	No
ABAC (tags in policies)	Partial
Temporary credentials	Yes
Principal permissions	Yes
Service roles	Yes
Service-linked roles	Yes

To get a high-level view of how AWS Transform 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 Transform

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 <u>Define custom IAM permissions with customer managed policies</u> 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 Transform

To view examples of AWS Transform identity-based policies, see <u>Identity-based policy examples for</u> AWS Transform.

Resource-based policies within AWS Transform

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must <u>specify a principal</u> 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 Transform

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 Transform actions, see <u>Actions, resources, and condition keys for AWS</u>
Transform in the *Service Authorization Reference*.

Policy actions in AWS Transform use the following prefix before the action:

```
transform
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
    "transform:action1",
    "transform:action2"
]
```

To view examples of AWS Transform identity-based policies, see <u>Identity-based policy examples for AWS Transform</u>.

Policy resources for AWS Transform

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 Transform resource types and their ARNs, and with which actions you can specify the ARN of each resource, see <u>Actions, resources, and condition keys for AWS Transform</u> in the <u>Service Authorization Reference</u>.

To view examples of AWS Transform identity-based policies, see <u>Identity-based policy examples for</u> AWS Transform.

Policy condition keys for AWS Transform

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 <u>condition operators</u>, 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 Transform condition keys, and with which actions and resources you can use a condition key, see <u>Actions, resources, and condition keys for AWS Transform</u> in the <u>Service Authorization Reference</u>.

To view examples of AWS Transform identity-based policies, see <u>Identity-based policy examples for</u> AWS Transform.

ACLs in AWS Transform

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 Transform

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 <u>condition element</u> 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 <u>Define permissions with ABAC authorization</u> in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see <u>Use attribute-based access control</u> (ABAC) in the *IAM User Guide*.

Using temporary credentials with AWS Transform

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 <u>AWS services that</u> 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.

Cross-service principal permissions for AWS Transform

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 Transform

Supports service roles: Yes

A service role is an <u>IAM role</u> 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 <u>Create a role to delegate permissions to an AWS service in the IAM User Guide</u>.

Marning

Changing the permissions for a service role might break AWS Transform functionality. Edit service roles only when AWS Transform provides guidance to do so.

Service-linked roles for AWS Transform

Supports service-linked roles: Yes

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 AWS Transform service-linked roles, see Using servicelinked roles for AWS Transform.

Identity-based policy examples for AWS Transform

By default, users and roles don't have permission to create or modify AWS Transform 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 Transform, including the format of the ARNs for each of the resource types, see Actions, Resources, and Condition Keys for AWS Transform in the Service Authorization Reference.

Topics

- Policy best practices
- Using the AWS Transform console
- Allow users to view their own permissions
- Allow administrators to accept a connector request from the account with AWS Transform

Allow administrators to assign existing IAM Identity Center users and create new IAM Identity
 Center users to assign to AWS Transform

Allow administrators to enable AWS Transform

Policy best practices

Identity-based policies determine whether someone can create, access, or delete AWS Transform 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 <u>AWS managed policies</u> or <u>AWS</u>
 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 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 <u>Validate policies with IAM Access Analyzer</u> 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 <u>Security best practices in IAM</u> in the *IAM User Guide*.

Using the AWS Transform console

To access the AWS Transform console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the AWS Transform resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                 "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
```

Allow administrators to accept a connector request from the account with AWS Transform

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "transform:GetConnector",
                "transform: AssociateConnectorResource",
                "transform: RejectConnector"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketPublicAccessBlock",
                "s3:GetAccountPublicAccessBlock"
            ],
            "Resource": "*"
        },
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:AttachRolePolicy",
```

Allow administrators to assign existing IAM Identity Center users and create new IAM Identity Center users to assign to AWS Transform

