

---

# AWS Database Migration Service

## User Guide



## AWS Database Migration Service: User Guide

Copyright © 2023 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 Database Migration Service? .....	1
Migration tasks that AWS DMS performs .....	1
How AWS DMS works .....	3
High-level view of AWS DMS .....	3
Components .....	4
Sources .....	8
Targets .....	10
Amazon Resource Names .....	11
With other AWS services .....	12
Support for AWS CloudFormation .....	13
Getting started .....	14
Setting up .....	14
Sign up for AWS .....	14
Create an IAM user .....	15
Prerequisites .....	16
Create a VPC .....	17
Create Amazon RDS parameter groups .....	18
Create your source Amazon RDS database .....	18
Create your target Amazon RDS database .....	19
Create an Amazon EC2 client .....	20
Populate your source database .....	21
Migrate schema .....	22
Replication .....	24
Step 1: Create a replication instance .....	24
Step 2: Specify source and target endpoints .....	26
Step 3: Create a task and migrate data .....	28
Step 4: Test replication .....	30
Step 5: Clean up AWS DMS resources .....	31
Additional resources .....	32
Using AWS DMS Fleet Advisor .....	33
Supported AWS Regions .....	33
Creating required resources .....	34
Amazon S3 bucket .....	34
IAM Resources .....	35
Creating database users .....	37
Using a database user with AWS DMS Fleet Advisor .....	37
Creating a database user with MySQL .....	37
Creating a database user with Oracle .....	38
Creating a database user with PostgreSQL .....	38
Creating a database user with Microsoft SQL Server .....	38
Creating a data collector .....	39
Installing a data collector .....	41
Configuring credentials for data forwarding .....	42
Discovering OS and database servers .....	43
Managing monitored objects .....	45
Using SSL .....	46
Supported Databases .....	46
Setting up SSL .....	46
Using SSL .....	46
Checking the Server Certificate Authority (CA) Certificate for SQL Server .....	46
Collecting data .....	47
Using inventories .....	48
Using the database inventory .....	48
Using the schema inventory .....	49

Troubleshooting .....	50
Network and server connection related issues .....	51
WMI related issues .....	51
WPC related issues .....	52
SSL related issues .....	52
Converting database schemas .....	53
Supported AWS Regions .....	54
Migration sources and targets .....	54
Features .....	54
Limitations .....	55
Getting started .....	56
Prerequisites .....	56
Step 1: Create an instance profile .....	59
Step 2: Configure data providers .....	59
Step 3: Create a migration project .....	60
Step 4: Create an assessment report .....	60
Step 5: Convert your source code .....	61
Step 6: Apply the converted code .....	61
Step 7: Clean up and troubleshoot .....	61
Instance profiles .....	62
Network configuration .....	63
Data providers .....	64
Creating data providers .....	65
Creating source data providers .....	66
Creating target data providers .....	68
Migration projects .....	69
Creating migration projects .....	70
Managing migration projects .....	71
Specifying migration project settings .....	71
Database migration assessment reports .....	72
Creating an assessment report .....	73
Viewing your assessment report .....	73
Saving assessment reports .....	74
Schema conversion .....	75
Setting up transformation rules .....	76
Converting your database schema .....	77
Specifying schema conversion settings .....	79
Refreshing your database schemas .....	82
Saving and applying your schema .....	82
Using extension packs .....	83
Best practices .....	85
Migration planning for AWS Database Migration Service .....	85
Schema conversion .....	86
Reviewing AWS DMS documentation .....	86
Running a proof of concept .....	87
Improving performance .....	87
Using your own on-premises name server .....	90
Using Amazon Route 53 Resolver with AWS DMS .....	91
Migrating large binary objects (LOBs) .....	92
Using limited LOB mode .....	92
Improved LOB performance .....	93
Improving performance when migrating large tables using row filtering .....	95
Ongoing replication .....	95
Reducing load on your source database .....	96
Reducing bottlenecks on your target database .....	96
Using data validation .....	96
Metrics monitoring .....	96

Events .....	97
Using the task log .....	97
Troubleshooting replication with Time Travel .....	97
Changing the user and schema for an Oracle target .....	98
Changing table and index tablespaces for an Oracle target .....	98
Upgrading a replication instance .....	99
Understanding migration cost .....	99
Working with replication instances .....	100
Choosing replication instance types .....	103
Deciding what instance class to use .....	106
Unlimited mode burstable instances .....	106
Sizing a replication instance .....	107
Factors to consider .....	107
Common issues .....	108
Best practices .....	108
Replication engine versions .....	109
Upgrading the engine version using the console .....	109
Upgrading the engine version using the AWS CLI .....	109
Public and private replication instances .....	110
IP addressing and network types .....	111
Setting up a network for a replication instance .....	112
Network configurations for database migration .....	112
Creating a replication subnet group .....	117
Resolving domain endpoints using DNS .....	118
Setting an encryption key .....	119
Creating a replication instance .....	119
Modifying a replication instance .....	123
Rebooting a replication instance .....	125
Deleting a replication instance .....	127
DMS maintenance window .....	128
Effect of maintenance on existing migration tasks .....	129
Changing the maintenance window setting .....	129
Endpoints .....	131
Creating source and target endpoints .....	131
Sources for data migration .....	134
Using Oracle as a source .....	134
Using SQL Server as a source .....	175
Using Azure SQL database as a source .....	195
Using Azure SQL Managed Instance as a source .....	195
Using Google Cloud for MySQL as a source .....	195
Using PostgreSQL as a source .....	195
Using MySQL as a source .....	216
Using SAP ASE as a source .....	225
Using MongoDB as a source .....	231
Using Amazon DocumentDB as a source .....	242
Using Amazon S3 as a source .....	254
Using IBM Db2 LUW as a source .....	261
Using IBM Db2 for z/OS as a source .....	265
Targets for data migration .....	293
Using Oracle as a target .....	293
Using SQL Server as a target .....	300
Using PostgreSQL as a target .....	304
Using MySQL as a target .....	311
Using Amazon Redshift as a target .....	315
Using SAP ASE as a target .....	330
Using Amazon S3 as a target .....	332
Using Amazon DynamoDB as a target .....	361

Using Amazon Kinesis Data Streams as a target .....	376
Using Apache Kafka as a target .....	387
Using OpenSearch as a target .....	404
Using Amazon DocumentDB as a target .....	408
Using Amazon Neptune as a target .....	415
Using Redis as a target .....	426
Using Babelfish as a target .....	431
VPC endpoints for data migration .....	437
Who is impacted when migrating to AWS DMS versions 3.4.7 and higher? .....	437
Who is not impacted when migrating to AWS DMS versions 3.4.7 and higher? .....	437
Preparing a migration to AWS DMS versions 3.4.7 and higher .....	437
Supported DDL statements .....	438
Tasks .....	440
Creating a task .....	442
Task settings .....	448
Setting LOB support .....	479
Creating multiple tasks .....	480
Continuous replication tasks .....	481
Replication starting from a CDC start point .....	482
Performing bidirectional replication .....	485
Modifying a task .....	487
Moving a task .....	487
Reloading tables during a task .....	488
AWS Management Console .....	488
Table mapping .....	491
Specifying table selection and transformations rules from the console .....	491
Specifying table selection and transformations rules using JSON .....	494
Selection rules and actions .....	495
Wildcards in table mapping .....	499
Transformation rules and actions .....	500
Using transformation rule expressions to define column content .....	513
Table and collection settings rules and operations .....	523
Using source filters .....	543
Applying filters .....	544
Filtering by time and date .....	547
Enabling and working with premigration assessments .....	548
IAM service role for storing assessment runs in an S3 Bucket .....	549
Specifying, starting, and viewing assessment runs .....	550
Starting and viewing data type assessments .....	555
Specifying supplemental data .....	557
Monitoring tasks .....	558
Task status .....	559
Table state during tasks .....	560
Monitoring replication tasks using Amazon CloudWatch .....	561
AWS Database Migration Service metrics .....	562
Replication instance metrics .....	564
Replication task metrics .....	565
Viewing and managing AWS DMS logs .....	567
Logging AWS DMS API calls with AWS CloudTrail .....	568
AWS DMS information in CloudTrail .....	568
Understanding AWS DMS log file entries .....	569
Working with EventBridge events .....	572
Using Amazon EventBridge event rules for AWS DMS .....	572
AWS DMS event categories and event messages .....	573
Working with Amazon SNS events .....	576
Moving event subscriptions to Amazon EventBridge .....	576
Working with Amazon SNS events and notifications .....	577

AWS DMS event categories and event messages for SNS notifications .....	578
Subscribing to AWS DMS event notification using SNS .....	580
Using the AWS Management Console .....	580
Validating the access policy of your SNS topic .....	583
Data validation .....	584
Replication task statistics .....	584
Replication task statistics with Amazon CloudWatch .....	586
Revalidating tables during a task .....	587
AWS Management Console .....	587
Using JSON editor to modify validation rules .....	587
Validation only tasks .....	588
Full load validation only .....	588
CDC validation only .....	588
Validation only use cases .....	588
Troubleshooting .....	589
Limitations .....	590
Tagging resources .....	591
API .....	592
Security .....	594
Data protection .....	596
Data encryption .....	596
Internetwork traffic privacy .....	597
Identity and access management .....	598
Audience .....	598
Authenticating with identities .....	598
Managing access using policies .....	600
How AWS Database Migration Service works with IAM .....	602
Identity-based policy examples .....	606
Resource-based policy examples .....	612
Using secrets to access resources .....	615
Troubleshooting .....	621
Compliance validation .....	624
Resilience .....	625
Infrastructure security .....	626
IAM permissions required .....	628
IAM roles for the CLI and API .....	632
Fine-grained access control .....	636
Using resource names to control access .....	636
Using tags to control access .....	638
Setting an encryption key .....	643
Network security .....	645
Using SSL .....	647
Limitations on using SSL with AWS DMS .....	648
Managing certificates .....	648
Enabling SSL for a MySQL-compatible, PostgreSQL, or SQL Server endpoint .....	649
Changing the database password .....	651
Limits .....	652
Resource quotas for AWS Database Migration Service .....	652
Understanding API request throttling .....	653
Troubleshooting and diagnostic support .....	654
Migration tasks run slowly .....	655
Task status bar doesn't move .....	655
Task completes but nothing was migrated .....	655
Foreign keys and secondary indexes are missing .....	655
AWS DMS does not create CloudWatch logs .....	656
Issues occur with connecting to Amazon RDS .....	656
Error message: Incorrect thread connection string: Incorrect thread value 0 .....	656

Networking issues occur .....	656
CDC is stuck after full load .....	657
Primary key violation errors occur when you restart a task .....	657
Initial load of a schema fails .....	657
Tasks fail with an unknown error .....	658
Task restart loads tables from the beginning .....	658
Number of tables per task causes issues .....	658
Tasks fail when a primary key is created on a LOB column .....	658
Duplicate records occur on a target table without a primary key .....	658
Source endpoints fall in the reserved IP range .....	658
Timestamps are garbled in Amazon Athena queries .....	659
Troubleshooting issues with Oracle .....	659
Pulling data from views .....	659
Migrating LOBs from Oracle 12c .....	660
Switching between Oracle LogMiner and Binary Reader .....	660
Error: Oracle CDC stopped 122301 oracle CDC maximum retry counter exceeded. ....	660
Automatically add supplemental logging to an Oracle source endpoint .....	661
LOB changes aren't being captured .....	661
Error: ORA-12899: Value too large for column <i>column-name</i> .....	661
NUMBER data type being misinterpreted .....	661
Records missing during full load .....	662
Table Error .....	662
Error: Cannot retrieve Oracle archived Redo log destination ids .....	662
Evaluating read performance of Oracle redo or archive logs .....	662
Troubleshooting issues with MySQL .....	664
CDC task failing for Amazon RDS DB instance endpoint because binary logging disabled .....	664
Connections to a target MySQL instance are disconnected during a task .....	664
Adding autocommit to a MySQL-compatible endpoint .....	665
Disable foreign keys on a target MySQL-compatible endpoint .....	665
Characters replaced with question mark .....	666
"Bad event" log entries .....	666
Change data capture with MySQL 5.5 .....	666
Increasing binary log retention for Amazon RDS DB instances .....	666
Log message: Some changes from the source database had no impact when applied to the target database .....	666
Error: Identifier too long .....	666
Error: Unsupported character set causes field data conversion to fail .....	667
Error: Codepage 1252 to UTF8 [120112] a field data conversion failed .....	667
Troubleshooting issues with PostgreSQL .....	668
JSON data types being truncated .....	668
Columns of a user-defined data type not being migrated correctly .....	669
Error: No schema has been selected to create in .....	669
Deletes and updates to a table aren't being replicated using CDC .....	669
Truncate statements aren't being propagated .....	669
Preventing PostgreSQL from capturing DDL .....	669
Selecting the schema where database objects for capturing DDL are created .....	669
Oracle tables missing after migrating to PostgreSQL .....	670
ReplicationSlotDiskUsage increases and restart_lsn stops moving forward during long transactions, such as ETL workloads .....	670
Task using view as a source has no rows copied .....	670
Troubleshooting issues with Microsoft SQL Server .....	670
Errors capturing changes for SQL server database .....	671
Missing identity columns .....	671
Error: SQL Server doesn't support publications .....	671
Changes don't appear in your target .....	671
Non-uniform table mapped across partitions .....	671
Troubleshooting issues with Amazon Redshift .....	672

Loading in to an Amazon Redshift cluster in a different AWS Region .....	672
Error: Relation "awsdms_apply_exceptions" already exists .....	672
Errors with tables whose name begins with "awsdms_changes" .....	673
Seeing tables in clusters with names like dms.awsdms_changes000000000XXXX .....	673
Permissions required to work with Amazon Redshift .....	673
Troubleshooting issues with Amazon Aurora MySQL .....	673
Error: CHARACTER SET UTF8 fields terminated by ',' enclosed by "" lines terminated by '\n' .....	673
Troubleshooting issues with SAP ASE .....	674
Error: LOB columns have NULL values when source has a composite unique index with NULL values .....	674
Troubleshooting issues with IBM Db2 .....	674
Error: Resume from timestamp is not supported Task .....	674
Working with diagnostic support scripts .....	674
Oracle support scripts .....	675
SQL Server support scripts .....	678
MySQL-compatible support scripts .....	680
PostgreSQL support scripts .....	681
Working with the diagnostic support AMI .....	683
Launch a new AWS DMS diagnostic Amazon EC2 instance .....	683
Create an IAM role .....	683
Run Diagnostic Tests .....	684
Next Steps .....	687
AMI IDs by region .....	687
Migrating large data stores with AWS Snowball Edge .....	689
Process overview .....	690
Prerequisites .....	691
Migration checklist .....	691
Step-by-step procedures .....	693
Step 1: Create a AWS Snowball Edge job .....	693
Step 2: Download and install the AWS Schema Conversion Tool (AWS SCT) .....	693
Step 3: Unlock the AWS Snowball Edge device .....	694
Step 4: Configure the AWS DMS agent host with ODBC drivers .....	696
Step 5: Install the AWS DMS Agent .....	699
Step 6: Create a new AWS SCT project .....	701
Step 7: Configure AWS SCT to use AWS Snowball Edge .....	701
Step 8: Register the AWS DMS Agent in AWS SCT .....	705
Step 9: Create a local and AWS DMS task .....	705
Step 10: Run and monitor the task in SCT .....	706
Limitations .....	708
Reference .....	710
AWS DMS data types .....	710
Release notes .....	712
AWS DMS 3.4.7 release notes .....	712
AWS DMS 3.4.6 release notes .....	716
AWS DMS 3.4.5 release notes .....	719
AWS DMS 3.4.4 release notes .....	721
AWS DMS 3.4.3 release notes .....	723
AWS DMS 3.4.2 release notes .....	724
AWS DMS 3.4.1 release notes .....	725
AWS DMS 3.4.0 release notes .....	726
AWS DMS 3.3.4 release notes .....	727
AWS DMS 3.3.3 release notes .....	727
Document history .....	729
AWS glossary .....	731

# What is AWS Database Migration Service?

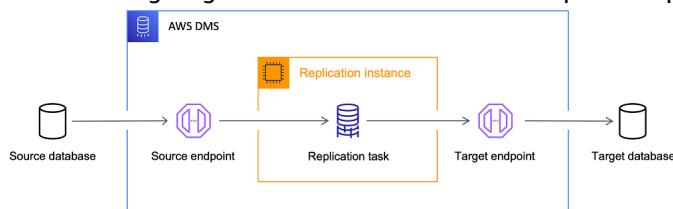
AWS Database Migration Service (AWS DMS) is a cloud service that makes it possible to migrate relational databases, data warehouses, NoSQL databases, and other types of data stores. You can use AWS DMS to migrate your data into the AWS Cloud or between combinations of cloud and on-premises setups.

With AWS DMS, you can discover your source data stores, convert your source schemas, and migrate your data.

- To discover your source data infrastructure, you can use DMS Fleet Advisor. This service collects data from your on-premises database and analytic servers, and builds an inventory of servers, databases, and schemas that you can migrate to the AWS Cloud.
- To migrate to a different database engine, you can use DMS Schema Conversion. This service automatically assesses and converts your source schemas to a new target engine. Alternatively, you can download the AWS Schema Conversion Tool (AWS SCT) to your local PC to convert your source schemas.
- After you convert your source schemas and apply the converted code to your target database, you can use AWS DMS to migrate your data. You can perform one-time migrations or replicate ongoing changes to keep sources and targets in sync. Because AWS DMS is a part of the AWS Cloud, you get the cost efficiency, speed to market, security, and flexibility that AWS services offer.

At a basic level, AWS DMS is a server in the AWS Cloud that runs replication software. You create a source and target connection to tell AWS DMS where to extract from and load to. Then you schedule a task that runs on this server to move your data. AWS DMS creates the tables and associated primary keys if they don't exist on the target. You can create the target tables yourself if you prefer. Or you can use AWS Schema Conversion Tool (AWS SCT) to create some or all of the target tables, indexes, views, triggers, and so on.

The following diagram illustrates the AWS DMS replication process.



For information about what AWS Regions support AWS DMS, see [Working with an AWS DMS replication instance \(p. 100\)](#). For information on the cost of database migration, see the [AWS Database Migration Service pricing page](#). For information about AWS DMS features and benefits, see [AWS Database Migration Service Features](#).

## Migration tasks that AWS DMS performs

AWS DMS takes over many of the difficult or tedious tasks involved in a migration project:

- In a traditional solution, you need to perform capacity analysis, procure hardware and software, install and administer systems, and test and debug the installation. AWS DMS automatically manages the

deployment, management, and monitoring of all hardware and software needed for your migration. Your migration can be up and running within minutes of starting the AWS DMS configuration process.

- With AWS DMS, you can scale up (or scale down) your migration resources as needed to match your actual workload. For example, if you determine that you need additional storage, you can easily increase your allocated storage and restart your migration, usually within minutes.
- AWS DMS uses a pay-as-you-go model. You only pay for AWS DMS resources while you use them, as opposed to traditional licensing models with up-front purchase costs and ongoing maintenance charges.
- AWS DMS automatically manages all of the infrastructure that supports your migration server, including hardware and software, software patching, and error reporting.
- AWS DMS provides automatic failover. If your primary replication server fails for any reason, a backup replication server can take over with little or no interruption of service.
- AWS DMS Fleet Advisor automatically inventories your data infrastructure. It creates reports that help you identify migration candidates and plan your migration.
- AWS DMS Schema Conversion automatically assesses the complexity of your migration for your source data provider. It also converts database schemas and code objects to a format compatible with the target database and then applies the converted code.
- AWS DMS can help you switch to a modern, perhaps more cost-effective, database engine than the one you are running now. For example, AWS DMS can help you take advantage of the managed database services provided by Amazon Relational Database Service (Amazon RDS) or Amazon Aurora. Or it can help you move to the managed data warehouse service provided by Amazon Redshift, NoSQL platforms like Amazon DynamoDB, or low-cost storage platforms like Amazon Simple Storage Service (Amazon S3). Conversely, if you want to migrate away from old infrastructure but continue to use the same database engine, AWS DMS also supports that process.
- AWS DMS supports nearly all of today's most popular DBMS engines as source endpoints. For more information, see [Sources for data migration \(p. 134\)](#).
- AWS DMS provides a broad coverage of available target engines. For more information, see [Targets for data migration \(p. 293\)](#).
- You can migrate from any of the supported data sources to any of the supported data targets. AWS DMS supports fully heterogeneous data migrations between the supported engines.
- AWS DMS ensures that your data migration is secure. Data at rest is encrypted with AWS Key Management Service (AWS KMS) encryption. During migration, you can use Secure Socket Layers (SSL) to encrypt your in-flight data as it travels from source to target.

# How AWS Database Migration Service works

AWS Database Migration Service (AWS DMS) is a web service that you can use to migrate data from a source data store to a target data store. These two data stores are called endpoints. You can migrate between source and target endpoints that use the same database engine, such as from an Oracle database to an Oracle database. You can also migrate between source and target endpoints that use different database engines, such as from an Oracle database to a PostgreSQL database. The only requirement to use AWS DMS is that one of your endpoints must be on an AWS service. You can't use AWS DMS to migrate from an on-premises database to another on-premises database.

For information on the cost of database migration, see the [AWS Database Migration Service pricing page](#).

Use the following topics to better understand AWS DMS.

## Topics

- [High-level view of AWS DMS \(p. 3\)](#)
- [Components of AWS DMS \(p. 4\)](#)
- [Sources for AWS DMS \(p. 8\)](#)
- [Targets for AWS DMS \(p. 10\)](#)
- [Constructing an Amazon Resource Name \(ARN\) for AWS DMS \(p. 11\)](#)
- [Using AWS DMS with other AWS services \(p. 12\)](#)

## High-level view of AWS DMS

To perform a database migration, AWS DMS connects to the source data store, reads the source data, and formats the data for consumption by the target data store. It then loads the data into the target data store. Most of this processing happens in memory, though large transactions might require some buffering to disk. Cached transactions and log files are also written to disk.

At a high level, when using AWS DMS you do the following:

- Discover databases in your network environment that are good candidates for migration.
- Automatically convert your source database schemas and most of the database code objects to a format compatible with the target database.
- Create a replication server.
- Create source and target endpoints that have connection information about your data stores.
- Create one or more migration tasks to migrate data between the source and target data stores.

A task can consist of three major phases:

- Migration of existing data (Full load)
- The application of cached changes

- Ongoing replication (Change Data Capture)

During a full load migration, where existing data from the source is moved to the target, AWS DMS loads data from tables on the source data store to tables on the target data store. While the full load is in progress, any changes made to the tables being loaded are cached on the replication server; these are the cached changes. It's important to note that AWS DMS doesn't capture changes for a given table until the full load for that table is started. In other words, the point when change capture starts is different for each individual table.

When the full load for a given table is complete, AWS DMS immediately begins to apply the cached changes for that table. Once the table is loaded and the cached changes applied, AWS DMS begins to collect changes as transactions for the ongoing replication phase. If a transaction has tables not yet fully loaded, the changes are stored locally on the replication instance. After AWS DMS applies all cached changes, tables are transactionally consistent. At this point, AWS DMS moves to the ongoing replication phase, applying changes as transactions.

At the start of the ongoing replication phase, a backlog of transactions generally causes some lag between the source and target databases. The migration eventually reaches a steady state after working through this backlog of transactions. At this point, you can shut down your applications, allow any remaining transactions to be applied to the target, and bring your applications up, now pointing at the target database.

AWS DMS creates the target schema objects necessary to perform a data migration. You can use AWS DMS to take a minimalist approach and create only those objects required to efficiently migrate the data. Using this approach, AWS DMS creates tables, primary keys, and in some cases unique indexes, but it won't create any other objects that are not required to efficiently migrate the data from the source.

Alternatively, you can use DMS Schema Conversion within AWS DMS to automatically convert your source database schemas and most of the database code objects to a format compatible with the target database. This conversion includes tables, views, stored procedures, functions, data types, synonyms, and so on. Any objects that DMS Schema Conversion can't convert automatically are clearly marked. To complete the migration, you can convert these objects manually.

## Components of AWS DMS

This section describes the internal components of AWS DMS and how they function together to accomplish your data migration. Understanding the underlying components of AWS DMS can help you migrate data more efficiently and provide better insight when troubleshooting or investigating issues.

An AWS DMS migration consists of five components: discovery of databases to migrate, automatic schema conversion, a replication instance, source and target endpoints, and a replication task. You create an AWS DMS migration by creating the necessary replication instance, endpoints, and tasks in an AWS Region.

### Database discovery

DMS Fleet Advisor collects data from multiple database environments to provide insight into your data infrastructure. DMS Fleet Advisor collects data from your on-premises database and analytic servers from one or more central locations without the need to install it on every computer. Currently, DMS Fleet Advisor supports Microsoft SQL Server, MySQL, Oracle, and PostgreSQL database servers.

Based on data discovered from your network, DMS Fleet Advisor builds an inventory that you can review to determine which database servers and objects to monitor. As details about these servers,

databases, and schemas are collected, you can analyze the feasibility of your intended database migrations.

### Schema and code migration

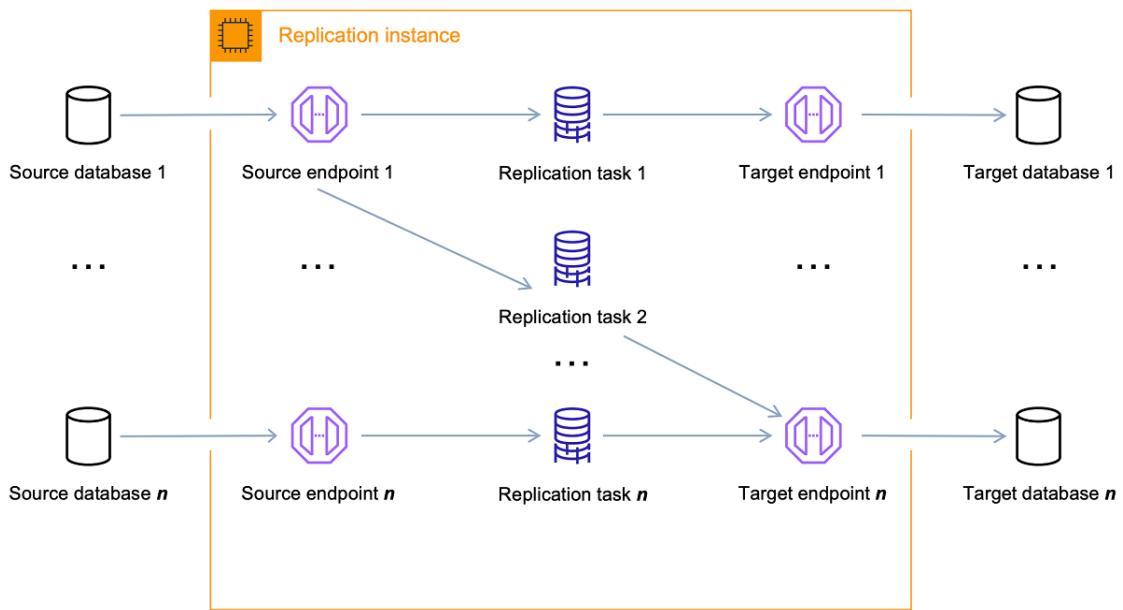
DMS Schema Conversion in AWS DMS makes database migrations between different types of databases more predictable. You can use DMS Schema Conversion to evaluate the complexity of your migration for your source data provider, and then use it to convert database schemas and code objects. You can then apply the converted code to your target database.

At a high level, DMS Schema Conversion operates with the following three components: instance profiles, data providers, and migration projects. An *instance profile* specifies network and security settings. A *data provider* stores database connection credentials. A *migration project* contains data providers, an instance profile, and migration rules. AWS DMS uses data providers and an instance profile to design a process that converts database schemas and code objects.

### Replication instance

At a high level, an AWS DMS replication instance is simply a managed Amazon Elastic Compute Cloud (Amazon EC2) instance that hosts one or more replication tasks.

The figure following shows an example replication instance running several associated replication tasks.



A single replication instance can host one or more replication tasks, depending on the characteristics of your migration and the capacity of the replication server. AWS DMS provides a variety of replication instances so you can choose the optimal configuration for your use case. For more information about the various classes of replication instances, see [Choosing the right AWS DMS replication instance for your migration \(p. 103\)](#).

AWS DMS creates the replication instance on an Amazon EC2 instance. Some of the smaller instance classes are sufficient for testing the service or for small migrations. If your migration involves a large number of tables, or if you intend to run multiple concurrent replication tasks, you should consider using one of the larger instances. We recommend this approach because AWS DMS can consume a significant amount of memory and CPU.

Depending on the Amazon EC2 instance class you select, your replication instance comes with either 50 GB or 100 GB of data storage. This amount is usually sufficient for most customers. However, if

your migration involves large transactions or a high-volume of data changes then you might want to increase the base storage allocation. Change data capture (CDC) might cause data to be written to disk, depending on how fast the target can write the changes. As log files are also written to the disk, increasing the level of severity for logging will also lead to a higher storage consumption.

AWS DMS can provide high availability and failover support using a Multi-AZ deployment. In a Multi-AZ deployment, AWS DMS automatically provisions and maintains a standby replica of the replication instance in a different Availability Zone. The primary replication instance is synchronously replicated to the standby replica. If the primary replication instance fails or becomes unresponsive, the standby resumes any running tasks with minimal interruption. Because the primary is constantly replicating its state to the standby, Multi-AZ deployment does incur some performance overhead.

For more detailed information about the AWS DMS replication instance, see [Working with an AWS DMS replication instance \(p. 100\)](#).

### Endpoint

AWS DMS uses an endpoint to access your source or target data store. The specific connection information is different, depending on your data store, but in general you supply the following information when you create an endpoint:

- Endpoint type – Source or target.
- Engine type – Type of database engine, such as Oracle or PostgreSQL.
- Server name – Server name or IP address that AWS DMS can reach.
- Port – Port number used for database server connections.
- Encryption – Secure Socket Layer (SSL) mode, if SSL is used to encrypt the connection.
- Credentials – User name and password for an account with the required access rights.

When you create an endpoint using the AWS DMS console, the console requires that you test the endpoint connection. The test must be successful before using the endpoint in a AWS DMS task. Like the connection information, the specific test criteria are different for different engine types. In general, AWS DMS verifies that the database exists at the given server name and port, and that the supplied credentials can be used to connect to the database with the necessary privileges to perform a migration. If the connection test is successful, AWS DMS downloads and stores schema information to use later during task configuration. Schema information might include table definitions, primary key definitions, and unique key definitions, for example.

More than one replication task can use a single endpoint. For example, you might have two logically distinct applications hosted on the same source database that you want to migrate separately. In this case, you create two replication tasks, one for each set of application tables. You can use the same AWS DMS endpoint in both tasks.

You can customize the behavior of an endpoint by using extra connection attributes. *Extra connection attributes* can control various behavior such as logging detail, file size, and other parameters. Each data store engine type has different extra connection attributes available. You can find the specific extra connection attributes for each data store in the source or target section for that data store. For a list of supported source and target data stores, see [Sources for AWS DMS \(p. 8\)](#) and [Targets for AWS DMS \(p. 10\)](#).

For more detailed information about AWS DMS endpoints, see [Working with AWS DMS endpoints \(p. 131\)](#).

### Replication tasks

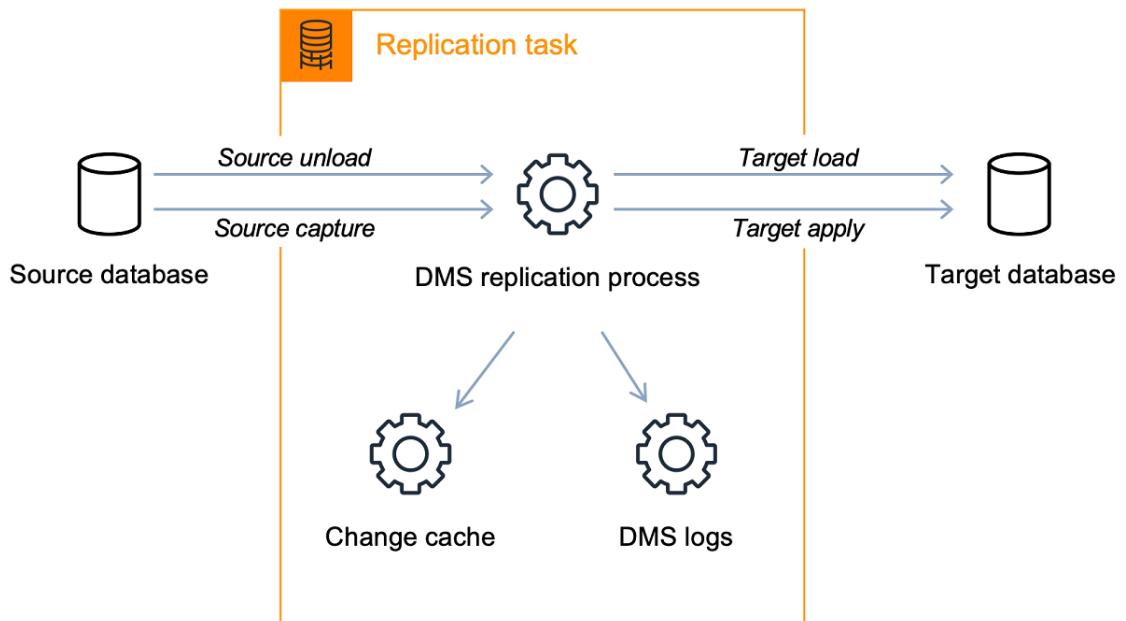
You use an AWS DMS replication task to move a set of data from the source endpoint to the target endpoint. Creating a replication task is the last step you need to take before you start a migration.

When you create a replication task, you specify the following task settings:

- Replication instance – the instance to host and run the task
- Source endpoint
- Target endpoint
- Migration type options, as listed following. For a full explanation of the migration type options, see [Creating a task \(p. 442\)](#).
  - Full load (Migrate existing data) – If you can afford an outage long enough to copy your existing data, this option is a good one to choose. This option simply migrates the data from your source database to your target database, creating tables when necessary.
  - Full load + CDC (Migrate existing data and replicate ongoing changes) – This option performs a full data load while capturing changes on the source. After the full load is complete, captured changes are applied to the target. Eventually, the application of changes reaches a steady state. At this point, you can shut down your applications, let the remaining changes flow through to the target, and then restart your applications pointing at the target.
  - CDC only (Replicate data changes only) – In some situations, it might be more efficient to copy existing data using a method other than AWS DMS. For example, in a homogeneous migration, using native export and import tools might be more efficient at loading bulk data. In this situation, you can use AWS DMS to replicate changes starting when you start your bulk load to bring and keep your source and target databases in sync.
- Target table preparation mode options, as listed following. For a full explanation of target table modes, see [Creating a task \(p. 442\)](#).
  - Do nothing – AWS DMS assumes that the target tables are precreated on the target.
  - Drop tables on target – AWS DMS drops and recreates the target tables.
  - Truncate – If you created tables on the target, AWS DMS truncates them before the migration starts. If no tables exist and you select this option, AWS DMS creates any missing tables.
- LOB mode options, as listed following. For a full explanation of LOB modes, see [Setting LOB support for source databases in an AWS DMS task \(p. 479\)](#).
  - Don't include LOB columns – LOB columns are excluded from the migration.
  - Full LOB mode – Migrate complete LOBs regardless of size. AWS DMS migrates LOBs piecewise in chunks controlled by the **Max LOB Size** parameter. This mode is slower than using limited LOB mode.
  - Limited LOB mode – Truncate LOBs to the value specified by the **Max LOB Size** parameter. This mode is faster than using full LOB mode.
- Table mappings – indicates the tables to migrate and how they are migrated. For more information, see [Using table mapping to specify task settings \(p. 491\)](#).
- Data transformations, as listed following. For more information on data transformations, see [Specifying table selection and transformations rules using JSON \(p. 494\)](#).
  - Changing schema, table, and column names.
  - Changing tablespace names (for Oracle target endpoints).
  - Defining primary keys and unique indexes on the target.
- Data validation
- Amazon CloudWatch logging

You use the task to migrate data from the source endpoint to the target endpoint, and the task processing is done on the replication instance. You specify what tables and schemas to migrate and any special processing, such as logging requirements, control table data, and error handling.

Conceptually, an AWS DMS replication task performs two distinct functions as shown in the diagram following.



The full load process is straight-forward to understand. Data is extracted from the source in a bulk extract manner and loaded directly into the target. You can specify the number of tables to extract and load in parallel on the AWS DMS console under **Advanced Settings**.

For more information about AWS DMS tasks, see [Working with AWS DMS tasks \(p. 440\)](#).

#### Ongoing replication, or change data capture (CDC)

You can also use an AWS DMS task to capture ongoing changes to the source data store while you are migrating your data to a target. The change capture process that AWS DMS uses when replicating ongoing changes from a source endpoint collects changes to the database logs by using the database engine's native API.

In the CDC process, the replication task is designed to stream changes from the source to the target, using in-memory buffers to hold data in-transit. If the in-memory buffers become exhausted for any reason, the replication task will spill pending changes to the Change Cache on disk. This could occur, for example, if AWS DMS is capturing changes from the source faster than they can be applied on the target. In this case, you will see the task's *target latency* exceed the task's *source latency*.

You can check this by navigating to your task on the AWS DMS console, and opening the Task Monitoring tab. The CDCLatencyTarget and CDCLatencySource graphs are shown at the bottom of the page. If you have a task that is showing target latency then there is likely some tuning on the target endpoint needed to increase the application rate.

The replication task also uses storage for task logs as discussed preceding. The disk space that comes pre-configured with your replication instance is usually sufficient for logging and spilled changes. If you need additional disk space, for example, when using detailed debugging to investigate a migration issue, you can modify the replication instance to allocate more space.

## Sources for AWS DMS

You can use the following data stores as source endpoints for data migration using AWS DMS.

## On-premises and EC2 instance databases

- Oracle versions 10.2 and later (for versions 10.x), 11g and up to 12.2, 18c, and 19c for the Enterprise, Standard, Standard One, and Standard Two editions
- Microsoft SQL Server versions 2005, 2008, 2008R2, 2012, 2014, 2016, 2017, and 2019 for the Enterprise, Standard, Workgroup, and Developer editions. The Web and Express editions are not supported.
- MySQL versions 5.5, 5.6, 5.7, and 8.0.

### Note

Support for MySQL 8.0 as a source is available in AWS DMS versions 3.4.0 and later, except when the transaction payload is compressed.

- MariaDB (supported as a MySQL-compatible data source) versions 10.0 (only versions 10.0.24 and later), 10.2, 10.3, 10.4, and 10.5.

### Note

Support for MariaDB as a source is available in all AWS DMS versions where MySQL is supported.

- PostgreSQL version 9.4 and later (for versions 9.x), 10.x, 11.x, 12.x, 13.x and 14.x.
- MongoDB versions 3.x, 4.0, 4.2, and 4.4.
- SAP Adaptive Server Enterprise (ASE) versions 12.5, 15, 15.5, 15.7, 16 and later.
- IBM Db2 for Linux, UNIX, and Windows (Db2 LUW) versions:
  - Version 9.7, all fix packs are supported.
  - Version 10.1, all fix packs are supported.
  - Version 10.5, all fix packs except for Fix Pack 5 are supported.
  - Version 11.1, all fix packs are supported.
  - Version 11.5, Mods (0-7) with only Fix Pack Zero supported
  - IBM Db2 for z/OS version 12

## Third-party managed database services:

- Microsoft Azure SQL Database
- Google Cloud for MySQL

## Amazon RDS instance databases, and Amazon Simple Storage Service (Amazon S3)

- Oracle versions 11g (versions 11.2.0.4 and later) and up to 12.2, 18c, and 19c for the Enterprise, Standard, Standard One, and Standard Two editions.
- Microsoft SQL Server versions 2012, 2014, 2016, 2017, and 2019 for the Enterprise, Standard, Workgroup, and Developer editions. The Web and Express editions are not supported.
- MySQL versions 5.5, 5.6, 5.7, and 8.0.

### Note

Support for MySQL 8.0 as a source is available in AWS DMS versions 3.4.0 and later, except when the transaction payload is compressed.

- MariaDB (supported as a MySQL-compatible data source) versions 10.0.24 to 10.0.28, 10.1, 10.2, and 10.3, 10.3.13 and 10.4

### Note

Support for MariaDB as a source is available in all AWS DMS versions where MySQL is supported.

- PostgreSQL version 10.x, 11.x, 12.x, 13.x, and 14.x.
- Amazon Aurora with MySQL compatibility (supported as a MySQL-compatible data source).

- Amazon Aurora with PostgreSQL compatibility (supported as a PostgreSQL-compatible data source).
- Amazon S3.
- Amazon DocumentDB (with MongoDB compatibility)

For information about working with a specific source, see [Working with AWS DMS endpoints \(p. 131\)](#).

## Targets for AWS DMS

You can use the following data stores as target endpoints for data migration using AWS DMS.

### On-premises and Amazon EC2 instance databases

- Oracle versions 10g, 11g, 12c, 18c, and 19c for the Enterprise, Standard, Standard One, and Standard Two editions.
- Microsoft SQL Server versions 2005, 2008, 2008R2, 2012, 2014, 2016, 2017, and 2019 for the Enterprise, Standard, Workgroup, and Developer editions. The Web and Express editions are not supported.
- MySQL versions 5.5, 5.6, 5.7, and 8.0.
- MariaDB (supported as a MySQL-compatible data target) versions 10.0.24 to 10.0.28, 10.1, 10.2, 10.3 and 10.4.

#### Note

Support for MariaDB as a target is available in all AWS DMS versions where MySQL is supported.

- PostgreSQL version 9.4 and later (for versions 9.x), 10.x, 11.x, 12.x, 13.x, and 14.x.
- SAP Adaptive Server Enterprise (ASE) versions 15, 15.5, 15.7, 16 and later.
- Redis versions 6.x.

### Amazon RDS instance databases, Amazon Redshift, Amazon DynamoDB, Amazon S3, Amazon OpenSearch Service, Amazon ElastiCache for Redis, Amazon Kinesis Data Streams, Amazon DocumentDB, Amazon Neptune, and Apache Kafka

- Oracle versions 11g (versions 11.2.0.3.v1 and later), 12c, 18c, and 19c for the Enterprise, Standard, Standard One, and Standard Two editions.
- Microsoft SQL Server versions 2012, 2014, 2016, 2017, and 2019 for the Enterprise, Standard, Workgroup, and Developer editions. The Web and Express editions are not supported.
- MySQL versions 5.5, 5.6, 5.7, and 8.0.
- MariaDB (supported as a MySQL-compatible data target) versions 10.0.24 to 10.0.28, 10.1, 10.2, 10.3 and 10.4.

#### Note

Support for MariaDB as a target is available in all AWS DMS versions where MySQL is supported.

- PostgreSQL version 10.x, 11.x, 12.x, 13.x, and 14.x.
- Amazon Aurora MySQL-Compatible Edition
- Amazon Aurora PostgreSQL-Compatible Edition
- Amazon Aurora Serverless v2
- Amazon Redshift
- Amazon S3
- Amazon DynamoDB

- Amazon OpenSearch Service
- Amazon ElastiCache for Redis
- Amazon Kinesis Data Streams
- Amazon DocumentDB (with MongoDB compatibility)
- Amazon Neptune
- Apache Kafka – [Amazon Managed Streaming for Apache Kafka \(Amazon MSK\)](#) and [self-managed Apache Kafka](#)
- Babelfish (version 1.2.0) for Aurora PostgreSQL (versions 13.4, 13.5, 13.6)

For information about working with a specific target, see [Working with AWS DMS endpoints \(p. 131\)](#).

## Constructing an Amazon Resource Name (ARN) for AWS DMS

If you use the AWS CLI or AWS DMS API to automate your database migration, then you work with Amazon Resource Name (ARNs). Each resource that is created in Amazon Web Services is identified by an ARN, which is a unique identifier. If you use the AWS CLI or AWS DMS API to set up your database migration, you supply the ARN of the resource that you want to work with.

An ARN for an AWS DMS resource uses the following syntax:

`arn:aws:dms:region:account number:resourcetype:resourcename`

In this syntax, the following apply:

- *region* is the ID of the AWS Region where the AWS DMS resource was created, such as us-west-2.

The following table shows AWS Region names and the values that you should use when constructing an ARN.

Region	Name
Asia Pacific (Tokyo) Region	ap-northeast-1
Asia Pacific (Seoul) Region	ap-northeast-2
Asia Pacific (Mumbai) Region	ap-south-1
Asia Pacific (Singapore) Region	ap-southeast-1
Asia Pacific (Sydney) Region	ap-southeast-2
Canada (Central) Region	ca-central-1
China (Beijing) Region	cn-north-1
China (Ningxia) Region	cn-northwest-1
Europe (Stockholm) Region	eu-north-1
Europe (Milan) Region	eu-south-1
EU (Frankfurt) Region	eu-central-1
Europe (Ireland) Region	eu-west-1

Region	Name
EU (London) Region	eu-west-2
EU (Paris) Region	eu-west-3
South America (São Paulo) Region	sa-east-1
US East (N. Virginia) Region	us-east-1
US East (Ohio) Region	us-east-2
US West (N. California) Region	us-west-1
US West (Oregon) Region	us-west-2

- *account number* is your account number with dashes omitted. To find your account number, sign in to your AWS account at <http://aws.amazon.com>, choose **My Account/Console**, and then choose **My Account**.
- *resourcetype* is the type of AWS DMS resource.

The following table shows the resource types to use when constructing an ARN for a particular AWS DMS resource.

AWS DMS resource type	ARN format
Replication instance	<code>arn:aws:dms:<i>region</i>:<i>account</i>:rep: <i>resourcename</i></code>
Endpoint	<code>arn:aws:dms:<i>region</i>:<i>account</i>:endpoint: <i>resourcename</i></code>
Replication task	<code>arn:aws:dms:<i>region</i>:<i>account</i>:task: <i>resourcename</i></code>
Subnet group	<code>arn:aws:dms:<i>region</i>:<i>account</i>:subgrp: <i>resourcename</i></code>

- *resourcename* is the resource name assigned to the AWS DMS resource. This is a generated arbitrary string.

The following table shows examples of ARNs for AWS DMS resources. Here, we assume an AWS account of 123456789012, which were created in the US East (N. Virginia) Region, and has a resource name.

Resource type	Sample ARN
Replication instance	<code>arn:aws:dms:us-east-1:123456789012:rep:QLXQZ64MH7CXF4QCQMGGRVYVXAI</code>
Endpoint	<code>arn:aws:dms:us-east-1:123456789012:endpoint:D3HMZ2IGUCGFF3NTAXUXGF6S5A</code>
Replication task	<code>arn:aws:dms:us-east-1:123456789012:task:2PVREMWNPGYJCVU2IBPTOYTIV4</code>
Subnet group	<code>arn:aws:dms:us-east-1:123456789012:subgrp:test-tag-grp</code>

## Using AWS DMS with other AWS services

You can use AWS DMS with several other AWS services:

- You can use an Amazon EC2 instance or Amazon RDS DB instance as a target for a data migration.
- You can use the AWS Schema Conversion Tool (AWS SCT) to convert your source schema and SQL code into an equivalent target schema and SQL code.
- You can use Amazon S3 as a storage site for your data, or you can use it as an intermediate step when migrating large amounts of data.
- You can use AWS CloudFormation to set up your AWS resources for infrastructure management or deployment. For example, you can provision AWS DMS resources such as replication instances, tasks, certificates, and endpoints. You create a template that describes all the AWS resources that you want, and AWS CloudFormation provisions and configures those resources for you.

## AWS DMS support for AWS CloudFormation

You can provision AWS DMS resources using AWS CloudFormation. AWS CloudFormation is a service that helps you model and set up your AWS resources for infrastructure management or deployment. For example, you can provision AWS DMS resources such as replication instances, tasks, certificates, and endpoints. You create a template that describes all the AWS resources that you want and AWS CloudFormation provisions and configures those resources for you.

As a developer or system administrator, you can create and manage collections of these resources that you can then use for repetitive migration tasks or deploying resources to your organization. For more information about AWS CloudFormation, see [AWS CloudFormation concepts](#) in the *AWS CloudFormation User Guide*.

AWS DMS supports creating the following AWS DMS resources using AWS CloudFormation:

- [AWS::DMS::Certificate](#)
- [AWS::DMS::Endpoint](#)
- [AWS::DMS::EventSubscription](#)
- [AWS::DMS::ReplicationInstance](#)
- [AWS::DMS::ReplicationSubnetGroup](#)
- [AWS::DMS::ReplicationTask](#)

# Getting started with AWS Database Migration Service

In the following tutorial, you can find out how to perform a database migration with AWS Database Migration Service (AWS DMS).

To perform a database migration, take the following steps:

1. Set up your AWS account by following the steps in [Setting up for AWS Database Migration Service \(p. 14\)](#).
2. Create your sample databases and an Amazon EC2 client to populate your source database and test replication. Also, create a virtual private cloud (VPC) based on the Amazon Virtual Private Cloud (Amazon VPC) service to contain your tutorial resources. To create these resources, follow the steps in [Prerequisites for AWS Database Migration Service \(p. 16\)](#).
3. Populate your source database using a [sample database creation script](#).
4. Use DMS Schema Conversion or the AWS Schema Conversion Tool (AWS SCT) to convert the schema from the source database to the target database. To use DMS Schema Conversion, follow the steps in [Getting started with DMS Schema Conversion \(p. 56\)](#). To convert the schema with AWS SCT, follow the steps in [Migrate schema \(p. 22\)](#).
5. Create a replication instance to perform all the processes for the migration. To do this and the following tasks, take the steps in [Replication \(p. 24\)](#).
6. Specify source and target database endpoints. For information about creating endpoints, see [Creating source and target endpoints \(p. 131\)](#).
7. Create a task to define what tables and replication processes you want to use, and start replication. For information about creating database migration tasks, see [Creating a task \(p. 442\)](#).
8. Verify that replication is working by running queries on the target database.

## Setting up for AWS Database Migration Service

Before you use AWS Database Migration Service (AWS DMS) for the first time, complete the following tasks.

- [Sign up for AWS \(p. 14\)](#)
- [Create an IAM user \(p. 15\)](#)

### Sign up for AWS

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

With AWS DMS, you pay only for the resources you use. The AWS DMS replication instance that you create will be live (not running in a sandbox). You incur the standard AWS DMS usage fees for the instance until you stop it. For more information about AWS DMS usage rates, see the [AWS DMS product page](#). If you are a new AWS customer, you can get started with AWS DMS for free; for more information, see [AWS free usage tier](#).

If you close your AWS account, all AWS DMS resources and configurations associated with your account are deleted after two days. These resources include all replication instances, source and target endpoint

configuration, replication tasks, and Secure Sockets Layer (SSL) certificates. If you decide to use AWS DMS again later than two days after closing your account, you recreate the resources that you need.

If you have an AWS account already, skip to the next task.

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

### To sign up for an AWS account

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

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

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

Note your AWS account number, because you need it for the next task.

## Create an IAM user

Services in AWS, such as AWS DMS, require that you provide credentials when you access them. Doing this allows the service to determine whether you have permission to access its resources. The console requires your password. You can create access keys for your AWS account to access the AWS Command Line Interface (AWS CLI) or the AWS DMS API.

However, we don't recommend that you access AWS using the credentials for your AWS account. Instead, we recommend that you use AWS Identity and Access Management (IAM) instead. Create an IAM user, and then add the user to an IAM group with administrative permissions and grant this user administrative permissions. Your account needs administrative permissions to create replication tasks that access endpoint resources. You can then access AWS using a special URL and the credentials for the IAM user.

If you signed up for AWS but have not created an IAM user for yourself, you can create one using the IAM console.

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

Choose one way to manage your administrator user	To	By	You can also
In IAM Identity Center  (Recommended) <small>This aligns with the security best practices. For information about best practices, see <a href="#">Security best practices</a></small>	Use short-term credentials to access AWS.	Following the instructions in <a href="#">Getting started</a> in the <i>AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide</i> .	Configure programmatic access by <a href="#">Configuring the AWS CLI to use AWS IAM Identity Center (successor to AWS Single Sign-On)</a> in the <i>AWS Command Line Interface User Guide</i> .