JSON

```
"Version": "2012-10-17",
"Statement": [{
"Sid": "AllowKmsAccessViaIdentityCenter",
"Effect": "Allow",
"Action": [
"kms:Decrypt"
],
"Resource": "*",
"Condition": {
"ArnLike": {
"kms:EncryptionContext:aws:sso:instance-arn": "arn:*:sso:::instance/*"
},
"StringLike": {
"kms:ViaService": "sso.*.amazonaws.com"
}
}
},
"Sid": "AllowKmsAccessViaIdentityStore",
```

```
"Effect": "Allow",
"Action": [
"kms:Decrypt"
],
"Resource": "*",
"Condition": {
"ArnLike": {
"kms:EncryptionContext:aws:identitystore:identitystore-arn":
"arn:*:identitystore::*:identitystore/*"
},
"StringLike": {
"kms:ViaService": "identitystore.*.amazonaws.com"
}
}
}
 ]
 }
```

Allow administrators to enable AWS Transform

JSON

```
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sso:ListInstances",
      "sso:CreateInstance",
      "sso:CreateApplication",
      "sso:PutApplicationAuthenticationMethod",
      "sso:PutApplicationGrant",
      "sso:PutApplicationAssignmentConfiguration",
      "sso:ListApplications",
      "sso:GetSharedSsoConfiguration",
      "sso:DescribeInstance",
      "sso:PutApplicationAccessScope",
      "sso:DescribeApplication",
      "sso:DeleteApplication",
      "sso:UpdateApplication",
```

```
"sso:DescribeRegisteredRegions",
    "sso:GetSSOStatus"
  ],
  "Resource": [
    11 * 11
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "sso-directory:GetUserPoolInfo",
    "sso-directory:DescribeUsers",
    "sso-directory:DescribeGroups",
    "sso-directory: SearchGroups",
    "sso-directory:SearchUsers",
    "sso-directory:DescribeDirectory"
  ],
  "Resource": [
    11 * 11
  1
},
  "Effect": "Allow",
  "Action": [
    "organizations:DescribeAccount",
    "organizations:DescribeOrganization",
    "organizations:ListAWSServiceAccessForOrganization",
    "organizations:DisableAWSServiceAccess",
    "organizations: EnableAWSServiceAccess"
  ],
  "Resource": [
  ]
},
  "Effect": "Allow",
  "Action": [
    "kms:ListAliases",
    "kms:CreateGrant",
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:GenerateDataKey*",
    "kms:RetireGrant",
    "kms:DescribeKey"
```

```
],
      "Resource": [
      ]
    },
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/aws-service-role/transform.amazonaws.com/
AWSServiceRoleForAWSTransform"
      1
    },
      "Effect": "Allow",
      "Action": [
        "transform: UpdateProfile",
        "transform:ListProfiles",
        "transform:CreateProfile",
        "transform:DeleteProfile"
      ],
      "Resource": [
      ]
    }
  ]
}
```

Troubleshooting AWS Transform identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS Transform and IAM.

Topics

- I am not authorized to perform an action in AWS Transform
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my AWS Transform resources