<b>Choose one way to manage your administration</b>	<b>To</b>	<b>By</b>	<b>You can also</b>
	<a href="#">in IAM</a> in the <i>IAM User Guide</i> .		
In IAM (Not recommended)	Use long-term credentials to access AWS.	Following the instructions in <a href="#">Creating your first IAM admin user and user group</a> in the <i>IAM User Guide</i> .	Configure programmatic access by <a href="#">Managing access keys for IAM users</a> in the <i>IAM User Guide</i> .

To sign in as this new IAM user, sign out of the AWS console. Then use the following URL, where [\*your\\_aws\\_account\\_id\*](#) is your AWS account number without the hyphens. For example, if your AWS account number is 1234-5678-9012, your AWS account ID is 123456789012.

[https://\*your\\_aws\\_account\\_id\*.signin.aws.amazon.com/console/](https://your_aws_account_id.signin.aws.amazon.com/console/)

Enter the IAM user name and password that you just created. When you're signed in, the navigation bar displays "[\*your\\_user\\_name@your\\_aws\\_account\\_id\*](#)".

If you don't want the URL for your sign-in page to contain your AWS account ID, you can create an account alias. On the IAM dashboard, choose **Customize** and type an alias, such as your company name. To sign in after you create an account alias, use the following URL.

[https://\*your\\_account\\_alias\*.signin.aws.amazon.com/console/](https://your_account_alias.signin.aws.amazon.com/console/)

To verify the sign-in link for IAM users for your account, open the IAM console and check under **AWS Account Alias** on the dashboard.

## Prerequisites for AWS Database Migration Service

In this section, you can learn the prerequisite tasks for AWS DMS, such as setting up your source and target databases. As part of these tasks, you also set up a virtual private cloud (VPC) based on the Amazon VPC service to contain your resources. In addition, you set up an Amazon EC2 instance that you use to populate your source database and verify replication on your target database.

**Note**

Populating the source database takes up to 45 minutes.

For this tutorial, you create a MySQL database as your source, and a PostgreSQL database as your target. This scenario uses commonly used, low-cost database engines to demonstrate replication. Using different database engines demonstrates AWS DMS features for migrating data between heterogeneous platforms.

The resources in this tutorial use the US West (Oregon) Region. If you want to use a different AWS Region, specify your chosen Region instead wherever US West (Oregon) appears.

**Note**

For the sake of simplicity, the databases that you create for this tutorial don't use encryption or other advanced security features. You must use security features to keep your production databases secure. For more information, see [Security in Amazon RDS](#).

For prerequisite steps, see the following topics.

#### Topics

- [Create a VPC \(p. 17\)](#)
- [Create Amazon RDS parameter groups \(p. 18\)](#)
- [Create your source Amazon RDS database \(p. 18\)](#)
- [Create your target Amazon RDS database \(p. 19\)](#)
- [Create an Amazon EC2 client \(p. 20\)](#)
- [Populate your source database \(p. 21\)](#)

## Create a VPC

In this section, you create a VPC to contain your AWS resources. Using a VPC is a best practice when using AWS resources, so that your databases, Amazon EC2 instances, security groups, and so on, are logically organized and secure.

Using a VPC for your tutorial resources also ensures that you delete all of the resources you use when you are done with the tutorial. You must delete all of the resources that a VPC contains before you can delete the VPC.

#### To create a VPC for use with AWS DMS

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. On the navigation pane, choose **VPC Dashboard**, and then choose **Create VPC**.
3. On the **Create VPC** page, enter the following options:
  - **Resources to create:** **VPC and more**
  - **Name tag auto generation:** Choose **Auto-generate**, and enter **DMSVPC**.
  - **IPv4 block:** **10.0.1.0/24**
  - **IPv6 CIDR block:** **No IPv6 CIDR block**
  - **Tenancy:** **Default**
  - **Number of availability zones:** **2**
  - **Number of public subnets:** **2**
  - **Number of private subnets:** **2**
  - **NAT gateways (\$):** **None**
  - **VPC endpoints:** **None**
- Choose **Create VPC**.
4. On the navigation pane, choose **Your VPCs**. Note the VPC ID for **DMSVPC**.
5. On the navigation pane, choose **Security Groups**.
6. Choose the group named **default** that has a **VPC ID** that matches the ID that you noted for **DMSVPC**.
7. Choose the **Inbound rules** tab, and choose **Edit inbound rules**.
8. Choose **Add rule**. Add a rule of type **MySQL/Aurora** and choose **Anywhere-IPv4** for **Source**.
9. Choose **Add rule** again. Add a rule of type **PostgreSQL** and choose **Anywhere-IPv4** for **Source**.
10. Choose **Save rules**.

## Create Amazon RDS parameter groups

To specify settings for your source and target databases for AWS DMS, use Amazon RDS parameter groups. To allow initial and ongoing replication between your databases, make sure to configure the following:

- Your source database's binary log, so that AWS DMS can determine what incremental updates it needs to replicate.
- Your target database's replication role, so that AWS DMS ignores foreign key constraints during the initial data transfer. With this setting, AWS DMS can migrate data out of order.

### To create parameter groups for use with AWS DMS

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. On the navigation pane, choose **Parameter groups**.
3. On the **Parameter groups** page, choose **Create parameter group**.
4. On the **Create parameter group** page, enter the following settings:
  - **Parameter group family:** mysql8.0
  - **Group name:** dms-mysql-parameters
  - **Description:** Group for specifying binary log settings for replication
- Choose **Create**.
5. On the **Parameter groups** page, choose **dms-mysql-parameters**, and on the **dms-parameter-group** page, choose **Edit parameters**.
6. Set the following parameters to the following values:
  - **binlog\_checksum:** NONE
  - **binlog\_format:** ROW
- Choose **Save changes**.
7. On the **Parameter groups** page, choose **Create parameter group** again.
8. On the **Create parameter group** page, enter the following settings:
  - **Parameter group family:** postgres13
  - **Group name:** dms-postgresql-parameters
  - **Description:** Group for specifying role setting for replication
- Choose **Create**.
9. On the **Parameter groups** page, choose **dms-postgresql-parameters**.
10. On the **dms-postgresql-parameters** page, choose **Edit parameters**, and set **session\_replication\_role** parameter to **replica**. Note that the **session\_replication\_role** parameter is not on the first page of parameters. Use the pagination controls or the search field to find the parameter.
11. Choose **Save changes**.

## Create your source Amazon RDS database

Use the following procedure to create your source Amazon RDS database.

### To create your source RDS for MySQL database

1. Open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. On the **Dashboard** page, choose **Create Database** in the **Database** section. Don't choose **Create Database** in the **Amazon Aurora** section at the top of the page.
3. On the **Create database** page, set the following options:
  - **Choose a database creation method:** Choose **Standard Create**.
  - **Engine options:** For **Engine type**, choose **MySQL**. For **Version**, leave **MySQL 8.0.28** selected.
  - **Templates:** Choose **Dev/Test**.
  - **Availability and Durability:** Leave **Multi-AZ DB Instance** selected.
  - **Settings:**
    - **DB instance identifier:** Enter **dms-mysql**.
    - **Master username:** Leave as **admin**.
    - **Auto generate a password:** Leave unselected.
    - **Master password:** Enter **changeit**.
    - **Confirm password:** Enter **changeit** again.
  - **DB instance class:**
    - **DB instance class:** Leave **Standard classes** chosen.
    - For **DB instance class**, choose **db.m5.large**.
  - **Storage:**
    - Clear the **Enable storage autoscaling** box.
    - Leave the rest of the settings as they are.
  - **Connectivity:**
    - **Virtual private cloud:** **DMSVPC**
    - **Public access:** **Yes**. You must enable public access to use the AWS Schema Conversion Tool.
    - **Availability zone:** **us-west-2a**
    - Leave the rest of the settings as they are.
  - **Database authentication:** Leave **Password authentication** selected.
  - Expand **Additional configuration:**
    - Under **Database options**, enter **dms\_sample** for **Initial database name**.
    - Under **DB parameter group**, choose **dms-mysql-parameters**.
    - Under **Backup**, leave **Enable automatic backups** selected. Your source database must have automatic backups enabled to support ongoing replication.
    - Under **Backup retention period**, choose **1 day**.
    - Under **Encryption**, clear the **Enable encryption** box.
    - Under **Performance Insights**, clear the **Enable Performance Insights** box.
    - Under **Monitoring**, clear the **Enable Enhanced monitoring** box.
    - Under **Maintenance**, clear the **Enable auto minor version upgrade** box.
4. Choose **Create database**.

## Create your target Amazon RDS database

Repeat the previous procedure to create your target Amazon RDS database, with the following changes.

### To create your target RDS for PostgreSQL database

1. Repeat steps 1 and 2 from the previous procedure.

2. On the **Create database** page, set the same options, except for these:
  - a. For **Engine options**, choose **PostgreSQL**.
  - b. For **Version**, choose **PostgreSQL 13.4-R1**
  - c. For **DB instance identifier**, enter **dms-postgresql**.
  - d. For **Master username**, leave **postgres** selected.
  - e. For **DB parameter group**, choose **dms-postgresql-parameters**.
  - f. Clear **Enable automatic backups**.
3. Choose **Create database**.

## Create an Amazon EC2 client

In this section, you create an Amazon EC2 client. You use this client to populate your source database with data to replicate. You also use this client to verify replication by running queries on the target database.

Using an Amazon EC2 client to access your databases provides the following advantages over accessing your databases over the internet:

- You can restrict access to your databases to clients that are in the same VPC.
- We have confirmed that the tools you use in this tutorial work, and are easy to install, on Amazon Linux 2, which we recommend for this tutorial.
- Data operations between components in a VPC generally perform better than those over the internet.

### To create and configure an Amazon EC2 client to populate your source database

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the **Dashboard**, choose **Launch instance**.
3. On the **Launch an Instance** page, enter the following values:
  - a. In the **Name and tags** section, enter **DMSClient** for **Name**.
  - b. In the **Application and OS Images (Amazon Machine Image)** section, leave the settings as they are.
  - c. In the **Instance Type** section, choose **t2.xlarge**.
  - d. In the **Key pair (login)** section, choose **Create a new key pair**.
  - e. On the **Create key pair** page, enter the following:
    - **Key pair name:** **DMSKeyPair**
    - **Key pair type:** Leave as **RSA**.
    - **Private key file format:** Choose **pem** for OpenSSH on MacOS or Linux, or **ppk** for PuTTY on Windows.

Save the key file when prompted.

#### Note

You can also use an existing Amazon EC2 key pair rather than creating a new one.

- f. In the **Network Settings** section, choose **Edit**. Choose the following settings:
  - **VPC - required:** Choose the VPC with the ID that you recorded for the **DMSVPC** VPC.
  - **Subnet:** Choose the first public subnet.
  - **Auto-assign public IP:** Choose **Enable**.

Leave the rest of the settings as they are, and choose **Launch instance**.

## Populate your source database

In this section, you find endpoints for your source and target databases for later use and use the following tools to populate the source database:

- Git, to download the script that populates your source database.
- MySQL client, to run this script.

### Get endpoints

Find and note the endpoints of your RDS for PostgreSQL and RDS for PostgreSQL DB instances for later use.

#### To find your DB instance endpoints

1. Sign in to the AWS Management Console and open the Amazon RDS console at <https://console.aws.amazon.com/rds/>.
2. On the navigation pane, choose **Databases**.
3. Choose the **dms-mysql** database, and note the **Endpoint** value for the database.
4. Repeat the previous steps for the **dms-postgresql** database.

## Populate your source database

Next, connect to your client instance, install the necessary software, download AWS sample database scripts from Git, and run the scripts to populate your source database.

#### To populate your source database

1. Connect to the client instance using the host name and public key that you saved in previous steps.

For more information on connecting to an Amazon EC2 instance, see [Accessing Instances](#) in the [Amazon EC2 User Guide for Linux Instances](#).

#### Note

If you are using PuTTY, enable TCP keepalives on the **Connection** settings page so that your connection doesn't time out from inactivity.

2. Install Git and MySQL. Confirm installation as needed.

```
$ sudo yum install git  
$ sudo yum install mysql
```

3. Install PSQL. You use the PSQL client later to verify replication.

To do this, take the following steps:

- a. Create the Yum Package Repository Configuration file (pgdg.repo).

```
$ sudo su  
# cd /etc/yum.repos.d
```

```
# nano pgdg.repo
```

- b. Add the following contents to the pgdg.repo file.

```
[pgdg11]
name=PostgreSQL 11 $releasever - $basearch
baseurl=https://download.postgresql.org/pub/repos/yum/11/redhat/rhel-7.5-x86_64
enabled=1
gpgcheck=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-PGDG-11
```

- c. Install, and confirm installation as needed.

```
sed -i "s/rhel-\$releasever-\$basearch/rhel-7.5-x86_64/g" "/etc/yum.repos.d/pgdg.repo"
yum groupinstall "PostgreSQL Database Server 11 PGDG"
```

- d. Exit sudo.

```
$ exit
```

4. Run the following command to download the database creation scripts from GitHub.

```
git clone https://github.com/aws-samples/aws-database-migration-samples.git
```

5. Change to the aws-database-migration-samples/mysql/sampledbs/v1/ directory.
6. Run the following command. Provide the endpoint for your source RDS instance that you noted previously, for example dms-mysql.cdv5fbeyiy4e.us-west-2.rds.amazonaws.com.

```
mysql -h dms-mysql.abcdefghijkl01.us-west-2.rds.amazonaws.com -P 3306 -u admin -p
dms_sample < ~/aws-database-migration-samples/mysql/sampledbs/v1/install-rds.sql
```

7. Let the database creation script run. The script takes up to 45 minutes to create the schema and populate the data.

## Migrating your source schema to your target database using AWS SCT

In this section, you use the AWS Schema Conversion Tool to migrate your source schema to your target database. Alternatively, you can use DMS Schema Conversion to convert your source database schemas. For more information, see [Getting started with DMS Schema Conversion \(p. 56\)](#).

### To migrate your source schema to your target database with AWS SCT

1. Install the AWS Schema Conversion Tool. For more information, see [Installing, verifying, and updating the AWS SCT](#) in the *AWS Schema Conversion Tool User Guide*.

When you download JDBC drivers for MySQL and PostgreSQL, note where you save the drivers, in case the tool prompts you for their locations.

2. Open the AWS Schema Conversion Tool. Choose **File**, then choose **New project**.
3. In the **New project** window, set the following values:
  - Set **Project name** to **DMSProject**.
  - Keep **Location** as it is to store your AWS SCT project in the default folder.

- Choose **OK**.
4. Choose **Add source** to add a source MySQL database to your project, then choose **MySQL**, and choose **Next**.
  5. In the **Add source** page, set the following values:
    - **Connection name: source**
    - **Server name:** Enter the endpoint for the MySQL database that you noted previously.
    - **Server port: 3306**
    - **User name: admin**
    - **Password: changeit**
  6. Choose **Add target** to add a target Amazon RDS for PostgreSQL database to your project, then choose **Amazon RDS for PostgreSQL**. Choose **Next**.
  7. In the **Add target** page, set the following values:
    - **Connection name: target**
    - **Server name:** Enter the endpoint for the PostgreSQL database that you noted previously.
    - **Server port: 5432**
    - **Database:** Enter the name of your PostgreSQL database.
    - **User name: postgres**
    - **Password: changeit**
  8. In the left pane, choose **dms\_sample** under **Schemas**. In the right pane, choose your target Amazon RDS for PostgreSQL database. Choose **Create mapping**. You can add multiple mapping rules to a single AWS SCT project. For more information about mapping rules, see [Creating mapping rules](#).
  9. Choose **Main view**.
  10. In the left pane, choose **dms\_sample** under **Schemas**. Open the context (right-click) menu and choose **Convert schema**. Confirm the action.
- After the tool converts the schema, the **dms\_sample** schema appears in the right pane.
11. In the right pane, under **Schemas**, open the context (right-click) menu for **dms\_sample** and choose **Apply to database**. Confirm the action.

Verify that the schema migration completed. Perform the following steps.

### To check your schema migration

1. Connect to your Amazon EC2 client.
2. Start the PSQL client with the following command. Specify your PostgreSQL database endpoint, and provide the database password when prompted.

```
psql \
--host=dms-postgresql.abcdefg12345.us-west-2.rds.amazonaws.com \
--port=5432 \
--username=postgres \
--password \
--dbname=dms_sample
```

3. Query one of the (empty) tables to verify that AWS SCT applied the schema correctly,

```
dms_sample=> SELECT * from dms_sample.player;
 id | sport_team_id | last_name | first_name | full_name
-----+-----+-----+-----+
```

(0 rows)

# Setting up replication for AWS Database Migration Service

In this topic, you set up replication between the source and target databases.

## Step 1: Create a replication instance using the AWS DMS console

To start work with AWS DMS, create a replication instance.

A *replication instance* performs the actual data migration between source and target endpoints. Your instance needs enough storage and processing power to perform the tasks that migrate data from your source database to your target database. How large this replication instance should be depends on the amount of data to migrate and the tasks your instance needs to do. For more information about replication instances, see [Working with an AWS DMS replication instance \(p. 100\)](#).

## Create replication instance

### Replication instance configuration

#### Name

The name must be unique among all of your replication instances in the current AWS region.

Type a unique name for your replication instance

Replication instance name must not start with a numeric value

#### Descriptive Amazon Resource Name (ARN) - *optional*

A friendly name to override the default DMS ARN. You cannot modify it after creation.

Friendly-ARN-name

#### Description

Type a short description for your replication instance

The description must only have unicode letters, digits, whitespace, or one of these symbols: \_.:;/=+-@. 1000 maximum character.

#### Instance class [Info](#)

Choose an appropriate instance class for your replication needs. Each instance class provides differing levels of compute, network and memory capacity. [DMS pricing](#)

dms.t2.medium

2 vCPUs 4 GiB Memory



Include previous-generation instance classes

### To create a replication instance using the console

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. On the navigation pane, choose **Replication instances**, and then choose **Create replication instance**.
3. On the **Create replication instance** page, specify your replication instance configuration:
  - a. For **Name**, enter **DMS-instance**.
  - b. For **Description**, enter a short description for your replication instance (optional).
  - c. For **Instance class**, leave **dms.t3.medium** chosen.

The instance needs enough storage, networking, and processing power for your migration. For more information about how to choose an instance class, see [Choosing the right AWS DMS replication instance for your migration \(p. 103\)](#).

- d. For **Engine version**, accept the default.
- e. For **Multi AZ**, choose **Dev or test workload (Single-AZ)**.
- f. For **Allocated storage (GiB)**, accept the default of 50 GiB.

In AWS DMS, storage is mostly used by log files and cached transactions. For cache transactions, storage is used only when the cached transactions need to be written to disk. As a result, AWS DMS doesn't use a significant amount of storage.

- g. For **Network type** choose **IPv4**.
- h. For **VPC**, choose **DMSVPC**.
- i. For **Replication subnet group**, leave the replication subnet group currently chosen.
- j. Clear **Publicly accessible**.
4. Choose the **Advanced security and network configuration** tab to set values for network and encryption settings if you need them:
  - a. For **Availability zone**, choose **us-west-2a**.
  - b. For **VPC security group(s)**, choose the **Default** security group if it isn't already chosen.
  - c. For **AWS KMS key**, leave **(Default)** **aws/dms** chosen.
5. Leave the settings on the **Maintenance** tab as they are. The default is a 30-minute window selected at random from an 8-hour block of time for each AWS Region, occurring on a random day of the week.
6. Choose **Create**.

AWS DMS creates a replication instance to perform your migration.

## Step 2: Specify source and target endpoints

While your replication instance is being created, you can specify the source and target data store endpoints for the Amazon RDS databases you created previously. You create each endpoint separately.

DMS > Endpoints > Create endpoint

## Create endpoint

**Endpoint type** [Info](#)

**Source endpoint**  
A source endpoint allows AWS DMS to read data from a database (on-premises or in the cloud), or from other data source such as Amazon S3.

**Target endpoint**  
A target endpoint allows AWS DMS to write data to a database, or to other data source.

Select RDS DB instance

## Endpoint configuration

**Endpoint identifier** [Info](#)  
A label for the endpoint to help you identify it.  
*ProdEndpoint*

Descriptive Amazon Resource Name (ARN) - *optional*  
A friendly name for your endpoint that you can use to identify it in the AWS Management Console.

### To specify a source endpoint and database endpoint using the AWS DMS console

1. On the console, choose **Endpoints** from the navigation pane and then choose **Create Endpoint**.
2. On the **Create endpoint** page, choose the **Source** endpoint type. Select the **Select RDS DB instance** box, and choose the **dms-mysql** instance.
3. In the **Endpoint configuration** section, enter **dms-mysql-source** for **Endpoint identifier**.
4. For **Source engine**, leave **MySQL** chosen.
5. For **Access to endpoint database**, choose **Provide access information manually**. Verify that the **Port**, **Secure Socket Layer (SSL) mode**, **User name**, and **Password** are correct.
6. Choose the **Test endpoint connection (optional)** tab. For **VPC**, choose **DMSVPC**.
7. For **Replication instance**, leave **dms-instance** chosen.
8. Choose **Run test**.

After you choose **Run test**, AWS DMS creates the endpoint with the details that you provided and connects to it. If the connection fails, edit the endpoint definition and test the connection again. You can also delete the endpoint manually.

9. After you have a successful test, choose **Create endpoint**.
10. Specify a target database endpoint using the AWS DMS console. To do this, repeat the steps preceding, with the following settings:
  - **Endpoint type:** Target endpoint
  - **RDS Instance:** dms-postgresql
  - **Endpoint identifier:** dms-postgresql-target
  - **Target engine:** Leave PostgreSQL chosen.

When you're finished providing all information for your endpoints, AWS DMS creates your source and target endpoints for use during database migration.

## Step 3: Create a task and migrate data

In this step, you create a task to migrate data between the databases you created.

The screenshot shows the 'Create database migration task' configuration page. At the top, a breadcrumb navigation bar indicates: DMS > Database migration tasks > Create database migration task. The main title 'Create database migration task' is displayed. Below it, a section titled 'Task configuration' contains several input fields:

- Task identifier:** A text input field with placeholder text 'Type a unique identifier for the task'.
- Replication instance:** A dropdown menu with placeholder text 'Choose a replication instance'.
- Source database endpoint:** A dropdown menu with placeholder text 'Choose a source database endpoint'.
- Target database endpoint:** A dropdown menu with placeholder text 'Choose a target database endpoint'.
- Migration type:** A section with a blue 'Info' link and a dropdown menu containing 'Migrate existing data'.

#### To create a migration task and start your database migration

1. In the console navigation pane, choose **Database migration tasks**, and then choose **Create task**. The **Create database migration task** page opens.
2. In the **Task configuration** section, specify the following task options:
  - **Task identifier:** Enter **dms-task**.
  - **Replication instance:** Choose your replication instance (**dms-instance-vpc-<vpc id>**).
  - **Source database endpoint:** Choose **dms-mysql-source**.
  - **Target database endpoint:** Choose **dms-postgresql-target**.
  - **Migration type:** Choose **Migrate existing data and replicate on-going changes**.
3. Choose the **Task settings** tab. Set the following settings:
  - **Target table preparation mode:** **Do nothing**
  - **Stop task after full load completes:** **Don't stop**

4. Choose the **Table mappings** tab, and expand **Selection rules**. Choose **Add new selection rule**. Set the following settings:
  - **Schema:** Enter a schema
  - **Schema name:** dms\_sample
5. Choose the **Migration task startup configuration** tab, and then choose **Automatically on create**.
6. Choose **Create task**.

AWS DMS then creates the migration task and starts it. The initial database replication takes about 10 minutes. Make sure to do the next step in the tutorial before AWS DMS finishes migrating the data.

## Step 4: Test replication

In this section, you insert data into the source database during and after initial replication, and query the target database for the inserted data.

### To test replication

1. Make sure that your database migration task shows a status of **Running** but your initial database replication, started in the previous step, isn't complete.
2. Connect to your Amazon EC2 client, and start the MySQL client with the following command. Provide your MySQL database endpoint.

```
mysql -h dms-mysql.abcdefg12345.us-west-2.rds.amazonaws.com -P 3306 -u admin -pchangeit  
dms_sample
```

3. Run the following command to insert a record into the source database.

```
MySQL [dms_sample]> insert person (full_name, last_name, first_name) VALUES ('Test  
User1', 'User1', 'Test');  
Query OK, 1 row affected (0.00 sec)
```

4. Exit the MySQL client.

```
MySQL [dms_sample]> exit  
Bye
```

5. Before replication completes, query the target database for the new record.

From the Amazon EC2 instance, connect to the target database using the following command, providing your target database endpoint.

```
psql \  
--host=dms-postgresql.abcdefg12345.us-west-2.rds.amazonaws.com \  
--port=5432 \  
--username=postgres \  
--password \  
--dbname=dms_sample
```

Provide the password (**changeit**) when prompted.

6. Before replication completes, query the target database for the new record.

```
dms_sample=> select * from dms_sample.person where first_name = 'Test';  
 id | full_name | last_name | first_name  
----+-----+-----+-----
```

```
(0 rows)
```

7. While your migration task is running, you can monitor the progress of your database migration as it happens:

- In the DMS console navigation pane, choose **Database migration tasks**.
- Choose **dms-task**.
- Choose **Table statistics**.

For more information about monitoring, see [Monitoring AWS DMS tasks \(p. 558\)](#).

8. After replication completes, query the target database again for the new record. AWS DMS migrates the new record after initial replication completes.

```
dms_sample=> select * from dms_sample.person where first_name = 'Test';
   id      | full_name    | last_name | first_name
-----+-----+-----+-----+
 7077784 | Test User1  | User1    | Test
(1 row)
```

9. Exit the psql client.

```
dms_sample=> quit
```

10. Repeat step 1 to connect to the source database again.

11. Insert another record into the person table.

```
MySQL [dms_sample]> insert person (full_name, last_name, first_name) VALUES ('Test
User2', 'User2', 'Test');
Query OK, 1 row affected (0.00 sec)
```

12. Repeat steps 3 and 4 to disconnect from the source database and connect to the target database.

13. Query the target database for the replicated data again.

```
dms_sample=> select * from dms_sample.person where first_name = 'Test';
   id      | full_name    | last_name | first_name
-----+-----+-----+-----+
 7077784 | Test User1  | User1    | Test
 7077785 | Test User2  | User2    | Test
(2 rows)
```

## Step 5: Clean up AWS DMS resources

After you complete the getting started tutorial, you can delete the resources you created. You can use the AWS console to remove them. Make sure to delete the migration tasks before deleting the replication instance and endpoints.

### To delete a migration task using the console

1. On the AWS DMS console navigation pane, choose **Database migration tasks**.
2. Choose **dms-task**.
3. Choose **Actions, Delete**.

### To delete a replication instance using the console

1. On the AWS DMS console navigation pane, choose **Replication instances**.
2. Choose **DMS-instance**.
3. Choose **Actions, Delete**.

AWS DMS deletes the replication instance and removes it from the **Replication instances** page.

### To remove endpoints using the console

1. On the AWS DMS console navigation pane, choose **Endpoints**.
2. Choose **dms-mysql-source**.
3. Choose **Actions, Delete**.

After you delete your AWS DMS resources, make sure also to delete the following resources. For help with deleting resources in other services, see each service's documentation.

- Your RDS databases.
- Your RDS database parameter groups.
- Your RDS subnet groups.
- Any Amazon CloudWatch logs that were created along with your databases and replication instance.
- Security groups that were created for your Amazon VPC and Amazon EC2 client. Make sure to remove the inbound rule from **Default** for the **launch-wizard-1** security groups, which is necessary for you to be able to delete them.
- Your Amazon EC2 client.
- Your Amazon VPC.
- Your Amazon EC2 key pair for your Amazon EC2 client.

## Additional resources for working with AWS Database Migration Service

Later in this guide, you can learn how to use AWS DMS to migrate your data to and from the most widely used commercial and open-source databases.

We also recommend that you check the following resources as you prepare and perform a database migration project:

- [AWS DMS Step-by-Step Migration Guide](#) – This guide provides step-by-step walkthroughs that go through the process of migrating data to AWS.
- [AWS DMS API Reference](#) – This reference describes all the API operations for AWS Database Migration Service in detail.
- [AWS CLI for AWS DMS](#) – This reference provides information about using the AWS Command Line Interface (AWS CLI) with AWS DMS.

# Discover and evaluate databases for migration using AWS DMS Fleet Advisor

AWS DMS Fleet Advisor collects data from multiple database environments to provide insight into your data infrastructure. [DMS Fleet Advisor](#) collects data from your on-premises database and analytic servers from one or more central locations without the need to install it on every computer. Currently, DMS Fleet Advisor supports Microsoft SQL Server, MySQL, Oracle, and PostgreSQL database servers.

Based on data discovered from the network, you can build an inventory to define the list of database servers for further data collection. As details about these servers, databases, and schemas are collected, you can analyze the feasibility of your intended database migrations.

Use the following topics to better understand how to use AWS DMS Fleet Advisor.

## Topics

- [Supported AWS Regions \(p. 33\)](#)
- [Creating required AWS resources for AWS DMS Fleet Advisor \(p. 34\)](#)
- [Creating database users for AWS DMS Fleet Advisor \(p. 37\)](#)
- [Creating a data collector for AWS DMS Fleet Advisor \(p. 39\)](#)
- [Installing and configuring a data collector \(p. 41\)](#)
- [Discovering OS and database servers to monitor \(p. 43\)](#)
- [Managing monitored objects \(p. 45\)](#)
- [Using SSL with AWS DMS Fleet Advisor \(p. 46\)](#)
- [Collecting data for AWS DMS Fleet Advisor \(p. 47\)](#)
- [Using inventories for analysis in AWS DMS Fleet Advisor \(p. 48\)](#)
- [Troubleshooting for DMS data collector \(p. 50\)](#)

## Supported AWS Regions

You can use DMS Fleet Advisor in the following AWS Regions.

Region Name	Region
US East (N. Virginia)	us-east-1
US East (Ohio)	us-east-2
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Asia Pacific (Hong Kong)	ap-east-1

Region Name	Region
Asia Pacific (Tokyo)	ap-northeast-1
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Canada (Central)	ca-central-1
Europe (Frankfurt)	eu-central-1
Europe (Stockholm)	eu-north-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3
Canada (Central)	ca-central-1
South America (São Paulo)	sa-east-1

## Creating required AWS resources for AWS DMS Fleet Advisor

DMS Fleet Advisor needs a set of AWS resources in your account to forward and import inventory information, and to update the status of the DMS data collector.

Before you collect data and create inventories of databases and schemas for the first time, complete the following prerequisites.

### Create an Amazon S3 bucket

Create an Amazon S3 bucket where inventory metadata can be stored. We recommend that you preconfigure this S3 bucket before using DMS Fleet Advisor. Make sure that this S3 bucket is in the same AWS Region where you plan to run DMS Fleet Advisor. AWS DMS stores your DMS Fleet Advisor inventory metadata in this S3 bucket.

For more information about creating an S3 bucket, see [Create your first S3 bucket](#) in the *Amazon S3 User Guide*.

#### To create an Amazon S3 bucket to store local data environment information

1. Sign in to the AWS Management Console and open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. Choose **Create bucket**.
3. On the **Create bucket** page, give the bucket a globally unique name using your sign-in name, for example **fa-bucket-*yoursignin***.
4. Choose the AWS Region where you use the DMS Fleet Advisor.

5. Keep the remaining settings and choose **Create bucket**.

## Create IAM Resources

In this section, you create IAM resources for accessing Amazon S3 and DMS Fleet Advisor.

### To create an IAM policy for DMS Fleet Advisor and your data collector to access Amazon S3

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Policies**.
3. Choose **Create policy**.
4. In the **Create policy** page, choose the **JSON** tab.
5. Paste the following JSON into the editor, replacing the example code. Replace `<fa_bucket>` with the Amazon S3 bucket you created in the previous section.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:GetObject*",  
                "s3:GetBucket*",  
                "s3>List*",  
                "s3>DeleteObject*",  
                "s3:PutObject*"  
            ],  
            "Resource": [  
                "arn:aws:s3:::<fa_bucket>",  
                "arn:aws:s3:::<fa_bucket>/*"  
            ]  
        }  
    ]  
}
```

6. Choose **Next: Tags** and **Next: Review**.
7. Enter **FleetAdvisorS3Policy** for **Name\***. Choose **Create policy**.

### To create an IAM policy for DMS data collector to access DMS Fleet Advisor

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Policies**.
3. Choose **Create policy**.
4. In the **Create policy** page, choose the **JSON** tab.
5. Paste the following JSON code into the editor, replacing the example code.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dms:DescribeFleetAdvisorCollectors",  
                "dms:ModifyFleetAdvisorCollectorStatuses",  
            ]  
        }  
    ]  
}
```

```
        "dms:UploadFileMetadataList"
    ],
    "Resource": "*"
}
}
```

6. Choose **Next: Tags** and **Next: Review**.
7. Enter **DMSCollectorPolicy** for **Name\***. Choose **Create policy**.

#### To create an IAM role for DMS Fleet Advisor and your data collector to access Amazon S3

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**.
3. Choose **Create role**.
4. On the **Select trusted entity** page, for **Trusted entity type**, choose **AWS Service**. For **Use cases for other AWS services**, choose **DMS**. Check **DMS**. Choose **Next**.
5. On the **Add permissions** page, choose **FleetAdvisorS3Policy**. Choose **Next**.
6. On the **Name, review, and create** page, enter **FleetAdvisorS3Role** for **Role name**. Choose **Create role**.
7. On the **Roles** page, enter **FleetAdvisorS3Role** for **Role name**. Choose **FleetAdvisorS3Role**.
8. On the **FleetAdvisorS3Role** page, choose the **Trust relationships** tab. Choose **Edit trust policy**.
9. On the **Edit trust policy** page, paste the following JSON into the editor, replacing the existing text.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "dms.amazonaws.com",
          "dms-fleet-advisor.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

The preceding policy document grants the `sts:AssumeRole` permission to the services that AWS DMS uses to import collected data from the Amazon S3 bucket.

10. Choose **Update policy**.

#### To create an IAM user with minimum permissions to use DMS Fleet Advisor

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Users**.
3. Choose **Add users**.
4. On the **Add user** page, enter **FleetAdvisorUser** for **User name\***. Choose **Access key-Programmatic Access** for **Select AWS Access Type**. Choose **Next: Permissions**.

5. In the **Set permissions** section, choose **Attach existing policies directly**.
6. Use the search control to find and choose the **DMSCollectorPolicy** and **FleetAdvisorS3Policy** policies. Choose **Next: Tags**.
7. On the **Tags** page, choose **Next: Review**.
8. On the **Review** page, choose **Create user**. On the next page, choose **Download .csv** to save the new user credentials. Use these credentials with DMS Fleet Advisor for minimum required access permissions.

## Creating database users for AWS DMS Fleet Advisor

This section describes how to create users for your source databases with the minimum permissions required for the DMS data collector.

**This section contains the following topics:**

- [Using a database user with AWS DMS Fleet Advisor \(p. 37\)](#)
- [Creating a database user with MySQL \(p. 37\)](#)
- [Creating a database user with Oracle \(p. 38\)](#)
- [Creating a database user with PostgreSQL \(p. 38\)](#)
- [Creating a database user with Microsoft SQL Server \(p. 38\)](#)

## Using a database user with AWS DMS Fleet Advisor

You can use a database user other than `root` with the DMS data collector. Specify the user name and password after adding the database to the inventory, but before starting data collection. For more information about adding databases to the inventory, see [Managing monitored objects \(p. 45\)](#).

## Creating a database user with MySQL

To create a database user in a MySQL source database, use the following script. Replace the sample values with your values.

```
CREATE USER {your_user_name} identified BY '{your_password}';

GRANT REFERENCES ON *.* TO {your_user_name};
GRANT TRIGGER ON *.* TO {your_user_name};
GRANT EXECUTE ON *.* TO {your_user_name};

SELECT
    IF(round(Value1 + Value2 / 100 + Value3 / 10000, 4) > 5.0129, 'GRANT EVENT ON *.*'
    TO {your_user_name};', 'SELECT ''Events are not applicable'';') sql_statement
INTO @stringStatement
FROM (
    SELECT
        substring_index(ver, '.', 1) value1,
        substring_index(substring_index(ver, '.', 2), '.', - 1) value2,
        substring_index(ver, '.', - 1) value3
    FROM (
        SELECT
            IF(@@version regexp '[^0-9\.]+' ) != 0, @@innodb_version, @@version) AS ver
        FROM dual
    ) vercase
) v;
```

```
PREPARE sqlStatement FROM @stringStatement;
SET @stringStatement := NULL;
EXECUTE sqlStatement;
DEALLOCATE PREPARE sqlStatement;
```

## Creating a database user with Oracle

To create a database user in an Oracle source database, use the following script. Replace the sample values with your values. For `{your_user_name}`, enter the user name in uppercase.

To run this SQL script, connect to your Oracle database using SYSDBA privileges. After you run this SQL script, connect to your database using the credentials of the user that you created with this script. Also, use the credentials of this user to run the DMS data collector.

The following script adds the C## prefix to the name of the user for Oracle multitenant container databases (CDB).

```
CREATE USER {your_user_name} IDENTIFIED BY "{your_password}";
GRANT CREATE SESSION TO {your_user_name};
GRANT SELECT ANY DICTIONARY TO {your_user_name};
BEGIN
    DBMS_NETWORK_ACL_ADMIN.CREATE_ACL(
        acl => '{your_user_name}' || '_Connect_Access.xml',
        description => 'Connect Network',
        principal => '{your_user_name}',
        is_grant => TRUE,
        privilege => 'resolve',
        start_date => NULL,
        end_date => NULL);

    DBMS_NETWORK_ACL_ADMIN.ASSIGN_ACL(
        acl => '{your_user_name}' || '_Connect_Access.xml',
        host => '*',
        lower_port => NULL,
        upper_port => NULL);
END;
```

## Creating a database user with PostgreSQL

To create a database user in a PostgreSQL source database, use the following script. Replace the sample values with your values.

```
CREATE USER "{your_user_name}" WITH LOGIN PASSWORD '{your_password}';
GRANT pg_read_all_settings TO "{your_user_name}";
```

## Creating a database user with Microsoft SQL Server

To create a database user in a Microsoft SQL Server source database, use the following script. Replace the sample values with your values.

```
USE master
GO

CREATE LOGIN [{your_user_name}] WITH PASSWORD=N'{your_password}',
    DEFAULT_DATABASE=[master], DEFAULT_LANGUAGE=[us_english], CHECK_EXPIRATION=OFF,
    CHECK_POLICY=OFF -- Use for SQL login
```

```

GO

GRANT VIEW SERVER STATE TO [{your_user_name}]

GRANT VIEW ANY DEFINITION TO [{your_user_name}]

GRANT VIEW ANY DATABASE TO [{your_user_name}]

IF LEFT(CONVERT(SYSNAME, SERVERPROPERTY('ProductVersion')), CHARINDEX('.',
    CONVERT(SYSNAME, SERVERPROPERTY('ProductVersion')), 0)-1) >= 12
EXECUTE('GRANT CONNECT ANY DATABASE TO [{your_user_name}]')

DECLARE @dbname VARCHAR(100)
DECLARE @statement NVARCHAR(max)

DECLARE db_cursor CURSOR
LOCAL FAST_FORWARD
FOR
    SELECT
        name
    FROM MASTER.sys.databases
    WHERE state = 0
        AND is_read_only = 0
OPEN db_cursor
FETCH NEXT FROM db_cursor INTO @dbname
    WHILE @@FETCH_STATUS = 0
BEGIN

    SELECT @statement = 'USE '+ quotename(@dbname) +'';
    IF NOT EXISTS (SELECT * FROM sys.syslogins WHERE name = ''{your_user_name}'') OR NOT
    EXISTS (SELECT * FROM sys.sysusers WHERE name = ''{your_user_name}'')
        CREATE USER [{your_user_name}] FOR LOGIN [{your_user_name}];

    EXECUTE sp_addrolemember N'db_datareader', [{your_user_name}]

    BEGIN TRY
        EXECUTE sp_executesql @statement
    END TRY
    BEGIN CATCH
        DECLARE @err NVARCHAR(255)

        SET @err = error_message()

        PRINT @dbname
        PRINT @err
    END CATCH

    FETCH NEXT FROM db_cursor INTO @dbname
END
CLOSE db_cursor
DEALLOCATE db_cursor

```

## Creating a data collector for AWS DMS Fleet Advisor

This section describes how to create and download DMS data collector.

### To create and download a data collector

1. On the console, choose **Data collectors**. The **Data collectors** page opens.

2. Choose **Create data collector**. The **Create data collector** page opens.

DMS > Discover: Data collectors > Create data collector

## Create data collector Info

Create a data collector to identify servers, databases, and schemas on a network. After the data collector is created, you're prompted to register it by downloading and installing a local collector.

 You can create a maximum of 10 data collectors. [Learn more](#) 

### General configuration

#### Name

data-collector-name

Can have only Unicode letters, digits, white space, or one of the symbols in parentheses: (.:;/=+-@()). Maximum of 60 characters.

#### Description - optional

Provide a description of the data collector purpose, environment, or network to help you identify it in the future.

Can have only Unicode letters, digits, white space, or one of the symbols in parentheses: (.:;/=+-@()). Maximum of 255 characters.

### Connectivity Info

#### Amazon S3 bucket

Choose or create an Amazon S3 bucket to store collected metadata. Ensure this bucket is the currently selected region.

 s3://bucket





To create a bucket role, go to S3 

#### IAM role

Choose or create an IAM role that grants AWS DMS permissions to access the specified S3 bucket.

 Choose an option

To create an IAM role, go to IAM console 

Cancel

**Create data collector**

3. In the **General configuration** section, enter a value for your data collector for **Name** and for **Description**.

4. In the **Connectivity** section, choose **Browse S3**. Choose the Amazon S3 bucket that you preconfigured from the list that appears.

AWS DMS stores your DMS Fleet Advisor inventory metadata in this S3 bucket. Ensure that this S3 bucket is in the same AWS Region where your AWS DMS Fleet Advisor is currently running.
  5. In the list of IAM roles, choose the IAM role that you preconfigured from the list that appears. This role grants AWS DMS permissions to access the specified Amazon S3 bucket.
  6. Choose **Create data collector**. The **Data collectors** page opens and the created data collector appears in the list.
- When creating your first data collector, AWS DMS configures an environment in your Amazon S3 bucket that formats data and stores attributes for use with DMS Fleet Advisor.
7. Choose **Download local collector** on the information banner to download your newly created data collector. A message informs you that the download is in progress. After the download has finished, you can access the `AWS_DMS_Collector_Installer.msi` file.

You can now install the DMS data collector on your client.

## Installing and configuring a data collector

This section describes how to install your data collector, how to specify data forwarding credentials, and how to add an LDAP server to your project.

The following table describes hardware and software requirements for installing data collector.

Minimum	Recommended
<b>Processor:</b> 2 cores with CPU benchmark score greater than 8,000	<b>Processor:</b> 4 cores with CPU benchmark score greater than 11,000
<b>RAM:</b> 8 GB	<b>RAM:</b> 16 GB
<b>Hard drive size:</b> 256 GB	<b>Hard drive size:</b> 512 GB
<b>Operating system:</b> Microsoft Windows Server 2012 or higher	<b>Operating system:</b> Windows Server 2016 or higher

### To install a data collector on a client on your network

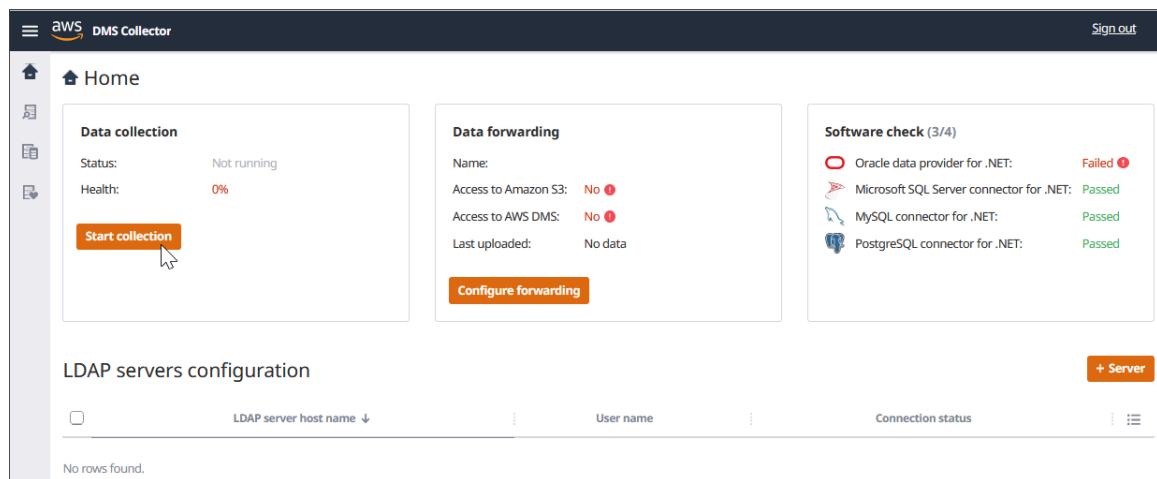
1. Run the .MSI installer. The **AWS DMS Fleet Advisor Collector Setup Wizard** page appears.
2. Choose **Next**. The **End-user license agreement** appears.
3. Read and accept the **End-user license agreement**.
4. Choose **Next**. The **Destination folder** page appears.
5. Choose **Next** to install the data collector in the default directory.

Or choose **Change** to enter another install directory. Then choose **Next**.
6. On the **Desktop shortcut** page, select the box to install an icon on your desktop.
7. Choose **Install**. The data collector is installed in the directory that you chose.
8. On the **Completed DMS Collector Setup Wizard** page, choose **Launch AWS DMS Collector**, and then choose **Finish**.

After you install the data collector, you can run it from a browser by entering **http://localhost:11000/** as the address. Optionally, from the Microsoft Windows **Start** menu, choose **AWS DMS Collector** in the list of programs. When you first run DMS Collector, you are asked to configure credentials. Create the user name and password for signing in to the data collector.

On the DMS data collector home page, you can find information for preparing and running data collection, including the following status conditions:

- Status and health of your data collection.
- Accessibility to your Amazon S3 bucket and to AWS DMS so the data collector can forward data to AWS DMS.
- Connectivity to your installed database drivers.
- Credentials of an LDAP server to perform initial discovery.



DMS data collector uses an LDAP directory to gather information about the machines and database servers in your network. *Lightweight Directory Access Protocol (LDAP)* is an open standard application protocol. It's used for accessing and maintaining distributed directory information services over an IP network. You can add an existing LDAP server to your project for data collector that you can use for discovering information about the infrastructure of your systems. To do so, choose the **+Server** option, then specify a fully-qualified domain name (FQDN) and the credentials for your domain controller. After adding the server, validate the connection check. If you want to get started with the discovery process, see the [Discovering OS and database servers to monitor \(p. 43\)](#) section.

## Configuring credentials for data forwarding

After you install the data collector, make sure that this application can send the collected data to AWS DMS Fleet Advisor.

### To configure credentials for data forwarding in AWS DMS Fleet Advisor

1. On the DMS data collector home page, in the **Data forwarding** section, choose **Configure credentials**. The **Credentials for data forwarding** dialog box opens.
2. Choose the AWS Region where you intend to use DMS Fleet Advisor.
3. Enter your **AWS Access Key ID** and your **AWS Secret Access Key** obtained earlier while creating an [Creating an IAM user \(p. 35\)](#).
4. Choose **Browse data collectors**.

If you haven't created a data collector in the specified Region yet, create a data collector before proceeding. For more information, see [??? \(p. 39\)](#).

5. In the **Choose data collector** window, select a data collector in the list and select **Choose**.
6. In the **Configure credentials for data forwarding** dialog box, choose **Save**.

On the **DMS Collector** home page, in the **Data forwarding** card, verify that both **Access to Amazon S3** and **Access to AWS DMS** are set to **Yes**.

## Discovering OS and database servers to monitor

You can use the DMS data collector to find and list all available servers in your network. Discovering all the available database servers in your network is recommended, but isn't required. Optionally, you can manually add or upload the list of servers for further data collection. For more information about manually adding a list of servers, see [Managing monitored objects \(p. 45\)](#).

We recommend that you discover all operating system (OS) servers before discovering databases on those servers. To discover OS servers, you need permission to run remote PowerShell, Secure Shell (SSH), and Windows Management Instrumentation (WMI) scripts and commands, as well as access to the Windows registry. To discover database servers in your network and collect data from them, you need read-only administrator permissions for a remote database connection. Make sure that you added an LDAP server before you proceed with discovery. For more information, see [Configuring credentials for data forwarding \(p. 42\)](#).

To get started with the DMS data collector, complete the following tasks:

- Discover all OS servers in your network.
- Add specific OS servers as objects to monitor.
- Verify connections for monitored OS servers.
- Discover Microsoft SQL Server, MySQL, Oracle, and PostgreSQL databases running on OS servers.
- Add database servers for data collection.
- Verify connections to the monitored databases.

### To discover OS servers in your network that you can monitor

1. In the DMS data collector navigation pane, choose **Discovery**. To display the navigation pane, choose the menu icon in the upper-left corner of the DMS data collector home page.

The **Discovery** page opens.

2. Make sure that the **OS servers** tab is selected, then choose **Run discovery**. The **Discovery parameters** dialog box appears.
3. Enter the LDAP servers that you want to use to scan your network.
4. Choose **Run discovery**. The page populates with a list of all OS servers discovered within your network, regardless of whether they're running a database.

We recommend that you run discovery for all OS servers before running discovery for databases on those servers. Your credentials make discovery possible first for the host servers, then for the databases that reside on them. You want to discover OS servers first before running discovery for databases on those servers. Be aware that the credentials that you use for an LDAP server to find OS servers in your network might differ from the credentials required to discover databases on a particular OS server. Therefore, we recommend that you add OS servers to monitored objects, verify the credentials and correct them if necessary, and then check connectivity before proceeding.

In the list of discovered OS servers in your network, you can now select the servers that you want to add to monitored objects.

#### To select OS servers as objects to monitor

1. On the **Discovery** page, choose the **OS servers** tab.
2. In the list of discovered OS servers shown, select the check box next to each server that you want to monitor.
3. Choose **Add to monitored objects**.

You can view the list of OS servers to monitor and verify connections on the **Monitor objects** page.

#### To verify connections of selected OS servers to monitor

1. In the DMS data collector navigation pane, choose **Monitored objects**.
2. On the **Monitored objects** page, choose the **All** tab. A list of discovered OS servers to be monitored appears.
3. Select the check box at the top of the column to choose all the listed OS servers.
4. Choose **Verify connection**. For each server object, view the results in the **Connections status** column.
5. Select servers with a connection status other than **Success**, and choose **Edit**. The **Edit monitored objects** dialog box opens.
6. Correct any improper information, then choose **Save**. The **Override credentials** dialog box opens.
7. Choose **Overwrite**. DMS data collector verifies and updates the status for each connection as **Success**.

You can now discover databases that reside on servers that you selected to monitor.

#### Discover databases running on servers

1. In the DMS data collector navigation pane, choose **Discovery**.
2. Choose the **Database servers** tab, and choose **Run discovery**. The **Discovery parameter** dialog box opens.
3. In the **Discovery parameter** dialog box, for **Discovered by**, choose **Monitored objects**. For **Servers**, choose the OS servers on which you want to run database discovery.
4. Choose **Run discovery**. The page becomes populated with a list of all databases that reside on the OS servers that you choose to monitor.

View information such as database address, server name, and database engine to help you select databases to monitor.

#### To select databases to monitor

1. On the **Discovery** page, choose the **Database servers** tab.
2. In the list of discovered databases shown, select the check box next to all the databases you want to monitor.
3. Choose **Add to monitored objects**.

You can now verify the connections to the databases you choose to monitor.