Troubleshooting 134

I am not authorized to perform an action in AWS Transform

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the mateojackson IAM user tries to use the console to view details about a fictional my-example-widget resource but doesn't have the fictional AWS Transform: GetWidget permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: AWS Transform:GetWidget on resource: my-example-widget
```

In this case, the policy for the mateojackson user must be updated to allow access to the *my-example-widget* resource by using the AWS Transform: *GetWidget* action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the iam: PassRole action, your policies must be updated to allow you to pass a role to AWS Transform.

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 Transform. 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.

Troubleshooting 135

I want to allow people outside of my AWS account to access my AWS Transform 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 Transform supports these features, see <u>How AWS Transform works with</u> 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 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.

Using service-linked roles for AWS Transform

AWS Transform uses AWS Identity and Access Management (IAM) <u>service-linked roles</u>. A service-linked role is a unique type of IAM role that is linked directly to AWS Transform. Service-linked roles are predefined by AWS Transform and include all the permissions that the service requires to call other AWS services on your behalf.

Using service-linked roles for AWS Transform

AWS Transform uses AWS Identity and Access Management (IAM) <u>service-linked roles</u>. A service-linked role is a unique type of IAM role that is linked directly to AWS Transform. Service-linked roles are predefined by AWS Transform and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up AWS Transform easier because you don't have to manually add the necessary permissions. AWS Transform defines the permissions of its service-linked roles,

Using service-linked roles 136

and unless defined otherwise, only AWS Transform can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This protects your AWS Transform resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see <u>AWS services that work</u> with <u>IAM</u> and look for the services that have **Yes** in the **Service-linked roles** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for AWS Transform

AWS Transform uses the service-linked role named <u>AWSServiceRoleForAWSTransform</u> – This Service-Linked Role provides AWS Transform with the ability to provide usage information.

The AWSServiceRoleForAWSTransform service-linked role trusts the following services to assume the role:

• transform.amazonaws.com

The role permissions policy named AWSServiceRoleForAWSTransform allows AWS Transform to complete the following actions on the specified resources:

- cloudwatch:PutMetricData
 - Send custom metrics to CloudWatch for AWS Transform operations
 - Track transformation progress, success rates, and performance metrics
 - Enable monitoring and alerting on transformation workflows
- sso:DescribeApplication
 - View details about a specific application in Identity Center
 - Get application metadata like name, description, status, and configuration
- sso:GetApplicationAssignmentConfiguration
 - Retrieve assignment configuration settings for an application
 - See how users/groups are configured to be assigned to the application
- sso:ListApplicationAssignmentsForPrincipal
 - List all applications assigned to a specific user or group (principal)

Using service-linked roles 137

- View which applications a particular identity has access to
- Enables decryption of KMS-encrypted data when accessed through IAM Identity Center. Only works when the encryption context contains a valid IAM Identity Center instance ARN and must be accessed via IAM Identity Center service endpoints.

Allows decryption of KMS-encrypted data when accessed through Identity Store. Only works
when the encryption context contains a valid Identity Store ARN and must be accessed via
Identity Store service endpoints.

You must configure permissions to allow your users, groups, or roles to create, edit, or delete a service-linked role. For more information, see <u>Service-linked role permissions</u> in the *IAM User Guide*.

Creating a service-linked role for AWS Transform

You don't need to manually create a service-linked role. When you create a profile for AWS Transform in the AWS Management Console, AWS Transform creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you update the settings, AWS Transform creates the service-linked role for you again.

You can also use the IAM console or AWS CLI to create a service-linked role with the transform. amazonaws. com service name. For more information, see Creating a service-linked role in the IAM User Guide. If you delete this service-linked role, you can use this same process to create the role again.

Editing a service-linked role for AWS Transform

AWS Transform does not allow you to edit the AWSServiceRoleForAWSTransform service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the IAM User Guide.