#### Verify connections to monitored databases

1. In the DMS data collector navigation pane, choose **Monitored objects**.

2. On the **Monitored objects** page, choose the **Database servers** tab. A list of discovered database servers you choose to monitor appears.
3. Select the check box at the top of the column to choose all the listed database servers.
4. Choose **Verify connection**. For each database, view the results in the **Connections status** column.
5. Select all connections that are undefined (blank) or resulted in **Failure**, then choose **Edit**. The **Edit monitored objects** dialog box opens.
6. Enter your **Login** and **Password** credentials, then choose **Save**. The **Change credentials** dialog box opens.
7. Choose **Overwrite**. DMS data collector verifies and updates the status for each connection as **Success**.

After discovering OS servers and databases to monitor, you can also perform actions to manage monitored objects.

## Managing monitored objects

You can select objects to monitor when you run the server discovery process as described in [Discovering OS and database servers \(p. 43\)](#). Also, you can manually manage objects such as operating system (OS) servers and database servers. You can perform the following actions to manage monitored objects:

- Add new objects to monitor
- Remove existing objects
- Edit existing objects
- Export and import a list of objects to monitor
- Check connections to objects
- Start data collection

For example, you can manually add an object to monitor.

### To add an object to monitor manually

1. On the **Monitored Objects** page, choose **+Server**. The **Add monitored object screen** opens.
2. Add information about the server, then choose **Save**.

You can also use a .csv file to import a large list of objects to monitor. Use the following .csv file format to import a list of objects to DMS data collector.

```
Hostname - Hostname or IP address of Monitored Object
Port - TCP port of Monitored Object
Engine: (one of the following)
    • Microsoft SQL Server
    • Microsoft Windows
    • Oracle Database
    • Linux
    • MySQL Server
    • PostgreSQL
Connection type: (one of the following)
    • Login/Password Authentication
    • Windows Authentication
```

- Key-Based Authentication
- Domain name:(Windows authentication)
- Use domain name for the account
- User name
- Password

### To import a .csv file with a list of objects to monitor

1. Choose **Import**. The **Import monitored objects** page opens.
2. Browse to the .csv file that you want to import, then choose **Next**.

You can view all of the objects and select those that you want to begin collecting data from.

## Using SSL with AWS DMS Fleet Advisor

To protect your data, AWS DMS Fleet Advisor can use SSL to access your databases.

### Supported Databases

AWS DMS Fleet Advisor supports using SSL to access following databases:

- Microsoft SQL Server
- MySQL
- PostgreSQL

### Setting up SSL

To use SSL to access your database, configure your database server to support SSL. For more information, see the following documentation for your database:

- SQL Server: [Enable encrypted connections to the Database Engine](#)
- MySQL: [Configuring MySQL to Use Encrypted Connections](#)
- PostgreSQL: [Secure TCP/IP Connections with SSL](#)

### Using SSL

To use SSL to connect to your database, select **Trust server certificate** and **Use SSL** when adding a server manually. For a MySQL database, you can use a custom certificate. To use a custom certificate, select the **Verify CA** check box. For information about adding a server, see [Managing monitored objects \(p. 45\)](#).

## Checking the Server Certificate Authority (CA) Certificate for SQL Server

If you want to validate your SQL Server's server CA certificate, clear **Trust server certificate** when adding the server. If your server uses a well-known CA, and the CA is installed by default on your OS, then verification should work normally. If DMS Fleet Advisor can't connect to your database server, install the CA certificate that your database server uses. For more information, see [Configure client](#).

# Collecting data for AWS DMS Fleet Advisor

You can start collecting data by choosing **Start collection** on the DMS data collector home page, or on the **Monitored objects** or **Data collection** page. DMS data collector can use up to four parallel threads to connect to a single database instance.

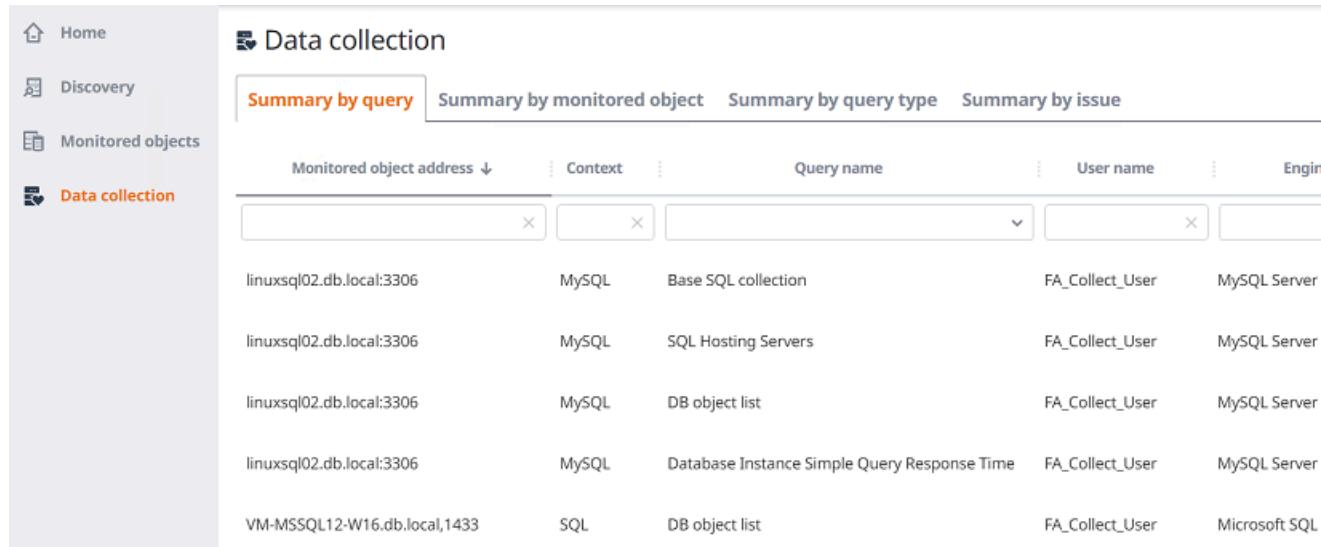
## Important

Before starting to collect data, view the **Software check** section on the DMS data collector home page. Verify that all database engines that you want to monitor have **Passed**. If some have **Failed**, and you have database servers with corresponding engines in your monitored objects list, fix the issue before proceeding. Helpful tips can be found next to the **Failed** state listed in the **Software check** section.

After data collection begins, you're redirected to the **Data collection** page, where collection queries are run and live progress is shown. Here, you can view overall collection health or on the DMS data collector home page. If overall data collection health is less than 100 percent, you might need to fix issues related to the collection.

On the **Data collection** page, you can see the collection status for each object. If something isn't working properly, a message appears showing how many issues occurred. To help determine a fix to an issue, you can check details. The following tabs list potential issues:

- **Summary by query** – Shows status of tests like the Ping test. You can filter results in the **Status** column. The **Status** column provides a message indicating how many failures occurred during data collection.
- **Summary by a monitored object** – Shows overall status per object.
- **Summary by query type** – Shows status for type of collector query, such as SQL, Secure Shell (SSH), or Windows Management Instrumentation (WMI) calls.
- **Summary by issue** – Shows all unique issues that occurred, with issue names and the number of times that each issue occurs.



The screenshot shows the AWS DMS Data collection page. On the left, there's a sidebar with links: Home, Discovery, Monitored objects, and Data collection (which is highlighted). The main area has a title 'Data collection' and a navigation bar with tabs: Summary by query (which is selected and highlighted in orange), Summary by monitored object, Summary by query type, and Summary by issue. Below the navigation bar is a table with the following columns: Monitored object address, Context, Query name, User name, and Engine. The table contains the following data:

Monitored object address	Context	Query name	User name	Engine
linuxsql02.db.local:3306	MySQL	Base SQL collection	FA_Collect_User	MySQL Server
linuxsql02.db.local:3306	MySQL	SQL Hosting Servers	FA_Collect_User	MySQL Server
linuxsql02.db.local:3306	MySQL	DB object list	FA_Collect_User	MySQL Server
linuxsql02.db.local:3306	MySQL	Database Instance Simple Query Response Time	FA_Collect_User	MySQL Server
VM-MSSQL12-W16.db.local,1433	SQL	DB object list	FA_Collect_User	Microsoft SQL

To export the collection results, choose **Export to CSV**.

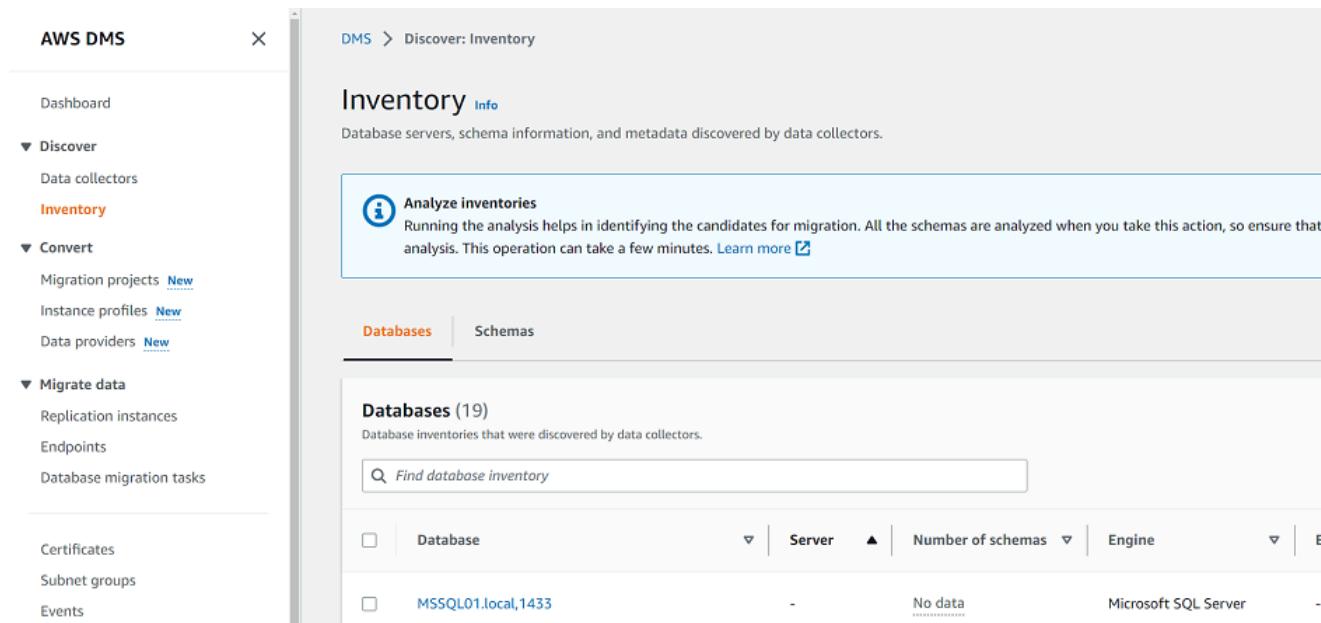
After identifying issues and resolving them, choose **Start collection** and rerun the data collection process. After performing data collection, the data collector uses secure connections to upload collected data to a DMS Fleet Advisor inventory. DMS Fleet Advisor stores information in your Amazon S3 bucket.

For information about configuring credentials for data forwarding, see [Configuring credentials for data forwarding \(p. 42\)](#).

# Using inventories for analysis in AWS DMS Fleet Advisor

To check the feasibility of potential database migrations, you can work with inventories of discovered databases and schemas. You can use the information in these inventories to understand which databases and schemas are good candidates for migration.

You can access database and schema inventories on the console. To do so, choose **Inventory** on the console.



## Using a database inventory for analysis

To view a list of all databases on all the discovered servers within your network from which data was collected, use the following procedure.

### To view a list of databases on your network servers that data was collected from

1. Choose **Inventory** on the console.

The **Inventory** page opens.

2. Choose the **Databases** tab.

A list of discovered databases appears.

Database	Server	Number of s...	Engine	Engine version	Engine ...
WinServ2016.d...	-	No data	PostgreSQL	-	-
VM-MSSQL14....	10.11.1.10	44	Microsoft SQL ...	⚠ 2014 (Extended support)	Enterprise
MSSQL01.dbm...	-	No data	Microsoft SQL ...	🟢 2019 (Mainstream support)	Express

- Choose **Analyze inventories** to determine schema properties, such as similarity and complexity. How much time the process takes depends on the number of objects to analyze, but it won't take more than one hour. Results from the analysis are found on the **Schemas** tab located on the **Inventory** page.

The entire inventory is analyzed together to identify duplicate schemas.

## Using a schema inventory for analysis

You can view a list of database schemas discovered on servers within your network from which the data was collected. Perform the following procedure.

### To view a list of schemas on your network servers that data was collected from

- Choose **Inventory** on the console. The **Inventory** page opens.
- Choose the **Schemas** tab. A list of schemas appears.
- Select a schema in the list to view information about it, including server, database, size, and complexity.

For each schema, you can view an object summary that provides information about object types, number of objects, object size, and lines of code.

- (Optional) Choose **Analyze inventories** to identify duplicate schemas. DMS Fleet Advisor analyzes database schemas to define the intersection of their objects. Schemas with intersections of more than 50 percent can be considered duplicates. Detailed analysis results are provided in the columns, **Similarity** and **Original schema**, found on the **Schemas** tab of the **Inventory** page.
- You can export inventory information to a .csv file for further review.

Analysis complete

DMS > Discover: Inventory

## Inventory Info

Database servers, schema information, and metadata discovered by data collectors.

**Analyze inventories**

Running the analysis helps in identifying the candidates for migration. All the schemas are analyzed when you take this action, so ensure analysis. This operation can take a few minutes. [Learn more](#)

**Schemas (13)**

Schema inventories that were discovered by data collectors.

Find schema inventory

Schema	Server	Database	Engine	Complexity
lsa_tests_src.lsa_tests_src	linuxsql02.db.local	linuxsql02.db.local:3306	MySQL Server	Simple
lsa_tests_src_90e_30a.lsa_t...	linuxsql02.db.local	linuxsql02.db.local:3306	MySQL Server	Simple
lsa_tests_src_50.lsa_tests_s...	linuxsql02.db.local	linuxsql02.db.local:3306	MySQL Server	Simple
lsa_tests_src_49.lsa_tests_s...	linuxsql02.db.local	linuxsql02.db.local:3306	MySQL Server	Simple

To identify schemas to migrate and determine the migration target, you can use AWS Schema Conversion Tool (AWS SCT). For more information, see [Using a new project wizard in AWS SCT](#).

After you have identified schemas to migrate, you can convert schemas using AWS SCT or DMS Schema Conversion. For more information about DMS Schema Conversion, see [Converting database schemas using DMS Schema Conversion \(p. 53\)](#).

## Troubleshooting for DMS data collector

In the following list, you can find actions to take when you encounter specific issues while collecting data with your data collector.

### Topics

- [Data collection issues related to network and server connections \(p. 51\)](#)
- [Data collection issues related to Windows Management Instrumentation \(p. 51\)](#)
- [Data collection issues related to Windows webpage composer \(p. 52\)](#)
- [Data collection issues related to SSL \(p. 52\)](#)

## Data collection issues related to network and server connections

### **NET: An exception occurred during a ping request.**

Check the name of the computer to see if it's in a state where it can't be resolved to an IP address.

For example, check if the computer is switched off, disconnected from the network, or decommissioned.

### **NET: Timed Out**

Turn on the inbound firewall rule "File and Printer Sharing (Echo Request - ICMPv4-In)". For example:

- \* Inbound ICMPv4

### **NET: DestinationHostUnreachable**

Check the IP address of the computer. In particular, check if it's on the same subnet as the computer running DMS data collector and whether it responds to Address Resolution Protocol (ARP) requests.

If the computer is on a different subnet, then the IP address of the gateway can't be resolved to the media access control (MAC) address.

Also, check if the computer is switched off, disconnected from the network, or decommissioned.

## Data collection issues related to Windows Management Instrumentation

### **WMI: The RPC server is unavailable. (Exception from HRESULT: 0x800706BA)**

Turn on the inbound firewall rule "Windows Management Instrumentation (DCOM-In)". For example:

- \* Inbound TCP/IP at local port 135.

Also, turn on the inbound firewall rule "Windows Management Instrumentation (WMI-In)". For example:

- \* Inbound TCP/IP at local port 49152 – 65535 for Windows Server 2008 and higher versions.
- \* Inbound TCP/IP at local port 1025 – 5000 for Windows Server 2003 and lower versions.

### **WMI: Access is denied. (Exception from HRESULT: 0x80070005 (E\_ACCESSDENIED))**

Try the following:

- Add the DMS data collector user to the Windows group, Distributed COM Users or Administrators.
- Start the Windows Management Instrumentation service and set its start-up type to Automatic.
- Make sure that your DMS data collector user name is in the \ format.

### **WMI: Access denied**

Add Remote Enable permission to the DMS data collector user on the root WMI namespace.

Use Advanced settings and make sure that permissions apply to "This namespace and subnamespaces."

**WMI: The call was canceled by the message filter. (Exception from HRESULT: 0x80010002...)**

Restart the Windows Management Instrumentation service.

## Data collection issues related to Windows webpage composer

**WPC: The network path was not found**

Turn on the inbound firewall rule "File and Printer Sharing (SMB-In)". For example:

\* Inbound TCP/IP at local port 445.

Also, start the Remote Registry service and set its start-up type to Automatic.

**WPC: Access is denied**

Add the DMS data collector user to the Performance Monitor Users or Administrators group.

**WPC: Category does not exist**

Run `loader /r` to rebuild the performance counter cache, then restart your computer.

### Note

For information about troubleshooting issues when migrating data using AWS Database Migration Service (AWS DMS), see [Troubleshooting migration tasks in AWS Database Migration Service \(p. 654\)](#)

## Data collection issues related to SSL

**SSL errors**

Your database requires a secure SSL connection, and you have not turned on the **Verify CA** and **Use SSL** options for the connection. Turn on these options and ensure that your local OS has the Certificate Authority installed that your database uses. For more information, see [Using SSL \(p. 46\)](#).

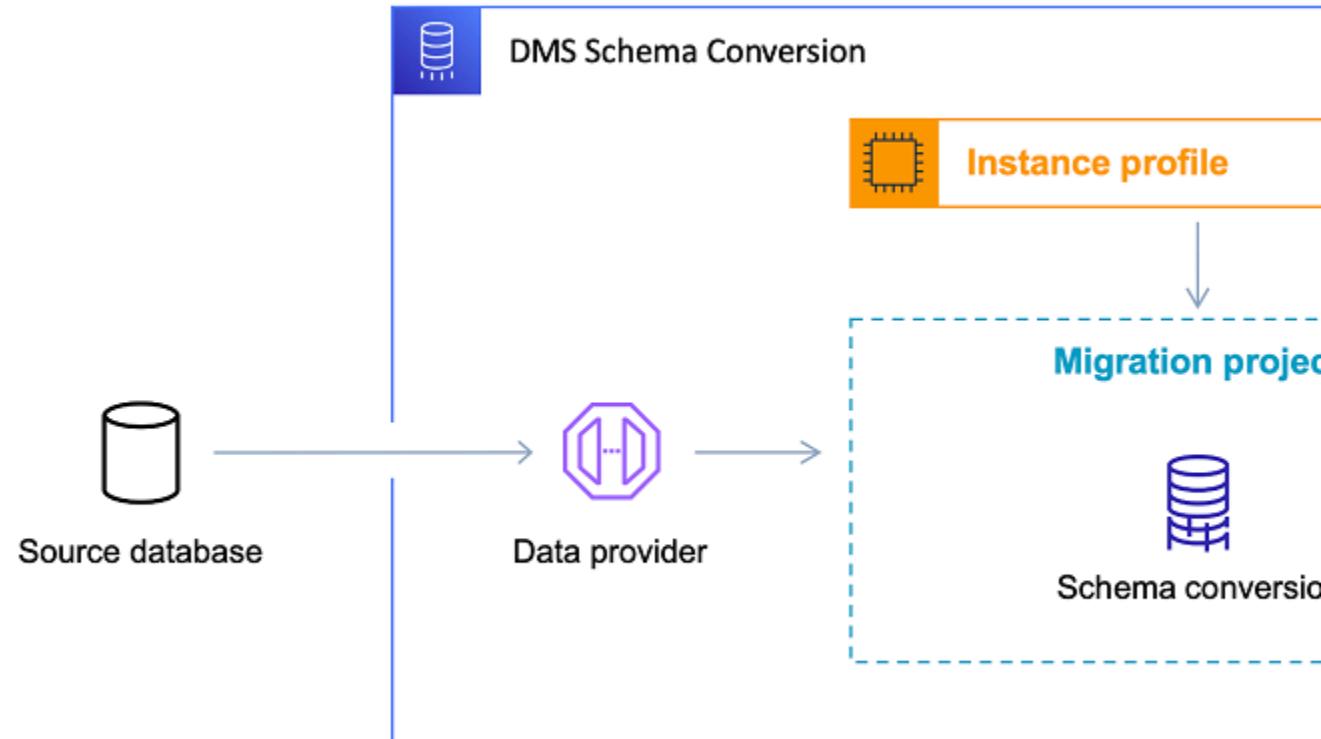
# Converting database schemas using DMS Schema Conversion

DMS Schema Conversion in AWS Database Migration Service (AWS DMS) makes database migrations between different types of databases more predictable. Use DMS Schema Conversion to assess the complexity of your migration for your source data provider, and to convert database schemas and code objects. You can then apply the converted code to your target database.

DMS Schema Conversion automatically converts your source database schemas and most of the database code objects to a format compatible with the target database. This conversion includes tables, views, stored procedures, functions, data types, synonyms, and so on. Any objects that DMS Schema Conversion can't convert automatically are clearly marked. To complete the migration, you can convert these objects manually.

At a high level, [DMS Schema Conversion](#) operates with the following three components: instance profiles, data providers, and migration projects. An *instance profile* specifies network and security settings. A *data provider* stores database connection credentials. A *migration project* contains data providers, an instance profile, and migration rules. AWS DMS uses data providers and an instance profile to design a process that converts database schemas and code objects.

The following diagram illustrates the DMS Schema Conversion process.



## Supported AWS Regions

You can create a DMS Schema Conversion migration project in the following AWS Regions. In other Regions, you can use the AWS Schema Conversion Tool. For more information about AWS SCT, see the [AWS Schema Conversion Tool User Guide](#).

Region Name	Region
US East (N. Virginia)	us-east-1
US East (Ohio)	us-east-2
US West (Oregon)	us-west-2
Asia Pacific (Tokyo)	ap-northeast-1
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Europe (Frankfurt)	eu-central-1
Europe (Stockholm)	eu-north-1
Europe (Ireland)	eu-west-1

## Supported source and target data providers

In DMS Schema Conversion, you create data providers to describe your source and target databases. Each data provider stores information about the data store type and the location of your database.

Your source data provider can be a self-managed engine running on-premises or on an Amazon Elastic Compute Cloud (Amazon EC2) instance.

DMS Schema Conversion supports the following data providers as sources for your migration projects:

- Microsoft SQL Server version 2008 R2 and higher
- Oracle version 10.2 and later, 11g and up to 12.2, 18c, and 19c

You can use DMS Schema Conversion to convert schemas for your source data provider to the target engine. That engine can run on an Amazon EC2 instance, an Amazon Relational Database Service (Amazon RDS) DB instance, or an Amazon Aurora DB instance.

DMS Schema Conversion supports the following data providers as targets for your migration projects:

- MySQL version 8.x
- PostgreSQL version 14.x

## Schema conversion features

DMS Schema Conversion provides the following features:

- DMS Schema Conversion automatically manages the AWS Cloud resources that are required for your database migration project. These resources include instance profiles, data providers, and AWS Secrets Manager secrets. They also include AWS Identity and Access Management (IAM) roles, Amazon S3 buckets, and migration projects.
- You can use DMS Schema Conversion to connect to your source database, read the metadata, and create database migration assessment reports. You can then save the report to an Amazon S3 bucket. With these reports, you get a summary of your schema conversion tasks and the details for items that DMS Schema Conversion can't automatically convert to your target database. Database migration assessment reports help evaluate how much of your migration project DMS Schema Conversion can automate. Also, these reports help estimate the amount of manual effort that is required to complete the conversion. For more information, see [Creating database migration assessment reports with DMS Schema Conversion \(p. 72\)](#).
- After you connect to your source and target data providers, DMS Schema Conversion can convert your existing source database schemas to the target database engine. You can choose any schema item from your source database to convert. After you convert your database code in DMS Schema Conversion, you can review your source code and the converted code. Also, you can save the converted SQL code to an Amazon S3 bucket.
- Before you convert your source database schemas, you can set up transformation rules. You can use transformation rules to change the data type of columns, move objects from one schema to another, and change the names of objects. You can apply transformation rules to databases, schemas, tables, and columns. For more information, see [Setting up transformation rules \(p. 76\)](#).
- You can change conversion settings to improve the performance of the converted code. These settings are specific for each conversion pair and depend on the features of the source database that you use in your code. For more information, see [Specifying schema conversion settings \(p. 79\)](#).
- In some cases, DMS Schema Conversion can't convert source database features to equivalent Amazon RDS features. For these cases, DMS Schema Conversion creates an extension pack in your target database to emulate the features that weren't converted. For more information, see [Using extension packs \(p. 83\)](#).
- You can apply the converted code and the extension pack schema to your target database. For more information, see [Applying your converted code \(p. 83\)](#).

## Schema conversion limitations

DMS Schema Conversion is a web-version of the AWS Schema Conversion Tool (AWS SCT). DMS Schema Conversion provides more limited functionality compared to the AWS SCT desktop application. For more information about AWS SCT, see the [AWS Schema Conversion Tool User Guide](#).

The following limitations apply when you use DMS Schema Conversion for database schema conversion:

- You can't save a migration project and use it in an offline mode.
- You can't edit SQL code for the source and target databases in a migration project for DMS Schema Conversion. To edit the SQL code of your source database, use your regular SQL editor. Choose **Refresh from database** to add the updated code in your migration project. To edit the converted code, save it as a SQL script. Then edit it in your code editor, and apply the updated code to your database manually.
- Migration rules in DMS Schema Conversion don't support changing column collation. Also, you can't use migration rules to move objects to a new schema.
- You can't apply filters to your source and target database trees to display only those database objects that meet the filter clause.
- DMS Schema Conversion extension pack doesn't include AWS Lambda functions that emulate email sending, job scheduling, and other features in your converted code.
- DMS Schema Conversion doesn't support the command line interface (CLI).