Deleting a service-linked role for AWS Transform

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

Using service-linked roles 138



Note

If the AWS Transform service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForAWSTransform service-linked role. For more information, see Deleting a service-linked role in the IAM User Guide.

Supported Regions for AWS Transform service-linked roles

AWS Transform does not support using service-linked roles in every Region where the service is available. You can use the AWSServiceRoleForAWSTransform role in the following Regions. For more information, see AWS Regions and endpoints.

Region name	Region identity	Support in AWS Transform
US East (N. Virginia)	us-east-1	Yes
Europe (Frankfurt)	eu-central-1	Yes

AWS managed policies for AWS Transform

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

AWS managed policies 139

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 Transform updates for AWS managed policies

View details about updates to AWS managed policies for AWS Transform since March 1, 2021.

Change	Description	Date
AWSServiceRoleForA WSTransform – Updated policy	Added support for customer managed keys in IAM Identity Center.	September 17, 2025
AWSTransformApplic ationDeploymentPolicy – New policy	Added a new AWS managed policy that enables AWS Transform to deploy transformed .NET applicati ons by creating and managing Amazon EC2 instances, AWS CloudFormation stacks, and associated resources.	August 28, 2025
AWSServiceRoleForA WSTransform – Updated policy	Added a new policy.	May 15, 2025

AWS managed policy: AWSServiceRoleForAWSTransform

This policy is attached to the AWSServiceRoleForAWSTransform service-linked role (SLR).

Permissions details

To view the policy permission details see <u>AWSServiceRoleForAWSTransform</u> in the AWS Managed Policy Reference Guide.

AWS managed policies 140

AWS managed policy: AWSTransformApplicationDeploymentPolicy

This policy enables AWS Transform to deploy transformed .NET applications by creating and managing Amazon EC2 instances, AWS CloudFormation stacks, and associated resources.

Description

This policy includes the following permissions:

- **CloudFormation** Allows creating, updating, deleting, and describing CloudFormation stacks with names that start with AWSTransform-Deploy-Infra-stack. Stack operations are restricted to resources tagged with CreatedBy: AWSTransform and limited to the same AWS account.
- Amazon EC2 Allows describing VPCs, subnets, security groups, images, and instances.
 Permits running, starting, stopping, terminating, and modifying EC2 instances, but only when called through CloudFormation. Tag creation is restricted to specific allowed tag keys (Name, CreatedBy, CreatedFor, Environment) and only during instance launch.
- AWS Identity and Access Management (IAM) Allows getting and passing the specific IAM role AWSTransform-Deploy-EC2-Instance-Role and accessing the instance profile AWSTransform-Deploy-EC2-Instance-Profile. Access is restricted to resources tagged with CreatedFor: AWSTransform.
- Amazon EC2 Systems Manager (SSM) Allows retrieving Amazon Linux AMI parameters from the AWS-managed parameter store.

The policy implements least-privilege access through resource-level permissions, tag-based conditions, service control restrictions using aws:CalledVia, account-level restrictions, and explicit deny statements to prevent unauthorized tag modifications outside of CloudFormation operations.

Permissions details

To view the policy permission details see <u>AWSTransformApplicationDeploymentPolicy</u> in the AWS Managed Policy Reference Guide.

AWS Transform permissions reference

This section provides information about the APIs used by AWS Transform, and what they do.

Topics

AWS Transform APIs

AWS Transform APIs

- · transform:BatchGetMessage
- transform:BatchGetUserDetails
- transform:CompleteArtifactUpload
- transform:CreateArtifactDownloadUrl
- transform:CreateArtifactUploadUrl
- transform:CreateAssetDownloadUrl
- transform:CreateConnector
- transform:CreateFeedback
- transform:CreateJob
- transform:CreateSession
- transform:CreateWorkspace
- transform:DeleteConnector
- transform:DeleteJob
- transform:DeleteSelfRoleMappings
- transform:DeleteUserRoleMappings
- transform:DeleteWorkspace
- transform:DetectIsAllowedForOperation
- transform:GetConnector
- transform:GetFeedbackQuestions
- transform:GetHitlTask
- transform:GetJob
- transform:GetLoginRedirectUri
- transform:GetUserDetails
- transform:GetUserPreferences
- transform:GetWorkspace
- transform:ListArtifacts
- transform:ListConnectors
- transform:ListHitlTasks

- transform:ListJobPlanSteps
- transform:ListJobs
- transform:ListMessages
- transform:ListPlanUpdates
- transform:ListUserRoleMappings
- transform:ListWorklogs
- · transform:ListWorkspaces
- transform:PutUserPreferences
- transform:PutUserRoleMappings
- transform:RevokeSession
- transform:SearchUsers
- · transform:SearchUsersTypeahead
- transform:SendMessage
- transform:StartJob
- transform:StopJob
- transform:SubmitCriticalHitlTask
- transform:SubmitStandardHitlTask
- transform:TestReleaseTraitPreProd
- transform:TestReleaseTraitProd
- transform:UpdateHitlTask
- transform:UpdateJob
- transform:UpdateWorkspace
- transform:VerifySession