# Getting started with DMS Schema Conversion

To get started with DMS Schema Conversion, use the following tutorial. In it, you can learn to set up DMS Schema Conversion, create a migration project, and connect to your data providers. Then, you can learn to assess the complexity of your migration, and convert your source database to a format compatible with your target database. In addition, you can learn to apply the converted code to your target database.

The following tutorial covers the prerequisite tasks and demonstrates the conversion of an Amazon RDS for SQL Server database to Amazon RDS for MySQL. You can use any of the supported source and target data providers. For more information, see [Migration sources and targets \(p. 54\)](#).

For more information about DMS Schema Conversion, read the step-by-step migration walkthroughs for [Oracle to PostgreSQL](#) and [SQL Server to MySQL](#) migrations.

[This video](#) introduces the DMS Schema Conversion user interface and helps you get familiar with the core components of this service.

## Topics

- [Prerequisites for working with DMS Schema Conversion \(p. 56\)](#)
- [Step 1: Create an instance profile \(p. 59\)](#)
- [Step 2: Configure your data providers \(p. 59\)](#)
- [Step 3: Create a migration project \(p. 60\)](#)
- [Step 4: Create an assessment report \(p. 60\)](#)
- [Step 5: Convert your source code \(p. 61\)](#)
- [Step 6: Apply the converted code \(p. 61\)](#)
- [Step 7: Clean up and troubleshoot \(p. 61\)](#)

## Prerequisites for working with DMS Schema Conversion

To set up DMS Schema Conversion, complete the following tasks. Then you can set up an instance profile, add data providers, and create a migration project.

## Topics

- [Create a VPC based on Amazon VPC \(p. 56\)](#)
- [Create an Amazon S3 bucket \(p. 57\)](#)
- [Store database credentials in AWS Secrets Manager \(p. 57\)](#)
- [Create IAM roles \(p. 58\)](#)

## Create a VPC based on Amazon VPC

In this step, you create a virtual private cloud (VPC) in your AWS account. This VPC is based on the Amazon Virtual Private Cloud (Amazon VPC) service and contains your AWS resources.

### To create a VPC for DMS Schema Conversion

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.

2. Choose **Create VPC**.
3. On the **Create VPC** page, enter the following settings:
  - **Resources to create – VPC and more**
  - **Name tag auto-generation** – Choose **Auto-generate** and enter a globally unique name. For example, enter **sc-vpc**.
  - **IPv4 CIDR block – 10.0.1.0/24**
  - **NAT gateways – In 1 AZ**
  - **VPC endpoints – None**
4. Keep the rest of the settings as they are, and then choose **Create VPC**.
5. Choose **Subnets**, and take a note of your public and private subnet IDs.

To connect to your Amazon RDS databases, use public subnet IDs.

To connect to your on-premises databases, use private subnet IDs.

6. Choose **NAT gateways**. Choose your **NAT gateway** and take a note of your **Elastic IP address**.

Configure your network to make sure that AWS DMS can access your source on-premises database from this NAT gateway's public IP address. For more information, see [Using an internet connection to a VPC \(p. 64\)](#).

Use this VPC when you create your instance profile and target databases on Amazon RDS.

## Create an Amazon S3 bucket

To store information from your migration project, create an Amazon S3 bucket. DMS Schema Conversion uses this Amazon S3 bucket to save items such as assessment reports, converted SQL code, information about database schema objects, and so on.

### To create an Amazon S3 bucket for DMS Schema Conversion

1. Sign in to the AWS Management Console and open the Amazon S3 console at <https://console.aws.amazon.com/s3/>.
2. Choose **Create bucket**.
3. On the **Create bucket** page, select a globally unique name for your S3 bucket. For example, enter **sc-s3-bucket**.
4. For **AWS Region**, choose your Region.
5. For **Bucket Versioning**, choose **Enable**.
6. Keep the rest of the settings as they are, and then choose **Create bucket**.

## Store database credentials in AWS Secrets Manager

Store your source and target database credentials in AWS Secrets Manager. Make sure that you replicate these secrets to your AWS Region. DMS Schema Conversion uses these secrets to connect to your databases in the migration project.

### To store your database credentials in AWS Secrets Manager

1. Sign in to the AWS Management Console and open the AWS Secrets Manager console at <https://console.aws.amazon.com/secretsmanager/>.
2. Choose **Store a new secret**.

3. The **Choose secret type** page opens. For **Secret type**, choose the type of database credentials to store:
  - **Credentials for Amazon RDS database** – Choose this option to store credentials for your Amazon RDS database. For **Credentials**, enter the credentials for your database. For **Database**, choose your database.
  - **Credentials for other database** – Choose this option to store credentials for your source Oracle or SQL Server databases. For **Credentials**, enter the credentials for your database.
  - **Other type of secret** – Choose this option to store only the user name and password to connect to your database. Choose **Add row** to add two key-value pairs. Make sure that you use **username** and **password** for key names. For values related to these keys, enter the credentials for your database.
4. For **Encryption key**, choose the AWS KMS key that Secrets Manager uses to encrypt the secret value. Choose **Next**.
5. On the **Configure secret** page, enter a descriptive **Secret name**. For example, enter **sc-source-secret** or **sc-target-secret**.
6. Choose **Replicate secret** and then for **AWS Region** choose your Region. Choose **Next**.
7. On the **Configure rotation** page, choose **Next**.
8. On the **Review** page, review your secret details, and then choose **Store**.

To store credentials for your source and target databases, repeat these steps.

## Create IAM roles

Create AWS Identity and Access Management (IAM) roles to use in your migration project. DMS Schema Conversion uses these IAM roles to access your Amazon S3 bucket and database credentials stored in AWS Secrets Manager.

### To create an IAM role that provides access to your Amazon S3 bucket

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**.
3. Choose **Create role**.
4. On the **Select trusted entity** page, choose **AWS service**. Choose **DMS**.
5. Choose **Next**. The **Add permissions** page opens.
6. For **Filter policies**, enter **S3**. Choose **AmazonS3FullAccess**.
7. Choose **Next**. The **Name, review, and create** page opens.
8. For **Role name**, enter a descriptive name. For example, enter **sc-s3-role**. Choose **Create role**.

### To create an IAM role that provides access to AWS Secrets Manager

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**.
3. Choose **Create role**.
4. On the **Select trusted entity** page, choose **AWS service**. Choose **DMS**.
5. Choose **Next**. The **Add permissions** page opens.
6. For **Filter policies**, enter **Secret**. Choose **SecretsManagerReadWrite**.
7. Choose **Next**. The **Name, review, and create** page opens.

8. For **Role name**, enter a descriptive name. For example, enter **sc-secrets-manager-role**. Choose **Create role**.

## Step 1: Create an instance profile

You can create an instance profile as described in the following procedure. In this instance profile, you specify network and security settings for your DMS Schema Conversion project.

### To create an instance profile

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. In the navigation pane, choose **Instance profiles**, and then choose **Create instance profile**.
3. For **Name**, enter a unique name for your instance profile. For example, enter **sc-instance**.
4. For **Network type**, choose **IPv4** to create an instance profile that supports only IPv4 addressing. To create an instance profile that supports IPv4 and IPv6 addressing, choose **Dual-stack mode**.
5. For **Virtual private cloud (VPC)**, choose the VPC that you created in the prerequisites step.
6. For **Subnet IDs**, choose subnet IDs for your instance profile. To connect to your Amazon RDS databases, use public subnet IDs. To connect to your on-premises databases, use private subnet IDs. Make sure that you choose at least two subnet IDs.
7. For **S3 bucket** under **Schema conversion settings - optional**, choose an Amazon S3 bucket that you created in the prerequisites step.
8. For **IAM role**, choose the AWS Identity and Access Management (IAM) role that grants access to Amazon S3. You created this role in the prerequisites step.
9. Choose **Create instance profile**.

To create a migration project, use this instance profile.

## Step 2: Configure your data providers

Next, you create data providers that describe your source and target databases. For each data provider, you specify a data store type and location information. You don't store your database credentials in a data provider.

### To create a data provider for a source on-premises database

1. Sign in to the AWS Management Console, and open the AWS DMS console.
2. In the navigation pane, choose **Data providers**, and then choose **Create data provider**.
3. For **Configuration**, choose **Enter manually**.
4. For **Name**, enter a unique name for your source data provider. For example, enter **sc-source**.
5. For **Engine type**, choose the type of database engine for your data provider.
6. Provide your connection information for the source database. The connection parameters depend on your source database engine. For more information, see [Data providers \(p. 64\)](#).
7. For **Secure Socket Layer (SSL) mode**, choose the type of SSL enforcement.
8. Choose **Create data provider**.

### To create a data provider for a target Amazon RDS database

1. Sign in to the AWS Management Console and open the AWS DMS console.
2. In the navigation pane, choose **Data providers**, and then choose **Create data provider**.

3. For **Configuration**, choose **RDS database instance**.
4. For **Database from RDS**, choose **Browse**, and choose your database. DMS Schema Conversion automatically retrieves the information about the engine type, server name, and port.
5. For **Name**, enter a unique name for your target data provider. For example, enter **sc-target**.
6. For **Database name**, enter the name of your database.
7. For **Secure Socket Layer (SSL) mode**, choose the type of SSL enforcement.
8. Choose **Create data provider**.

## Step 3: Create a migration project

Now you can create a migration project. In the migration project, you specify your source and target data providers, and your instance profile.

### To create a migration project

1. Choose **Migration projects**, and then choose **Create migration project**.
2. For **Name**, enter a unique name for your migration project. For example, enter **sc-project**.
3. For **Instance profile**, choose **sc-instance**.
4. For **Source**, choose **Browse**, and then choose **sc-source**.
5. For **Secret ID**, choose **sc-source-secret**.
6. For **IAM role**, choose **sc-secrets-manager-role**.
7. For **Target**, choose **Browse**, and then choose **sc-target**.
8. For **Secret ID**, choose **sc-target-secret**.
9. For **IAM role**, choose **sc-secrets-manager-role**.
10. Choose **Create migration project**.

## Step 4: Create an assessment report

To assess the complexity of the migration, create the database migration assessment report. This report includes the list of all database objects that DMS Schema Conversion can't convert automatically.

### To create an assessment report

1. Choose **Migration projects**, and then choose **sc-project**.
2. Choose **Schema conversion**, and then choose **Launch schema conversion**.
3. In the source database pane, choose the database schema to assess. Also, select the check box for the name of this schema.
4. In the source database pane, choose **Assess** in the **Actions** menu. The **Assess** dialog box appears.
5. Choose **Assess** in the dialog box to confirm your choice.

The **Summary** tab shows the number of items that DMS Schema Conversion can automatically convert for database storage objects and database code objects.

6. Choose **Action items** to see the list of all database objects that DMS Schema Conversion can't convert automatically. Review the recommended actions for each item.
7. To save a copy of your assessment report, choose **Export results**. Next, choose one of the following formats: **CSV** or **PDF**. The **Export** dialog box appears.
8. Choose **Export** to confirm your choice.
9. Choose **S3 bucket**. The Amazon S3 console opens.
10. Choose **Download** to save your assessment report.

## Step 5: Convert your source code

You can convert your source database schema using the following procedure. Then you can save the converted code as SQL scripts in a text file.

### To convert your database schema

1. In the source database pane, choose the database schema to convert. Also, select the check box for the name of this schema.
2. In the source database pane, choose **Convert** in the **Actions** menu. The **Convert** dialog box appears.
3. Choose **Convert** in the dialog box to confirm your choice.
4. Choose a database object in the source database pane. DMS Schema Conversion displays the source code and the converted code for this object.
5. In the target database pane, choose the converted database schema. Also, select the check box for the name of this schema.
6. For **Actions**, choose **Save as SQL**. The **Save** dialog box appears.
7. Choose **Save as SQL** to confirm your choice.
8. Choose **S3 bucket**. The Amazon S3 console opens.
9. Choose **Download** to save your SQL scripts.

## Step 6: Apply the converted code

DMS Schema Conversion doesn't immediately apply the converted code to your target database. To update your target database, you can use the SQL scripts that you created in the previous step. Alternatively, use the following procedure to apply the converted code from DMS Schema Conversion.

### To apply the converted code

1. In the target database pane, choose the converted database schema. Also, select the check box for the name of this schema.
2. For **Actions**, choose **Apply changes**. The **Apply changes** dialog box appears.
3. Choose **Apply** to confirm your choice.

## Step 7: Clean up and troubleshoot

You can use Amazon CloudWatch to review or share your DMS Schema Conversion logs.

### To review DMS Schema Conversion logs

1. Sign in to the AWS Management Console and open the CloudWatch console at <https://console.aws.amazon.com/cloudwatch/>.
2. Choose **Logs, Log groups**.

The name of your DMS Schema Conversion log group starts with `dms-tasks-sct`. You can sort the log groups by **Creation time** to find DMS Schema Conversion log group.

Also, the name of your log group includes the Amazon Resource Name (ARN) of your migration project. You can see the ARN of your project on the **Migration projects** page in DMS Schema Conversion. Make sure that you choose **ARN** in **Preferences**.

3. Choose the name of your log group, and then choose the name of your log stream.
4. For **Actions**, choose **Export results** to save your DMS Schema Conversion log.

After you've finished your schema conversion in DMS Schema Conversion, clean up your resources.

#### To clean up your DMS Schema Conversion resources

1. Sign in to the AWS Management Console and open the AWS DMS console.
2. In the navigation pane, choose **Migration projects**.
  - a. Choose **sc-project**.
  - b. Choose **Schema conversion**, and then choose **Close schema conversion**.
  - c. Choose **Delete** and confirm your choice.
3. In the navigation pane, choose **Instance profiles**.
  - a. Choose **sc-instance**.
  - b. Choose **Delete** and confirm your choice.
4. In the navigation pane, choose **Data providers**.
  - a. Select **sc-source** and **sc-target**.
  - b. Choose **Delete** and confirm your choice.

Also, make sure that you clean up other AWS resources that you created, such as your Amazon S3 bucket, database secrets in AWS Secrets Manager, IAM roles, and virtual private cloud (VPC).

## Creating instance profiles for DMS Schema Conversion

Before you create your migration project in DMS Schema Conversion, you set up an instance profile. An *instance profile* specifies network and security settings for your DMS Schema Conversion instances that you use for database schema conversion.

You can create multiple instance profiles. Select an instance profile to use for each migration project that you create for DMS Schema Conversion.

You can create up to 60 instance profiles in DMS Schema Conversion.

#### To create an instance profile

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. In the navigation pane, choose **Instance profiles**.
3. Choose **Create instance profile**.
4. On the **Create instance profile** page, enter a descriptive value for **Name** for your instance profile.
5. For **Network type**, choose **Dual-stack mode** to create an instance profile that supports IPv4 and IPv6 addressing. Keep the default option to create an instance profile that supports only IPv4 addressing.
6. Next, choose **Virtual private cloud (VPC)** to run your instance of the selected network type. Then choose values for **Subnet IDs** and **VPC security groups** for your instance profile.

To connect to Amazon RDS databases, use public subnet IDs. To connect to on-premises databases, use private subnet IDs. Make sure that you configured your network so that AWS DMS can access your source on-premises database using the NAT gateway's public IP address. For more information, see [Create a VPC based on Amazon VPC \(p. 56\)](#).

7. For **Schema conversion settings - optional**, choose an Amazon S3 bucket to store information from your migration project. Then choose the AWS Identity and Access Management (IAM) role that provides access to this Amazon S3 bucket. For more information, see [Create an Amazon S3 bucket \(p. 57\)](#).
8. Choose **Create instance profile**.

After you create your instance profile, you can modify or delete it.

#### To modify an instance profile

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Instance profiles**. The **Instance profiles** page opens.
3. Choose your instance profile, and then choose **Modify**.
4. Update the name of your instance profile, edit the VPC or Amazon S3 bucket settings.
5. Choose **Save changes**.

#### To delete an instance profile

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Instance profiles**. The **Instance profiles** page opens.
3. Choose your instance profile, and then choose **Delete**.
4. Choose **Delete** to confirm your choice.

## Setting up a network for DMS Schema Conversion

DMS Schema Conversion creates a schema conversion instance in a virtual private cloud (VPC) based on the Amazon VPC service. When you create your instance profile, you specify the VPC to use. You can use your default VPC for your account and AWS Region, or you can create a new VPC.

You can use different network configurations to set up interaction for your source and target databases with DMS Schema Conversion. These configurations depend on the location of your source data provider and your network settings. The following topics provide descriptions of common network configurations.

### Using a single VPC for source and target data providers

The simplest network configuration for DMS Schema Conversion is a single VPC configuration. Here, your source data provider, instance profile, and the target data provider are all located in the same VPC. You can use this configuration to convert your source database on an Amazon EC2 instance.

To use this configuration, make sure that the VPC security group used by the instance profile has access to the data providers. For example, you can allow either a VPC Classless Inter-Domain Routing (CIDR) range or the Elastic IP address for your Network Address Translation (NAT) gateway.

### Using multiple VPCs for source and target data providers

If your source and target data providers are in different VPCs, you can create your instance profile in one of the VPCs. You can then link these two VPCs by using VPC peering. You can use this configuration to convert your source database on an Amazon EC2 instance.

A *VPC peering connection* is a networking connection between two VPCs that activates routing using the private IP address of each VPC\ as if they were in the same network. You can create a VPC peering

connection between your own VPCs, with a VPC in another AWS account, or with a VPC in a different AWS Region. For more information about VPC peering, see [VPC peering](#) in the *Amazon VPC User Guide*.

To implement VPC peering, follow the instructions in [Work with VPC peering connections](#) in the *Amazon VPC User Guide*. Make sure that the route table of one VPC contains the CIDR block of the other. For example, suppose that VPC A is using destination 10.0.0.0/16 and VPC B is using destination 172.31.0.0. In this case, the route table of VPC A should contain 172.31.0.0, and the route table of VPC B must contain 10.0.0.0/16. For more detailed information, see [Update your route tables for VPC peering connection](#) in the *Amazon VPC Peering Guide*.

## Using AWS Direct Connect or a VPN to configure a network to a VPC

Remote networks can connect to a VPC using several options, such as AWS Direct Connect or a software or hardware VPN connection. You can use these options to integrate existing on-site services by extending an internal network into the AWS Cloud. You might integrate on-site services such as monitoring, authentication, security, data, or other systems. By using this type of network extension, you can seamlessly connect on-site services to resources hosted by AWS, such as a VPC. You can use this configuration to convert your source on-premises database.

In this configuration, the VPC security group must include a routing rule that sends traffic destined for a VPC CIDR range or specific IP address to a host. This host must be able to bridge traffic from the VPC into the on-premises VPN. In this case, the NAT host includes its own security group settings. These settings must allow traffic from your VPC CIDR range or security group into the NAT instance. For more information, see [Create a Site-to-Site VPN connection](#) in the *AWS Site-to-Site VPN User Guide*.

## Using an internet connection to a VPC

If you don't use a VPN or AWS Direct Connect to connect to AWS resources, you can use an internet connection. This configuration involves a private subnet in a VPC with an internet gateway. The gateway contains the target data provider and the instance profile. You can use this configuration to convert your source on-premises database.

To add an internet gateway to your VPC, see [Attaching an internet gateway](#) in the *Amazon VPC User Guide*.

The VPC route table must include routing rules that send traffic *not* destined for the VPC by default to the internet gateway. In this configuration, the connection to the data provider appears to come from the public IP address of your NAT gateway. For more information, see [VPC Route Tables](#) in the *Amazon VPC User Guide*.

# Adding data providers for DMS Schema Conversion

Before you create your migration project in DMS Schema Conversion, you first add data providers. A *data provider* stores a data store type and the location information about your database.

After you add a data provider to your migration project, you provide the database credentials from AWS Secrets Manager. DMS Schema Conversion uses this information to connect to a database. Then DMS Schema Conversion converts schemas of your source data provider to a format that is compatible with your target data provider. After you apply the converted code to your new target database, you can migrate the data using AWS DMS.

### Topics

- [Creating data providers in DMS Schema Conversion \(p. 65\)](#)
- [Creating source data providers in DMS Schema Conversion \(p. 66\)](#)

- [Creating target data providers in DMS Schema Conversion \(p. 68\)](#)

## Creating data providers in DMS Schema Conversion

You can create data providers and use them in migration projects for DMS Schema Conversion. Your data provider can be a self-managed engine running on-premises or on an Amazon EC2 instance. Also, your data provider can be a fully managed engine, such as Amazon Relational Database Service (Amazon RDS) or Amazon Aurora.

For each database, you can create a single data provider. You can use a single data provider in multiple migration projects.

You can create up to 1,000 data providers in DMS Schema Conversion.

Before creating a migration project, make sure that you have created at least two data providers. One of your data providers must be on an AWS service. You can't use DMS Schema Conversion to migrate your database to an on-premises database.

The following procedure shows you how to create data providers in the AWS DMS console wizard.

### To create a data provider

1. Sign in to the AWS Management Console, then open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Data providers**. The **Data providers** page opens.
3. Choose **Create data provider**. The following table describes the settings.

Option	Action
<b>Configuration</b>	Choose whether to enter the information about your data provider manually or to use the Amazon RDS DB instance.
<b>Name</b>	Enter a name for your data provider. Make sure that you use a unique name for your data provider so that you can easily identify it.
<b>Engine type</b>	Choose the type of the database engine for your data provider.
<b>Server name</b>	Enter the Domain Name Service (DNS) name or IP address of your database server.
<b>Port</b>	Enter the port used to connect to your database server.
<b>Service ID (SID) or service name</b>	Enter the Oracle System ID (SID). To find the Oracle SID, submit the following query to your Oracle database:  <pre>SELECT sys_context('userenv','instance_name') AS SID FROM dual;</pre>
<b>Database name</b>	Enter the name of the database for this data provider.
<b>Secure Socket Layer (SSL) mode</b>	Choose an SSL mode if you want to turn on connection encryption for this data provider. Depending on the mode that you select, you might need to provide certificate and server certificate information. For further details, see <a href="#">Using SSL with AWS Database Migration Service</a> .

4. Choose **Create data provider**.

After you create a data provider, make sure that you add database connection credentials in AWS Secrets Manager.

## Creating source data providers in DMS Schema Conversion

You can use a Microsoft SQL Server or Oracle database as a source data provider in migration projects for DMS Schema Conversion. Your source data provider can be a self-managed engine running on-premises or on an Amazon EC2 instance.

Make sure that you configure the network to permit interaction between your source data provider and DMS Schema Conversion. For more information, see [Setting up a network for DMS Schema Conversion \(p. 63\)](#).

### Topics

- [Using a Microsoft SQL Server database as a source in DMS Schema Conversion \(p. 66\)](#)
- [Using an Oracle database as a source in DMS Schema Conversion \(p. 67\)](#)

## Using a Microsoft SQL Server database as a source in DMS Schema Conversion

DMS Schema Conversion supports the following versions of on-premises SQL Server databases: 2008R2, 2012, 2014, 2016, 2017, and 2019. Also, you can connect to the following versions of RDS for SQL Server databases: 2012, 2014, 2016, 2017, and 2019.

You can use DMS Schema Conversion to convert database code objects from SQL Server to the following targets:

- Aurora MySQL
- Aurora PostgreSQL
- RDS for MySQL
- RDS for PostgreSQL

For more information about using DMS Schema Conversion with a source SQL Server database, see the [SQL Server to MySQL migration step-by-step walkthrough](#).

### Privileges for Microsoft SQL Server as a source

View the following list of privileges required for Microsoft SQL Server as a source:

- VIEW DEFINITION
- VIEW DATABASE STATE

The VIEW DEFINITION privilege enables users that have public access to see object definitions. DMS Schema Conversion uses the VIEW DATABASE STATE privilege to check the features of the SQL Server Enterprise edition.

Repeat the grant for each database whose schema you are converting.

In addition, grant the following privileges on the master database:

- VIEW SERVER STATE
- VIEW ANY DEFINITION

DMS Schema Conversion uses the VIEW SERVER STATE privilege to collect server settings and configuration. Make sure that you grant the VIEW ANY DEFINITION privilege to view data providers.

To read information about Microsoft Analysis Services, run the following command on the master database.

```
EXEC master..sp_addsrvrolemember @loginame = N'<user_name>', @rolename = N'sysadmin'
```

In the preceding example, replace the `<user_name>` placeholder with the name of the user who you previously granted with the required privileges.

To read information about SQL Server Agent, add your user to the SQLAgentUser role. Run the following command on the msdb database.

```
EXEC sp_addrolemember <SQLAgentRole>, <user_name>;
```

In the preceding example, replace the `<SQLAgentRole>` placeholder with the name of the SQL Server Agent role. Then replace the `<user_name>` placeholder with the name of the user who you previously granted with the required privileges. For more information, see [Adding a user to the SQLAgentUser role in the Amazon RDS User Guide](#).

To detect log shipping, grant the SELECT on `dbo.log_shipping_primary_databases` privilege on the msdb database.

To use the notification approach of the data definition language (DDL) replication, grant the RECEIVE ON `<schema_name>. <queue_name>` privilege on your source databases. In this example, replace the `<schema_name>` placeholder with the schema name of your database. Then, replace the `<queue_name>` placeholder with the name of a queue table.

## Using an Oracle database as a source in DMS Schema Conversion

DMS Schema Conversion supports the following versions of on-premises Oracle databases: 10.2 and later (for versions 10.x), 11g and up to 12.2, 18c, and 19c. Also, you can connect to the following versions of Amazon RDS for Oracle databases: 11g (versions 11.2.0.4 and later) and up to 12.2, 18c, and 19c.

To connect to your Oracle database, use the Oracle System ID (SID). To find the Oracle SID, submit the following query to your Oracle database:

```
SELECT sys_context('userenv','instance_name') AS SID FROM dual;
```

You can use DMS Schema Conversion to convert database code objects from Oracle Database to the following targets:

- Aurora MySQL
- Aurora PostgreSQL
- RDS for MySQL
- RDS for PostgreSQL

For more information about using DMS Schema Conversion with a source Oracle database, see the [Oracle to PostgreSQL migration step-by-step walkthrough](#).