Compliance validation for AWS Transform

To learn whether an AWS service is within the scope of specific compliance programs, see <u>AWS</u> <u>services in Scope by Compliance Program</u> and choose the compliance program that you are interested in. For general information, see <u>AWS Compliance Programs</u>.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Compliance validation 143

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:

- <u>Security Compliance & Governance</u> 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.
- <u>AWS Compliance Resources</u> This collection of workbooks and guides might apply to your industry and location.
- <u>AWS Customer Compliance Guides</u> 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)).
- <u>Evaluating Resources with Rules</u> in the *AWS Config Developer Guide* The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- <u>AWS Security Hub</u> 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 <u>Security Hub controls</u> reference.
- <u>Amazon GuardDuty</u> 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.
- <u>AWS Audit Manager</u> This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

Resilience in AWS Transform

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

Resilience 144

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 Transform offers several features to help support your data resiliency and backup needs.

AWS Transform and interface endpoints (AWS PrivateLink)

You can establish a private connection between your VPC and AWS Transform by creating an *interface VPC endpoint*. Interface endpoints are powered by <u>AWS PrivateLink</u>, a technology that enables you to privately access the AWS Transform console without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Traffic between your VPC and AWS Transform does not leave the Amazon network.

Each interface endpoint is represented by one or more Elastic Network Interfaces in your subnets.

For more information, see <u>Interface VPC endpoints (AWS PrivateLink)</u> in the *Amazon VPC User Guide*.



- Only the AWS Transform console can be accessed thru AWS PrivateLink.
- The AWS Transform WebApp and other related services such as .NET IDE cannot be accessed thru AWS PrivateLink.

Considerations for AWS Transform VPC endpoints

Before you set up an interface VPC endpoint for AWS Transform, ensure that you review <u>Interface</u> endpoint properties and limitations in the *Amazon VPC User Guide*.

Prerequisites

Before you begin any of the procedures below, ensure that you have the following:

• An AWS account with appropriate permissions to create and configure resources.

- A VPC already created in your AWS account.
- Familiarity with AWS services, especially Amazon VPC and AWS Transform.

Creating an interface VPC endpoint for AWS Transform

You can create a VPC endpoint for the AWS Transform service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see <u>Creating an interface endpoint</u> in the *Amazon VPC User Guide*.

Create the following VPC endpoints for AWS Transform using this service name:

• com.amazonaws.region.transform

Replace *region* with AWS Region where your AWS Transform profile is installed, for example, *com.amazonaws.us-east-1.transform*.

For more information, see <u>Supported Regions for AWS Transform</u> and <u>Accessing a service through</u> an interface endpoint in the *Amazon VPC User Guide*.

Using an on-premises computer to connect to a AWS Transform endpoint

This section describes the process of using an on-premises computer to connect to AWS Transform through a AWS PrivateLink endpoint in your AWS VPC.

- Create a VPN connection between your on-premises device and your VPC.
- 2. <u>Create an interface VPC endpoint for AWS Transform.</u>
- 3. <u>Set up an inbound Amazon Route 53 endpoint.</u> This will enable you to use the DNS name of your AWS Transform endpoint from your on-premises device.

Monitoring AWS Transform

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Transform and your other AWS solutions. AWS provides the following monitoring tools to watch AWS Transform, report when something is wrong, and take automatic actions when appropriate:

- Amazon CloudWatch monitors your AWS resources and and the applications you run on AWS
 in real time. You can collect and track metrics, create customized dashboards, and set alarms
 that notify you or take actions when a specified metric reaches a threshold that you specify.
 For example, you can have CloudWatch track CPU usage or other metrics of your Amazon EC2
 instances and automatically launch new instances when needed. For more information, see the
 Amazon CloudWatch User Guide.
- Amazon CloudWatch Logs enables you to monitor, store, and access your log files from Amazon EC2 instances, CloudTrail, and other sources. CloudWatch Logs can monitor information in the log files and notify 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.
- Amazon EventBridge can be used to automate your AWS services and respond automatically
 to system events, such as application availability issues or resource changes. Events from AWS
 services are delivered to EventBridge in near real time. You can write simple rules to indicate
 which events are of interest to you and which automated actions to take when an event matches
 a rule. For more information, see Amazon EventBridge User Guide.
- 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 address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.

Monitoring AWS Transform with Amazon CloudWatch

You can monitor AWS Transform using CloudWatch, which collects raw data and processes it into readable, near real-time metrics. These statistics are kept for 15 months, so that you can access historical information and gain a better perspective on how your web application or service is performing. You can also set alarms that watch for certain thresholds, and send notifications or take actions when those thresholds are met. For more information, see the <u>Amazon CloudWatch</u> User Guide.

Monitoring with CloudWatch 147

Monitoring AWS Transform events in Amazon EventBridge

You can monitor AWS Transform events in EventBridge, which delivers a stream of real-time data from your own applications, software-as-a-service (SaaS) applications, and AWS services. EventBridge routes that data to targets such as AWS Lambda and Amazon Simple Notification Service. These events are the same as those that appear in Amazon CloudWatch Events, which delivers a near real-time stream of system events that describe changes in AWS resources.

Logging AWS Transform API calls using AWS CloudTrail

AWS Transform is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Transform. CloudTrail captures all API calls for AWS Transform as events. The calls captured include calls from the AWS Transform console and code calls to the AWS Transform API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Transform. 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 Transform, 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 <u>AWS CloudTrail User Guide</u>.

AWS Transform information in CloudTrail

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

Monitoring events 148

- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

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 Transform 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.

Notifications

AWS Transform provides email notifications to keep you informed about workspace access changes, job status updates, and pending collaborator requests.

Email notifications

Email notifications are enabled by default, and are sent to users who have read-only or higher permissions on a job. Users can modify their notification preferences. There are three categories of email notifications:

Workspace access updates

You receive a notification when you are added to a new workspace, and when your role is changed within a workspace.

Daily digest emails

You receive a daily digest email at 9:00 AM Pacific time summarizing your ongoing jobs if you have active jobs with one or more open required collaborator requests. The email includes a table showing workspace name, job name, job status, collaborator request details, and required actions (limited to 10 rows)



Note

Daily digest emails are only sent for jobs that have required collaborator requests. Optional collaborator requests do not trigger digest emails.

Job status updates

You receive an email when a job you have access to completes or fails.

Managing email notifications

You can modify your email notification preferences in the web application by clicking the Settings cog in the top right corner of the application, and choosing **Notification settings**.

The notification settings page provides the following controls:

Notification Type	Description	Default Setting
Workspace Access Updates	Receive emails when added to a workspace, or when your role is changed within a workspace	Enabled
Daily Collaborator Request Digest	Receive a daily digest of open collaborator requests for jobs in accessible workspaces	Enabled
Job Completion Updates	Receive emails when a job is completed or fails	Enabled

You can use a general setting to enable or disable all notifications, or configure individual notification types separately.

Email notifications 150

Quotas for AWS Transform

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, and other quotas cannot be increased.

To view the quotas for AWS Transform, open the <u>Service Quotas console</u>. In the navigation pane, choose **AWS services** and select **AWS Transform**.

To request a quota increase, see <u>Requesting a Quota Increase</u> in the <u>Service Quotas User Guide</u>. If the quota is not yet available in Service Quotas use the <u>limit increase</u> form.

Your AWS account has the following quotas related to AWS Transform.

Workload	Dimension	Quota	Adjustable	Description
Platform	Workspaces per AWS Region per AWS Transform- enabled account	100 workspace s for each supported AWS Region	Yes	The maximum number of workspaces per AWS Transform- enabled account per AWS Region
VMWare Migration	Server count for each wave	Each wave: 150 servers	No	The maximum number of servers in a wave
Mainframe	Concurrent jobs for each IAM Identity Center account	10 jobs for each supported Region	Yes	The maximum number of concurrent Mainframe jobs in an IAM Identity Center account
	Lines of code for each job	3,000,000 lines of code for	No	The maximum number of lines