## Privileges for Oracle as a source

The following privileges are required for Oracle as a source:

- CONNECT
- SELECT\_CATALOG\_ROLE
- SELECT ANY DICTIONARY
- SELECT ON SYS.ARGSUMENT\$

# Creating target data providers in DMS Schema Conversion

You can use MySQL and PostgreSQL databases as a target data provider in migration projects for DMS Schema Conversion. Your target data provider can be an Amazon EC2, an Amazon RDS, or an Amazon Aurora instance.

### Topics

- [Using a MySQL database as a target in DMS Schema Conversion \(p. 68\)](#)
- [Using a PostgreSQL database as a target in DMS Schema Conversion \(p. 69\)](#)

## Using a MySQL database as a target in DMS Schema Conversion

You can use MySQL databases version 8.x as a target database in DMS Schema Conversion.

### Privileges for MySQL as a target

The following privileges are required for MySQL as a target:

- CREATE ON \*.\*
- ALTER ON \*.\*
- DROP ON \*.\*
- INDEX ON \*.\*
- REFERENCES ON \*.\*
- SELECT ON \*.\*
- CREATE VIEW ON \*.\*
- SHOW VIEW ON \*.\*
- TRIGGER ON \*.\*
- CREATE ROUTINE ON \*.\*
- ALTER ROUTINE ON \*.\*
- EXECUTE ON \*.\*
- CREATE TEMPORARY TABLES ON \*.\*
- INVOKE LAMBDA ON \*.\*
- INSERT, UPDATE ON AWS\_ORACLE\_EXT.\*
- INSERT, UPDATE, DELETE ON AWS\_ORACLE\_EXT\_DATA.\*
- INSERT, UPDATE ON AWS\_SQLSERVER\_EXT.\*
- INSERT, UPDATE, DELETE ON AWS\_SQLSERVER\_EXT\_DATA.\*

- CREATE TEMPORARY TABLES ON AWS\_SQLSERVER\_EXT\_DATA.\*

To use RDS for MySQL as a target, set the `log_bin_trust_function_creators` parameter to 1 and the `character_set_server` parameter to latin1.

To use Amazon Aurora MySQL as a target, set the `lower_case_table_names` parameter to 1. Also, set the `log_bin_trust_function_creators` parameter to 1, and the `character_set_server` parameter to latin1.

## Using a PostgreSQL database as a target in DMS Schema Conversion

You can use PostgreSQL databases version 14.x as a target database in DMS Schema Conversion.

### Privileges for PostgreSQL as a target

To use PostgreSQL as a target, DMS Schema Conversion requires the `CREATE ON DATABASE` privilege. Create a user and grant this user with this privilege for each database that you want to use in migration project for DMS Schema Conversion.

To use Amazon RDS for PostgreSQL as a target, DMS Schema Conversion requires the `rds_superuser` role.

To use the converted public synonyms, change the database default search path using the following command.

```
ALTER DATABASE <db_name> SET SEARCH_PATH = "$user", public_synonyms, public;
```

In this example, replace the `<db_name>` placeholder with the name of your database.

In PostgreSQL, only the schema owner or a superuser can drop a schema. The owner can drop a schema and all objects that this schema includes, even if the owner of the schema doesn't own some of its objects.

When you use different users to convert and apply different schemas to your target database, you may encounter an error message when DMS Schema Conversion can't drop a schema. To avoid this error message, use the `superuser` role.

## Working with migration projects in DMS Schema Conversion

After you create data providers, your instance profile, and other AWS resources, you can create a migration project for DMS Schema Conversion. A *migration project* describes your source and target data providers, your instance profile, and migration rules. You can create multiple migration projects for different source and target data providers.

The migration project is the foundation of your work with DMS Schema Conversion. Here, you assess the objects of your source data provider and convert them to a format compatible with the target database. Then, you can apply converted code to your target data provider or save it as a SQL script.

Migration projects in DMS Schema Conversion are serverless only. AWS DMS automatically provisions the cloud resources for your migration projects.

You can create up to 3 migration projects in DMS Schema Conversion.

#### Topics

- [Creating migration projects in DMS Schema Conversion \(p. 70\)](#)
- [Managing migration projects in DMS Schema Conversion \(p. 71\)](#)
- [Specifying migration project settings in DMS Schema Conversion \(p. 71\)](#)

## Creating migration projects in DMS Schema Conversion

Before you create a migration project for DMS Schema Conversion, make sure that you create the following resources:

- Data providers that describe your source and target databases
- Secrets with database credentials stored in AWS Secrets Manager
- The AWS Identity and Access Management (IAM) role that provides access to Secrets Manager
- An instance profile that includes network and security settings

#### To create a migration project

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Migration projects**. The **Migration projects** page opens.
3. Choose **Create migration project**. The following table describes the settings.

Option	Action
<b>Name</b>	Enter a name for your migration project. Make sure that you use a unique name for your migration project so that you can easily identify it.
<b>Instance profile</b>	Choose your instance profile to use for your migration project.
<b>Source</b>	Choose <b>Browse</b> , and then choose your source data provider.
<b>Secret ID</b>	Choose the Amazon Resource Name (ARN) of your secret in Secrets Manager that stores your source database credentials.
<b>IAM role</b>	Choose an IAM role to provide access to your source database credentials in Secrets Manager.
<b>Target</b>	Choose <b>Browse</b> , and then choose your source data provider.
<b>Secret ID</b>	Choose the ARN of your secret in Secrets Manager that stores your target database credentials.
<b>IAM role</b>	Choose an IAM role to provide access to your target database credentials in Secrets Manager.
<b>Transformation rules</b>	Choose <b>Add transformation rule</b> to set up transformation rules. Transformation rules make it possible for you to change the object names according to the rule that you specify. For more information, see <a href="#">Setting up transformation rules (p. 76)</a> .

#### 4. Choose **Create migration project**.

After AWS DMS creates your migration project, you can use the project to create assessment reports and convert your database objects. To start working with your migration project, on the **Migration projects** page, choose your project from the list. Choose **Schema conversion**, then **Launch schema conversion**.

The first launch of DMS Schema Conversion requires some setup. AWS Database Migration Service (AWS DMS) starts a schema conversion instance, which can take 10-15 minutes. This process also reads the metadata from the source and target databases. After a successful first launch, you can access DMS Schema Conversion faster.

The schema conversion instance that your migration project uses may be terminated by Amazon three days after you complete the project. You can retrieve your converted schema and assessment report from the Amazon S3 bucket that you use for DMS Schema Conversion.

## Managing migration projects in DMS Schema Conversion

After you create your migration project, you can modify or delete it. For example, to change the source or target data provider, modify your migration project.

You can modify or delete your migration project only after you close the schema conversion operation. To do so, choose your migration project from the list, and choose **Schema conversion**. Next, choose **Close schema conversion** and confirm your choice. After you edit your migration project, choose **Launch schema conversion**. AWS DMS starts a schema conversion instance again. This operation takes 10-15 minutes.

### To modify a migration project

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Migration projects**. The **Migration projects** page opens.
3. Choose your migration project, and then choose **Modify**.
4. Update the name of your project, edit the instance profile, or change source and target data providers. Optionally, add or edit migration rules that change the object names during conversion.
5. Choose **Save changes**.

### To delete a migration project

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Migration projects**. The **Migration projects** page opens.
3. Choose your migration project, and then choose **Delete**.
4. Choose **Delete** to confirm your choice.

## Specifying migration project settings in DMS Schema Conversion

After you create your migration project and launch schema conversion, you can specify migration project settings. You can change conversion settings to improve the performance of converted code. Also, you can customize your schema conversion view.

Conversion settings depend on your source and target database platforms. For more information, see [Data providers \(p. 64\)](#).

To specify what schemas and databases you want to see in the source and target database panes, use the tree view settings. You can hide empty schemas, empty databases, system databases, and user-defined databases or schemas.

#### To hide databases and schemas in tree view

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Migration projects**. The **Migration projects** page opens.
3. Choose your migration project, choose **Schema conversion**, then **Launch schema conversion**.
4. Choose **Settings**. The **Settings** page opens.
5. In the **Tree view** section, do the following:
  - Choose **Hide empty schemas** to hide empty schemas.
  - Choose **Hide empty databases** to hide empty databases.
  - For **System databases or schemas**, choose system databases and schemas by name to hide them.
  - For **User-defined databases or schemas**, enter the names of user-defined databases and schemas that you want to hide. Choose **Add**. The names are case-insensitive.

To add multiple databases or schemas, use a comma to separate their names. To add multiple objects with a similar name, use the percent (%) as a wildcard. This wildcard replaces any number of any symbols in the database or schema name.

Repeat these steps for the **Source** and **Target** sections.

6. Choose **Apply**, and then choose **Schema conversion**.

## Creating database migration assessment reports with DMS Schema Conversion

An important part of DMS Schema Conversion is the report that it generates to help you convert your schema. This *database migration assessment report* summarizes all of the schema conversion tasks. It also details the action items for schema that can't be converted to the DB engine of your target DB instance. You can view the report in the AWS DMS console or save a copy of this report as a PDF or comma-separated value (CSV) files.

The migration assessment report includes the following:

- An executive summary
- Recommendations, including conversion of server objects, backup suggestions, and linked server changes

When you have items that DMS Schema Conversion can't converted automatically, the report provides estimates of how much effort is required to write the equivalent code for your target DB instance.

#### Topics

- [Creating a database migration assessment report \(p. 73\)](#)
- [Viewing your database migration assessment report \(p. 73\)](#)

- Saving your database migration assessment report (p. 74)

## Creating a database migration assessment report

After you create a migration project, use the following procedure to create a database migration assessment report.

### To create a database migration assessment report

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Migration projects**. The **Migration projects** page opens.
3. Choose your migration project, and then choose **Schema conversion**.
4. Choose **Launch schema conversion**. The **Schema conversion** page opens.
5. In the source database pane, choose the database schema or schema items that you want to assess. To include multiple objects in the report, make sure that you select all items.
6. After you select the check boxes for all schema objects that you want to assess, you must choose the parent node for the selected objects. The **Actions** menu in the source database pane is now available.
7. Choose **Assess** in the **Actions** menu. A confirmation dialog box appears.
8. Choose **Assess** in the dialog box to confirm your choice.

## Viewing your database migration assessment report

After you create an assessment report, DMS Schema Conversion adds information in the following tabs:

- **Summary**
- **Action items**

The **Summary** tab shows the number of items that DMS Schema Conversion can automatically convert.

The **Action items** tab shows items that DMS Schema Conversion can't convert automatically, and provides recommendations about how to manage these items.

### Assessment report summary

The **Summary** tab displays the summary information from the database migration assessment report. It shows the number of items that DMS Schema Conversion can automatically convert for database storage objects and database code objects.

In most cases, DMS Schema Conversion can't automatically convert all schema items to the target database engine. The **Summary** tab provides an estimate of the required effort to create schema items in your target DB instance that are equivalent to those in your source.

To see the conversion summary for database storage objects such as tables, sequences, constraints, data types, and so on, choose **Database storage objects**.

To see the conversion summary for database code objects such as procedures, functions, views, triggers, and so on, choose **Database code objects**.

To change the scope of the assessment report, select the required node in the source database tree. DMS Schema Conversion updates the assessment report summary to match the selected scope.

## Assessment report action items

The **Action items** tab contains a list of items that DMS Schema Conversion can't automatically convert to a format compatible with the target database engine. For each action item, DMS Schema Conversion provides the description of the issue and the recommended action. DMS Schema Conversion groups similar action items and displays the number of occurrences.

To view the code for the related database object, select an action item in the list.

## Saving your database migration assessment report

After you create a database migration assessment report, you can save a copy of this report as a PDF or comma-separated value (CSV) files.

### To save a database migration assessment report as a PDF file

1. Choose **Export**, then choose **PDF**. Review the dialog box, and choose **Export to PDF**.
2. DMS Schema Conversion creates an archive with your PDF file and stores this archive in your Amazon S3 bucket. To change the Amazon S3 bucket, edit the schema conversion settings in your instance profile.
3. Open the assessment report file in your Amazon S3 bucket.

### To save a database migration assessment report as CSV files

1. Choose **Export**, then choose **CSV**. Review the dialog box, and choose **Export to CSV**.
2. DMS Schema Conversion creates an archive with CSV files and stores this archive in your Amazon S3 bucket. To change the Amazon S3 bucket, edit the schema conversion settings in your instance profile.
3. Open the assessment report files in your Amazon S3 bucket.

The PDF file contains both the summary and action item information.

When you export your assessment report to CSV, DMS Schema Conversion creates three CSV files.

The first CSV file contains the following information about action items:

- Category
- Occurrence
- Action item
- Subject
- Group
- Description
- Documentation references
- Recommended action
- Line
- Position
- Source
- Target
- Server IP address and port
- Database

- Schema

The second CSV file includes the Action\_Items\_Summary suffix in its name and contains the following information:

- Schema
- Action item
- Number of occurrences
- Learning curve efforts, which is the amount of effort required to design an approach to converting each action item
- Efforts to convert an occurrence of the action item, which shows the effort required to convert each action item, following the designed approach
- Action item description
- Recommended action

The values that indicate the level of required efforts are based on a weighted scale, ranging from low (least) to high (most).

The third CSV file includes Summary in its name and contains the following information:

- Category
- Number of objects
- Objects automatically converted
- Objects with simple actions
- Objects with medium-complexity actions
- Objects with complex actions
- Total lines of code

## Using DMS Schema Conversion

DMS Schema Conversion converts your existing database schemas and a majority of the database code objects to a format compatible with the target database.

DMS Schema Conversion automates much of the process of converting your online transaction processing (OLTP) database schemas to Amazon RDS for MySQL or RDS for PostgreSQL. The source and target database engines contain many different features and capabilities, and DMS Schema Conversion attempts to create an equivalent schema wherever possible. For database objects where direct conversion isn't possible, DMS Schema Conversion provides a list of actions for you to take.

To convert your database schema, use the following process:

- Before you convert your database schemas, set up transformation rules that change the names of your database objects during conversion.
- Create a database migration assessment report to estimate the complexity of the migration. This report provides details about the schema elements that DMS Schema Conversion can't convert automatically.
- Convert your source database storage and code objects. DMS Schema Conversion creates a local version of the converted database objects. You can access these converted objects in your migration project.
- Save the converted code to SQL files to review, edit, or address conversion action items. Optionally, apply the converted code directly to your target database.

To convert data warehouse schemas, use the desktop AWS Schema Conversion Tool. For more information, see [Converting data warehouse schemas to Amazon Redshift](#) in the *AWS Schema Conversion Tool User Guide*.

#### Topics

- [Setting up transformation rules in DMS Schema Conversion \(p. 76\)](#)
- [Converting database schemas in DMS Schema Conversion \(p. 77\)](#)
- [Specifying schema conversion settings for migration projects \(p. 79\)](#)
- [Refreshing your database schemas in DMS Schema Conversion \(p. 82\)](#)
- [Saving and applying your converted code in DMS Schema Conversion \(p. 82\)](#)

## Setting up transformation rules in DMS Schema Conversion

Before you convert your database schema with DMS Schema Conversion, you can set up transformation rules. *Transformation rules* can do such things as change an object name to lowercase or uppercase, add or remove a prefix or suffix, and rename objects. For example, suppose that you have a set of tables in your source schema named `test_TABLE_NAME`. You can set up a rule that changes the prefix `test_` to the prefix `demo_` in the target schema.

You can create transformation rules that perform the following tasks:

- Add, remove, or replace a prefix
- Add, remove, or replace a suffix
- Change the data type of a column
- Change the object name to lowercase or uppercase
- Rename objects

You can create transformation rules for the following objects:

- Schema
- Table
- Column

## Creating transformation rules

DMS Schema Conversion stores transformation rules as part of your migration project. You can set up transformation rules when you create your migration project, or edit them later.

You can add multiple transformation rules in your project. DMS Schema Conversion applies transformation rules during conversion in the same order as you added them.

### To create transformation rules

1. On the **Create migration project** page, choose **Add transformation rule**. For more information, see [Creating migration projects \(p. 70\)](#).
2. For **Rule target**, choose the type of database objects to which this rule applies.
3. For **Source schema**, choose **Enter a schema**. Then, enter the names of your source schemas, tables, and columns to which this rule applies. You can enter an exact name to select one object, or you can

enter a pattern to select multiple objects. Use the percent (%) as a wildcard to replace any number of any symbols in the database object name.

4. For **Action**, choose the task to perform.
5. Depending on the rule type, enter one or two additional values. For example, to rename an object, enter the new name of the object. To replace a prefix, enter the old prefix and the new prefix.
6. Choose **Add transformation rule** to add another transformation rule.

After you are done adding rules, choose **Create migration project**.

To duplicate an existing transformation rule, choose **Duplicate**. To edit an existing transformation rule, choose the rule from the list. To delete an existing transformation rule, choose **Remove**.

## Editing transformation rules

You can add new, remove, or edit existing transformation rules in your migration project. Because DMS Schema Conversion applies the transformation rules during the launch of schema conversion, make sure that you close schema conversion and launch it again after you edit your rules.

### To edit transformation rules

1. Sign in to the AWS Management Console, and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Migration projects**, and then choose your migration project.
3. Choose **Schema conversion**, and then choose **Close schema conversion**.
4. After AWS DMS closes schema conversion, choose **Modify** to edit your migration project settings.
5. For **Transformation rules**, choose one of the following actions:
  - Choose **Duplicate** to duplicate an existing transformation rule and add it in the end of the list.
  - Choose **Remove** to remove an existing transformation rule.
  - Choose the existing transformation rule to edit it.
6. After you are done editing rules, choose **Save changes**.
7. On the **Migration projects** page, choose your project from the list. Choose **Schema conversion**, then choose **Launch schema conversion**.

## Converting database schemas in DMS Schema Conversion

After you create the migration project and connect to your source and target databases, you can convert your source database objects to a format compatible with your target database. DMS Schema Conversion displays your source database schema in the left panel in a tree-view format.

Each node of the database tree is *lazy loaded*. When you choose a node in the tree view, DMS Schema Conversion requests the schema information from your source database at that time. To load the schema information faster, choose your schema, and then choose **Load metadata** from the **Actions** menu. DMS Schema Conversion then reads the database metadata and stores the information on an Amazon S3 bucket. You can now browse the database objects faster.

You can convert the whole database schema, or you can choose any schema item from your source database to convert. If the schema item that you choose depends on a parent item, then DMS Schema Conversion also generates the schema for the parent item. For example, when you choose a table to convert, DMS Schema Conversion creates the converted table and the database schema that the table is in.

## Converting database objects

You can use DMS Schema Conversion to convert an entire database schema or separate database schema objects.

### To convert an entire database schema

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Migration projects**. The **Migration projects** page opens.
3. Choose your migration project, and then choose **Schema conversion**.
4. Choose **Launch schema conversion**. The **Schema conversion** page opens.
5. In the source database pane, select the check box for the schema name.
6. Choose this schema in the left pane of the migration project. DMS Schema Conversion highlights the schema name in blue and activates the **Actions** menu.
7. For **Actions**, choose **Convert**. The conversion dialog box appears.
8. Choose **Convert** in the dialog box to confirm your choice.

### To convert your source database objects

1. Sign in to the AWS Management Console, and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Migration projects**. The **Migration projects** page opens.
3. Choose your migration project, and then choose **Schema conversion**.
4. Choose **Launch schema conversion**. The **Schema conversion** page opens.
5. In the source database pane, select your source database objects.
6. After you select all check boxes for the objects that you want to convert, choose the parent node for all selected objects in your left panel.  
  
DMS Schema Conversion highlights the parent node in blue and activates the **Actions** menu.
7. For **Actions**, choose **Convert**. The conversion dialog box appears.
8. Choose **Convert** in the dialog box to confirm your choice.

For example, to convert two out of 10 tables, select the check boxes for the two tables that you want to convert. Notice that the **Actions** menu is inactive. After you choose the **Tables** node, DMS Schema Conversion highlights its name in blue and activates the **Actions** menu. Then you can choose **Convert** from this menu.

Likewise, to convert two tables and three procedures, select the check boxes for the object names. Then, choose the schema node to activate the **Actions** menu, and choose **Convert schema**.

## Reviewing converted database objects

After you have converted your source database objects, you can choose an object in the left pane of your project. You can then view the source and converted code for that object. DMS Schema Conversion automatically loads the converted code for the object that you selected in the left pane. You can also see the properties or parameters of the object that you selected.

DMS Schema Conversion automatically stores the converted code as part of your migration project. It doesn't apply these code changes to your target database. For more information about applying converted code to your target database, see [Applying your converted code \(p. 83\)](#). To remove the converted code from your migration project, select your target schema in the right pane, and then choose **Refresh from database** from **Actions**.

After you have converted your source database objects, you can see the conversion summary and action items in the lower-center pane. You can see the same information when you create an assessment report. The assessment report is useful for identifying and resolving schema items that DMS Schema Conversion can't convert. You can save the assessment report summary and the list of conversion action items in CSV files. For more information, see [Database migration assessment reports \(p. 72\)](#).

## Specifying schema conversion settings for migration projects

After you create a migration project, you can specify conversion settings in DMS Schema Conversion. Configuring your schema conversion settings improves the performance of the converted code.

### To edit conversion settings

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Migration projects**. The **Migration projects** page opens.
3. Choose your migration project. Choose **Schema conversion**, then **Launch schema conversion**.
4. Choose **Settings**. The **Settings** page opens.
5. In the **Conversion** section, change the settings.
6. Choose **Apply**, and then choose **Schema conversion**.

For all conversion pairs, you can limit the number of comments with action items in the converted code. To limit the number of comments in the converted code, open the conversion settings in your migration project.

For the **Comments in converted SQL code**, choose the severity level of action items. DMS Schema Conversion adds comments in the converted code for action items of the selected severity and higher. For example, to minimize the number of comments in your converted code, choose **Errors only**.

To include comments for all action items in your converted code, choose **All messages**.

Other conversion settings are different for each pair of source and target databases.

### Topics

- [Oracle to MySQL conversion settings \(p. 79\)](#)
- [Oracle to PostgreSQL conversion settings \(p. 80\)](#)
- [SQL Server to MySQL conversion settings \(p. 81\)](#)
- [SQL Server to PostgreSQL conversion settings \(p. 81\)](#)

## Oracle to MySQL conversion settings

Oracle to MySQL conversion settings in DMS Schema Conversion include the following:

- Your source Oracle database can use the ROWID pseudocolumn. MySQL doesn't support similar functionality. DMS Schema Conversion can emulate the ROWID pseudocolumn in the converted code. To do so, turn on the **Generate row ID** option.

If your source Oracle code doesn't use the ROWID pseudocolumn, turn off the **Generate row ID** option. In this case, the converted code works faster.

- Your source Oracle code can include the TO\_CHAR, TO\_DATE, and TO\_NUMBER functions with parameters that MySQL doesn't support. By default, DMS Schema Conversion emulates the usage of these parameters in the converted code.

You can use native MySQL TO\_CHAR, TO\_DATE, and TO\_NUMBER functions when your source Oracle code lacks parameters that are unsupported by MySQL. In this case, the converted code works faster. To do so, select the following values:

- **Use a native MySQL TO\_CHAR function**
- **Use a native MySQL TO\_DATE function**
- **Use a native MySQL TO\_NUMBER function**
- Your database and applications can run in different time zones. By default, DMS Schema Conversion emulates time zones in the converted code. However, you don't need this emulation when your database and applications use the same time zone. In this case, select **Improve the performance of the converted code where the database and applications use the same time zone**.

## Oracle to PostgreSQL conversion settings

Oracle to PostgreSQL conversion settings in DMS Schema Conversion include the following:

- AWS DMS can convert Oracle materialized views to tables or materialized views on PostgreSQL. For **Materialized views**, choose how to convert your source materialized views.
- Your source Oracle database can use the ROWID pseudocolumn. PostgreSQL doesn't support similar functionality. DMS Schema Conversion can emulate the ROWID pseudocolumn in the converted code using the bigint or character varying data type. To do so, choose **Use the bigint data type to emulate the ROWID pseudocolumn** or **Use the character varying data type to emulate the ROWID pseudocolumn** for Row ID.

If your source Oracle code doesn't use the ROWID pseudocolumn, choose **Don't generate**. In this case, the converted code works faster.

- Your source Oracle code can include the TO\_CHAR, TO\_DATE, and TO\_NUMBER functions with parameters that PostgreSQL doesn't support. By default, DMS Schema Conversion emulates the usage of these parameters in the converted code.

You can use native PostgreSQL TO\_CHAR, TO\_DATE, and TO\_NUMBER functions when your source Oracle code lacks parameters that are unsupported by PostgreSQL. In this case, the converted code works faster. To do so, select the following values:

- **Use a native PostgreSQL TO\_CHAR function**
- **Use a native PostgreSQL TO\_DATE function**
- **Use a native PostgreSQL TO\_NUMBER function**
- Your database and applications can run in different time zones. By default, DMS Schema Conversion emulates time zones in the converted code. However, you don't need this emulation when your database and applications use the same time zone. In this case, select **Improve the performance of the converted code where the database and applications use the same time zone**.
- To continue using sequences in your converted code, select **Populate converted sequences with the last value generated on the source side**.
- In some cases, your source Oracle database might store only integer values in the primary or foreign key columns of the NUMBER data type. In these cases, AWS DMS can convert these columns to the BIGINT data type. This approach improves the performance of your converted code. To do so, select **Convert primary and foreign key columns of the NUMBER data type to the BIGINT data type**. Make sure that your source doesn't include floating point values in these columns to avoid data loss.
- To skip deactivated triggers and constraints in your source code, choose **Convert only active triggers and constraints**.
- You can use DMS Schema Conversion to convert string variables that are called as dynamic SQL. Your database code can change the values of these string variables. To make sure that AWS DMS always converts the latest value of this string variable, select **Convert the dynamic SQL code that is created in called routines**.