Workload	Dimension	Quota	Adjustable	Description
		each supported region		of code for each job.
	Analysis monthly lines of code	100,000,000 lines of code for each supported region	Yes	The maximum number of lines of code for Analyze Code job objective in a calendar month
	Business document generation monthly lines of code	50,000,000 lines of code for each supported region	Yes	The maximum number of lines of code for Business Documentation Generation job objective in a calendar month
	Technical document generation monthly lines of code	50,000,000 lines of code for each supported region	Yes	The maximum number of lines of code for Technical Documentation Generation job objective in a calendar month
	Transform of lines of code	10,000,000 lines of code for each supported region	Yes	The maximum number of lines of code transformed in a calendar month

Workload	Dimension	Quota	Adjustable	Description
	Decomposition monthly lines of code	50,000,000 lines of code for each supported region	Yes	The maximum number of lines of code for Decomposition job objective in a calendar month
	Refactor monthly lines of code	50,000,000 lines of code for each supported region	Yes	The maximum number of lines of code for Refactor job objective in a calendar month
	Reforge monthly lines of code	50,000,000 lines of code for each supported region	Yes	The maximum number of lines of code for Reforge job objective in a calendar month
	Extracted business logic monthly lines of code	50,000,000 lines of code for each supported region	Yes	The maximum number of lines of code for Extracted Business Logic job objective in a calendar month

Workload	Dimension	Quota	Adjustable	Description
.NET	.NET monthly lines of code (web & IDE)	1,000,000 lines of code for each supported Region	Yes	The maximum lines of .NET application code that can be transform ed in a calendar month in each supported AWS Region of an IAM Identity Center account
	Concurrent .NET jobs with the Microsoft Visual Studio Extension for each IAM Identity Center account	10 jobs for each supported Region	Yes	The maximum number of concurrent .NET transformation jobs in each supported AWS Region of an IAM Identity Center account when using the Microsoft Visual Studio Extension .

Workload	Dimension	Quota	Adjustable	Description
	Concurren t .NET web jobs for each IAM Identity Center account	5 jobs for each supported Region	Yes	The maximum number of concurrent .NET transformation jobs in each supported AWS Region of an IAM Identity Center account when using the AWS Transform web app.
Assessment	Server count limit for each assessment	Each assessmen t: 30,000 servers	No	The maximum number of servers in a single assessment job

Supported Regions for AWS Transform



Note

If you make a request that requires AWS Transform to retrieve information from an opt-in Region not listed on this page, AWS Transform can make calls to that Region. To manage access to Regions AWS Transform can make calls to, see Security in AWS Transform.

This topic describes the AWS Regions where you can use AWS Transform. For more information about AWS Regions, see Specify which AWS Regions your account can use in the AWS Account Management Reference Guide.

Your data might be processed in a different Region from the Region where you use AWS Transform. For information on cross-region processing in AWS Transform, see Cross-region processing. For information on where data is stored during processing, see Data protection.

Supported AWS Regions (enabled by default)

You can create AWS Transform workspaces in the following AWS Regions. These Regions are enabled by default, meaning you don't need to enable them before use. For more information, see Regions that are enabled by default.

- US East (N. Virginia)
- Europe (Frankfurt)

The workspace in which you create a job determines the AWS Region of the job. To create a job in a different Region, you must use a different workspace that is in your desired Region.

For VMware projects, the Region of the workspace is used for discovery. However, you can specify a different Region as the migration target. That target Region is where your workloads are hosted when the migration is complete. For more information about AWS Region considerations for VMware migrations, including the list of possible target Regions, see the section called "AWS" account connectors".

Document history for the AWS Transform User Guide

The following table describes the document history for the AWS Transform User Guide.

Change	Description	Date
Updated AWS managed policy	Updated the <u>AWSServic</u> <u>eRoleForAWSTransform</u> documentation.	September 17, 2025
Storage assessment support	Added NetApp Data Infrastructure Insights (DII) file support for storage assessments.	September 2, 2025
Target regions update	Added three <u>supported target</u> <u>regions</u> .	August 31, 2025
Managed policy update	Added IAM Identity Center read-only permisions to be used by <u>AWSServiceRoleForA</u> <u>WSTransform</u> .	August 29, 2025
Documented new AWS managed policy	Added <u>AWSTransformApplic</u> <u>ationDeploymentPolicy</u> section.	August 28, 2025
interface endpoints	Added <u>interface endpoints</u> topic.	July 9, 2025
<u>Initial release</u>	Initial release of the AWS Transform User Guide.	May 15, 2025