- PostgreSQL versions 10 and earlier don't support procedures. If you aren't familiar with using procedures in PostgreSQL, AWS DMS can convert Oracle procedures to PostgreSQL functions. To do so, select **Convert procedures to functions**.
- To see additional information about the occurred action items, you can add specific functions to the extension pack. To do so, select **Add extension pack functions that raise user-defined exceptions**. Then choose severity levels to raise user-defined exceptions. Make sure that you apply the extension pack schema after you convert your source database objects. For more information about extension packs, see [Using extension packs \(p. 83\)](#).
- Your source Oracle database can include constraints with the automatically generated names. If your source code uses these names, make sure that you select **Keep the names of system generated constraints**. If your source code uses these constraints, but doesn't use their names, clear this option to increase the conversion speed.
- If your source and target databases run in different time zones, the function that emulates the SYSDATE built-in Oracle function returns different values compared to the source function. To make sure that your source and target functions return the same values, choose **Set the time zone of your source database**.
- You can use the functions from the orafce extension in your converted code. To do so, for **Orafce built-in routines**, select the functions to use. For more information about orafce, see [orafce](#) on GitHub.

## SQL Server to MySQL conversion settings

SQL Server to MySQL conversion settings in DMS Schema Conversion include the following:

- Your source SQL Server database can store the output of EXEC in a table. DMS Schema Conversion creates temporary tables and an additional procedure to emulate this feature. To use this emulation, select **Create additional routines to handle open datasets**.

## SQL Server to PostgreSQL conversion settings

SQL Server to PostgreSQL conversion settings in DMS Schema Conversion include the following:

- In SQL Server, you can use indexes with the same name in different tables. However, in PostgreSQL, all index names that you use in the schema must be unique. To make sure that DMS Schema Conversion generates unique names for all your indexes, select **Generate unique names for indexes**.
- PostgreSQL versions 10 and earlier don't support procedures. If you aren't familiar with using procedures in PostgreSQL, AWS DMS can convert SQL Server procedures to PostgreSQL functions. To do so, select **Convert procedures to functions**.
- Your source SQL Server database can store the output of EXEC in a table. DMS Schema Conversion creates temporary tables and an additional procedure to emulate this feature. To use this emulation, select **Create additional routines to handle open datasets**.
- You can define the template to use for the schema names in the converted code. For **Schema names**, choose one of the following options:
  - **DB** – Uses the SQL Server database name as a schema name in PostgreSQL.
  - **SCHEMA** – Uses the SQL Server schema name as a schema name in PostgreSQL.
  - **DB\_SCHEMA** – Uses a combination of the SQL Server database and schema names as a schema name in PostgreSQL.
- You can keep the letter case of your source object names. To avoid conversion of object names to lowercase, select **Keep object names in the same case**. This option applies only when you turn on the case sensitivity option in your target database.
- You can keep the parameter names from your source database. DMS Schema Conversion can add double quotation marks to the names of parameters in the converted code. To do so, select **Keep original parameter names**.

## Refreshing your database schemas in DMS Schema Conversion

After you create a migration project, DMS Schema Conversion stores the information about your source and target schemas in this project. DMS Schema Conversion uses *lazy loading* to load metadata only as it is needed, such as when you choose a node in your database tree. You can use *eager loading* to load the schema information faster. To do so, choose your schema, and then choose **Load metadata** from **Actions**.

After you automatically or manually load the object to your migration project, DMS Schema Conversion doesn't use lazy loading again. So when you change objects, such as tables and procedures in your database, make sure to refresh them in your migration project.

To refresh schemas from the database, select the objects that you want to refresh, and choose **Refresh from database** from **Actions**. You can refresh database objects in your source and target database schemas:

- **Source** – If you update your source database schema, choose **Refresh from database** to replace the schema in your project with the latest schema from your source database.
- **Target** – If you update the schema for your target database, DMS Schema Conversion replaces the schema in your project with the latest schema from your target database. DMS Schema Conversion replaces your converted code with the code from your target database. Make sure that you applied the converted code to your target database before you choose **Refresh from database**. Otherwise, convert your source database schema again.

## Saving and applying your converted code in DMS Schema Conversion

After DMS Schema Conversion converts your source database objects, it doesn't immediately apply the converted code to your target database. Instead, DMS Schema Conversion stores the converted code in your project until you are ready to apply it to your target database.

Before you apply the converted code, you can update your source database code and convert the updated objects again to address the existing action items. For more information about items that DMS Schema Conversion can't convert automatically, see [Creating database migration assessment reports with DMS Schema Conversion \(p. 72\)](#). For more information about refreshing your source database objects in migration project for DMS Schema Conversion, see [Refreshing your database schemas \(p. 82\)](#).

Instead of applying the converted code directly to your database in DMS Schema Conversion, you can save the code to a file as a SQL script. You can review these SQL scripts, edit them where necessary, and then manually apply these SQL scripts to your target database.

### Saving your converted code to a SQL file

You can save your converted schema as SQL scripts in a text file. You can modify the converted code to address action items that DMS Schema Conversion can't convert automatically. You can then run your updated SQL scripts on your target database to apply the converted code to your target database.

#### To save your converted schema as SQL scripts

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Migration projects**. The **Migration projects** page opens.

3. Choose your migration project, and then choose **Schema conversion**.
4. Choose **Launch schema conversion**. The **Schema conversion** page opens.
5. In the right pane, choose the target database schema or select the converted objects that you want to save. Make sure that DMS Schema Conversion highlights the parent node name in blue and activates the **Actions** menu for the target database.
6. Choose **Save as SQL for Actions**. The **Save** dialog box appears.
7. Choose **Save as SQL** to confirm your choice.

DMS Schema Conversion creates an archive with SQL files and stores this archive in your Amazon S3 bucket.
8. (Optional) Change the S3 bucket for the archive by editing the schema conversion settings in your instance profile.
9. Open the SQL scripts from your S3 bucket.

## Applying your converted code

When you are ready to apply your converted code to your target database, choose the database objects in the right pane of your project. You can apply changes to an entire database schema or selected database schema objects.

After you select the database objects, DMS Schema Conversion highlights the name of the selected node or the parent node in blue. It then activates the **Actions** menu. Choose **Apply changes for Actions**. In the dialog box that appears, choose **Apply** to confirm your choice and apply the converted code to your target database.

## Applying the extension pack schema

When you apply your converted schema to your target database for the first time, DMS Schema Conversion might also apply the extension pack schema. The extension pack schema emulates system functions of the source database that are required to run your converted code for your target database. If your converted code uses the functions of the extension pack, make sure that you apply the extension pack schema.

To apply the extension pack to your target database manually, choose **Apply changes for Actions**. In the dialog box that appears, choose **confirm** to apply the extension pack to your target database.

We recommend that you don't modify the extension pack schema to avoid unexpected results in the converted code.

For more information, see [Using extension packs in DMS Schema Conversion \(p. 83\)](#).

# Using extension packs in DMS Schema Conversion

An *extension pack* in DMS Schema Conversion is an add-on module that emulates source database functions that aren't supported in the target database. Use an extension pack to make sure that the converted code produces the same results as the source code. Before you can install an extension pack, convert your database schemas.

Each extension pack includes a database schema. This schema includes SQL functions, procedures, tables, and views for emulating specific online transaction processing (OLTP) objects or unsupported built-in functions from the source database.

When you convert your source database, DMS Schema Conversion adds an additional schema to your target database. This schema implements SQL system functions of the source database that are required

to run your converted code on your target database. This additional schema is called the extension pack schema.

The extension pack schema is named according to your source database as follows:

- Microsoft SQL Server – aws\_sqlserver\_ext
- Oracle – aws\_oracle\_ext

You can apply extension packs in two ways:

- DMS Schema Conversion can automatically apply an extension pack when you apply your converted code. DMS Schema Conversion applies the extension pack before it applies all other schema objects.
- You can apply an extension pack manually. To do so, choose the extension pack schema in your target database tree, and then choose **Apply**, then **Apply extension pack**.

# Best practices for AWS Database Migration Service

To use AWS Database Migration Service (AWS DMS) most effectively, see this section's recommendations on the most efficient way to migrate your data.

## Topics

- [Migration planning for AWS Database Migration Service \(p. 85\)](#)
- [Converting schema \(p. 86\)](#)
- [Reviewing the AWS DMS public documentation \(p. 86\)](#)
- [Running a proof of concept \(p. 87\)](#)
- [Improving the performance of an AWS DMS migration \(p. 87\)](#)
- [Using your own on-premises name server \(p. 90\)](#)
- [Migrating large binary objects \(LOBs\) \(p. 92\)](#)
- [Improving performance when migrating large tables using row filtering \(p. 95\)](#)
- [Ongoing replication \(p. 95\)](#)
- [Reducing the load on your source database \(p. 96\)](#)
- [Reducing the bottlenecks on your target database \(p. 96\)](#)
- [Using data validation during migration \(p. 96\)](#)
- [Monitoring your AWS DMS tasks using metrics \(p. 96\)](#)
- [Events and notifications \(p. 97\)](#)
- [Using the task log to troubleshoot migration issues \(p. 97\)](#)
- [Troubleshooting replication tasks with Time Travel \(p. 97\)](#)
- [Changing the user and schema for an Oracle target \(p. 98\)](#)
- [Changing table and index tablespaces for an Oracle target \(p. 98\)](#)
- [Upgrading a replication instance version \(p. 99\)](#)
- [Understanding your migration cost \(p. 99\)](#)

## Migration planning for AWS Database Migration Service

When planning a database migration using AWS Database Migration Service, consider the following:

- To connect your source and target databases to an AWS DMS replication instance, you configure a network. Doing this can be as simple as connecting two AWS resources in the same virtual private cloud (VPC) as your replication instance. It can range to more complex configurations such as connecting an on-premises database to an Amazon RDS DB instance over a virtual private network (VPN). For more information, see [Network configurations for database migration \(p. 112\)](#).
- **Source and target endpoints** – Make sure that you know what information and tables in the source database need to be migrated to the target database. AWS DMS supports basic schema migration, including the creation of tables and primary keys. However, AWS DMS doesn't automatically create

secondary indexes, foreign keys, user accounts, and so on, in the target database. Depending on your source and target database engine, you might need to set up supplemental logging or modify other settings for a source or target database. For more information, see [Sources for data migration \(p. 134\)](#) and [Targets for data migration \(p. 293\)](#).

- **Schema and code migration** – AWS DMS doesn't perform schema or code conversion. You can use tools such as Oracle SQL Developer, MySQL Workbench, and pgAdmin III to convert your schema. To convert an existing schema to a different database engine, you can use the AWS Schema Conversion Tool (AWS SCT). It can create a target schema and can generate and create an entire schema: tables, indexes, views, and so on. You can also use the tool to convert PL/SQL or TSQL to PgSQL and other formats. For more information on the AWS SCT, see the [AWS SCT User Guide](#).
- **Unsupported data types** – Make sure that you can convert source data types into the equivalent data types for the target database. For more information on supported data types, see the source or target section for your data store.
- **Diagnostic support script results** – When you plan your migration, we recommend that you run diagnostic support scripts. With the results from these scripts, you can find advance information about potential migration failures.

If a support script is available for your database, download it using the link in the corresponding script topic in the following section. After verifying and reviewing the script, you can run it according to the procedure described in the script topic in your local environment. When the script run is complete, you can review the results. We recommend running these scripts as a first step of any troubleshooting effort. The results can be useful while working with an AWS Support team. For more information, see [Working with diagnostic support scripts in AWS DMS \(p. 674\)](#).

- **Premigration assessments** – A *premigration assessment* evaluates specified components of a database migration task to help identify any problems that might prevent a migration task from running as expected. By using this assessment, you can identify potential problems before you run a new or modified task. For more information on working with premigration assessments, see [Enabling and working with premigration assessments for a task \(p. 548\)](#).

## Converting schema

AWS DMS doesn't perform schema or code conversion. If you want to convert an existing schema to a different database engine, you can use AWS SCT. AWS SCT converts your source objects, table, indexes, views, triggers, and other system objects into the target data definition language (DDL) format. You can also use AWS SCT to convert most of your application code, like PL/SQL or TSQL, to the equivalent target language.

You can get AWS SCT as a free download from AWS. For more information on AWS SCT, see the [AWS SCT User Guide](#).

If your source and target endpoints are on the same database engine, you can use tools such as Oracle SQL Developer, MySQL Workbench, or PgAdmin4 to move your schema.

## Reviewing the AWS DMS public documentation

We highly recommended that you go through the AWS DMS public documentation pages for your source and target endpoints before your first migration. This documentation can help you to identify the prerequisites for the migration and understand the current limitations before you begin. For more information, see [Working with AWS DMS endpoints \(p. 131\)](#).

During migration, the public documentation can help you to troubleshoot any issues with AWS DMS. Troubleshooting pages in the documentation can help you to resolve common issues using both AWS

DMS and selected endpoint databases. For more information, see [Troubleshooting migration tasks in AWS Database Migration Service \(p. 654\)](#).

## Running a proof of concept

To help discover issues with your environment in early phases of your database migration, we recommend that you run a small test migration. Doing this can also help you to set a more realistic migration time line. In addition, you might need to run a full-scale test migration to measure whether AWS DMS can handle the throughput of your database over your network. During this time, we recommend to benchmark and optimize your initial full load and ongoing replication. Doing this can help you to understand your network latency and gauge overall performance.

At this point, you also have an opportunity to understand your data profile and how large your database is, including the following:

- How many tables are large, medium, and small in size.
- How AWS DMS handles data type and character-set conversions.
- How many tables having large object (LOB) columns.
- How long it takes to run a test migration.

## Improving the performance of an AWS DMS migration

A number of factors affect the performance of your AWS DMS migration:

- Resource availability on the source.
- The available network throughput.
- The resource capacity of the replication server.
- The ability of the target to ingest changes.
- The type and distribution of source data.
- The number of objects to be migrated.

You can improve performance by using some or all of the best practices mentioned following. Whether you can use one of these practices depends on your specific use case. You can find some limitations following:

### Provisioning a proper replication server

AWS DMS is a managed service that runs on an Amazon EC2 instance. This service connects to the source database, reads the source data, formats the data for consumption by the target database, and loads the data into the target database.

Most of this processing happens in memory. However, large transactions might require some buffering on disk. Cached transactions and log files are also written to disk. In the following sections, you can find what to consider when you choose your replication server.

#### CPU

AWS DMS is designed for heterogeneous migrations, but it also supports homogeneous migrations. To perform a homogeneous migration, first convert each source data type to its equivalent AWS

DMS data type. Then convert each AWS DMS type data to the target data type. You can find references for these conversions for each database engine within the *AWS DMS User Guide*.

For AWS DMS to perform these conversions optimally, the CPU must be available when the conversions happen. Overloading the CPU and not having enough CPU resources can result in slow migrations, which can also cause other side effects.

#### Replication instance class

Some of the smaller instance classes are sufficient for testing the service or for small migrations. If your migration involves a large number of tables, or if you intend to run multiple concurrent replication tasks, consider using one of the larger instances. A larger instance can be a good idea because the service consumes a fair amount of memory and CPU.

T2 type instances are designed to provide moderate baseline performance and the capability to burst to significantly higher performance, as required by your workload. They are intended for workloads that don't use the full CPU often or consistently, but that occasionally need to burst. T2 instances are well suited for general purpose workloads, such as web servers, developer environments, and small databases. If you're troubleshooting a slow migration and using a T2 instance type, check the CPU Utilization host metric. It can show you if you're bursting over the baseline for that instance type.

The C4 instance classes are designed to deliver the highest level of processor performance for computer-intensive workloads. They achieve significantly higher packet per second (PPS) performance, lower network jitter, and lower network latency. AWS DMS can be CPU-intensive, especially when performing heterogeneous migrations and replications such as migrating from Oracle to PostgreSQL. C4 instances can be a good choice for these situations.

The R4 instance classes are memory optimized for memory-intensive workloads. Ongoing migrations or replications of high-throughput transaction systems using AWS DMS can, at times, consume large amounts of CPU and memory. R4 instances include more memory per vCPU.

#### AWS DMS support for R5 and C5 instance classes

The R5 instance classes are memory-optimized instances that are designed to deliver fast performance for workloads that process large data sets in memory. Ongoing migrations or replications of high-throughput transaction systems using AWS DMS can, at times, consume large amounts of CPU and memory. R5 instances deliver 5 percent additional memory per vCPU than R4 and the largest size provides 768 GiB of memory. In addition, R5 instances deliver a 10 percent price per GiB improvement and a ~20% increased CPU performance over R4.

The C5 instance classes optimized for compute-intensive workloads and deliver cost-effective high performance at a low price per compute ratio. They achieve significantly higher network performance. Elastic Network Adapter (ENA) provides C5 instances with up to 25 Gbps of network bandwidth and up to 14 Gbps of dedicated bandwidth to Amazon EBS. AWS DMS can be CPU-intensive, especially when performing heterogeneous migrations and replications such as migrating from Oracle to PostgreSQL. C5 instances can be a good choice for these situations.

#### Storage

Depending on the instance class, your replication server comes with either 50 GB or 100 GB of data storage. This storage is used for log files and any cached changes that are collected during the load. If your source system is busy or takes large transactions, you might need to increase your storage. If you're running multiple tasks on the replication server, you might also need a storage increase. However, the default amount is usually sufficient.

All storage volumes in AWS DMS are GP2 or General-Purpose solid-state drives (SSDs). GP2 volumes come with a base performance of three I/O operations per second (IOPS), with abilities to burst up to 3,000 IOPS on a credit basis. As a rule of thumb, check the ReadIOPS and WriteIOPS metrics for the replication instance. Make sure that the sum of these values doesn't cross the base performance for that volume.

## Multi-AZ

Choosing a Multi-AZ instance can protect your migration from storage failures. Most migrations are transient and aren't intended to run for long periods of time. If you use AWS DMS for ongoing replication purposes, choosing a Multi-AZ instance can improve your availability should a storage issue occur.

When using a single AZ or Multi-AZ replication instance during a FULL LOAD and a failover or host replacement occurs, the full load task is expected to fail. You can restart the task from the point of failure for the remaining tables that didn't complete, or are in an error state.

## Loading multiple tables in parallel

By default, AWS DMS loads eight tables at a time. You might see some performance improvement by increasing this slightly when using a very large replication server, such as a dms.c4.xlarge or larger instance. However, at some point, increasing this parallelism reduces performance. If your replication server is relatively small, such as a dms.t2.medium, we recommend that you reduce the number of tables loaded in parallel.

To change this number in the AWS Management Console, open the console, choose **Tasks**, choose to create or modify a task, and then choose **Advanced Settings**. Under **Tuning Settings**, change the **Maximum number of tables to load in parallel** option.

To change this number using the AWS CLI, change the `MaxFullLoadSubTasks` parameter under `TaskSettings`.

## Using parallel full load

You can use a parallel load from Oracle, Microsoft SQL Server, MySQL, Sybase, and IBM Db2 LUW sources based on partitions and subpartitions. Doing this can improve overall full load duration. In addition, while running an AWS DMS migration task, you can accelerate the migration of large or partitioned tables. To do this, split the table into segments and load the segments in parallel in the same migration task.

To use a parallel load, create a table mapping rule of type `table-settings` with the `parallel-load` option. Within the `table-settings` rule, specify the selection criteria for the table or tables that you want to load in parallel. To specify the selection criteria, set the `type` element for `parallel-load` to one of the following settings:

- `partitions-auto`
- `subpartitions-auto`
- `partitions-list`
- `ranges`
- `none`

For more information on these settings, see [Table and collection settings rules and operations \(p. 523\)](#).

## Working with indexes, triggers, and referential integrity constraints

Indexes, triggers, and referential integrity constraints can affect your migration performance and cause your migration to fail. How these affect migration depends on whether your replication task is a full load task or an ongoing replication (change data capture, or CDC) task.

For a full load task, we recommend that you drop primary key indexes, secondary indexes, referential integrity constraints, and data manipulation language (DML) triggers. Or you can delay their creation until after the full load tasks are complete. You don't need indexes during a full load task, and indexes incur maintenance overhead if they are present. Because the full load task loads groups of tables at a time, referential integrity constraints are violated. Similarly, insert, update, and delete

triggers can cause errors, for example if a row insert is triggered for a previously bulk loaded table. Other types of triggers also affect performance due to added processing.

If your data volumes are relatively small and the additional migration time doesn't concern you, you can build primary key and secondary indexes before a full load task. Always turn off referential integrity constraints and triggers.

For a full load plus CDC task, we recommend that you add secondary indexes before the CDC phase. Because AWS DMS uses logical replication, make sure that secondary indexes that support DML operations are in place to prevent full table scans. You can pause the replication task before the CDC phase to build indexes and create referential integrity constraints before you restart the task.

You should enable triggers right before the cutover.

#### Turn off backups and transaction logging

When migrating to an Amazon RDS database, it's a good idea to turn off backups and Multi-AZ on the target until you're ready to cut over. Similarly, when migrating to systems other than Amazon RDS, turning off any logging on the target until after cutover is usually a good idea.

#### Use multiple tasks

Sometimes using multiple tasks for a single migration can improve performance. If you have sets of tables that don't participate in common transactions, you might be able to divide your migration into multiple tasks. Transactional consistency is maintained within a task, so it's important that tables in separate tasks don't participate in common transactions. Also, each task independently reads the transaction stream, so be careful not to put too much stress on the source database.

You can use multiple tasks to create separate streams of replication. By doing this, you can parallelize the reads on the source, the processes on the replication instance, and the writes to the target database.

#### Optimizing change processing

By default, AWS DMS processes changes in a transactional mode, which preserves transactional integrity. If you can afford temporary lapses in transactional integrity, you can use the batch optimized apply option instead. This option efficiently groups transactions and applies them in batches for efficiency purposes. Using the batch optimized apply option almost always violates referential integrity constraints. So we recommend that you turn these constraints off during the migration process and turn them on again as part of the cutover process.

## Using your own on-premises name server

Usually, an AWS DMS replication instance uses the Domain Name System (DNS) resolver in an Amazon EC2 instance to resolve domain endpoints. However, you can use your own on-premises name server to resolve certain endpoints if you use the Amazon Route 53 Resolver. With this tool, you can query between on-premises and AWS using inbound and outbound endpoints, forwarding rules, and a private connection. The benefits of using an on-premises name server include improved security and ease of use behind a firewall.

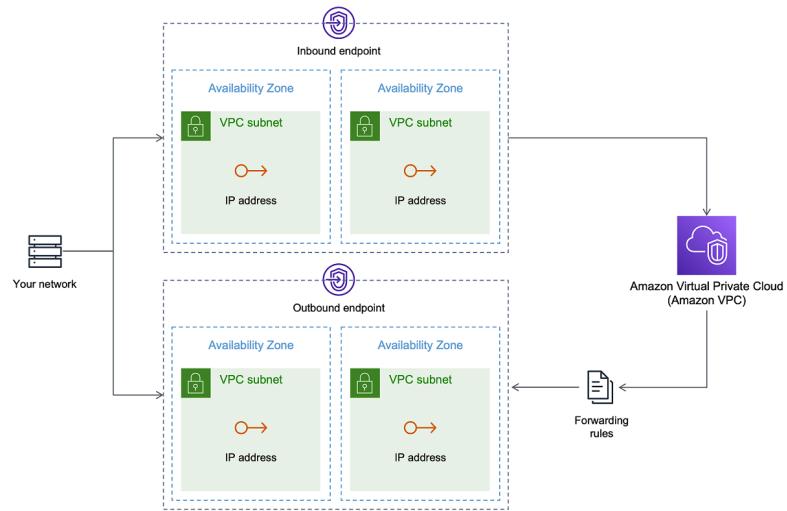
If you have inbound endpoints, you can use DNS queries that originate on-premises to resolve AWS-hosted domains. To configure the endpoints, assign IP addresses in each subnet that you want to provide a resolver. To establish connectivity between your on-premises DNS infrastructure and AWS, use AWS Direct Connect or a virtual private network (VPN).

Outbound endpoints connect to your on-premises name server. The name server only grants access to IP addresses included in an allow list and set in an outbound endpoint. The IP address of your name server is the target IP address. When you choose a security group for an outbound endpoint, choose the same security group used by the replication instance.

To forward select domains to the name server, use forwarding rules. An outbound endpoint can handle multiple forwarding rules. The scope of the forwarding rule is your virtual private cloud (VPC). By using a forwarding rule associated with a VPC, you can provision a logically isolated section of the AWS Cloud. From this logically isolated section, you can launch AWS resources in a virtual network.

You can configure domains hosted within your on-premises DNS infrastructure as conditional forwarding rules that set up outbound DNS queries. When a query is made to one of those domains, rules trigger an attempt to forward DNS requests to servers that were configured with the rules. Again, a private connection over AWS Direct Connect or VPN is required.

The following diagram shows the Route 53 Resolver architecture.



For more information about Route 53 DNS Resolver, see [Getting started with Route 53 Resolver](#) in the [Amazon Route 53 Developer Guide](#).

## Using Amazon Route 53 Resolver with AWS DMS

You can create an on-premises name server for AWS DMS to resolve endpoints using [Amazon Route 53 Resolver](#).

### To create an on-premises name server for AWS DMS based on Route 53

1. Sign in to the AWS Management Console and open the Route 53 console at <https://console.aws.amazon.com/route53/>.
2. On the Route 53 console, choose the AWS Region where you want to configure your Route 53 Resolver. The Route 53 Resolver is specific to a Region.
3. Choose the query direction—inbound, outbound, or both.
4. Provide your inbound query configuration:
  - a. Enter an endpoint name and choose a VPC.
  - b. Assign one or more subnets from within the VPC (for example, choose two for availability).
  - c. Assign specific IP addresses to use as endpoints, or have Route 53 Resolver assign them automatically.
5. Create a rule for your on-premises domain so that workloads inside the VPC can route DNS queries to your DNS infrastructure.
6. Enter one or more IP addresses for your on-premises DNS servers.
7. Submit your rule.

When everything is created, your VPC is associated with your inbound and outbound rules and can start routing traffic.

For more information about Route 53 Resolver, see [Getting started with Route 53 Resolver](#) in the [Amazon Route 53 Developer Guide](#).

## Migrating large binary objects (LOBs)

In general, AWS DMS migrates LOB data in two phases:

1. AWS DMS creates a new row in the target table and populates the row with all data except the associated LOB value.
2. AWS DMS updates the row in the target table with the LOB data.

This migration process for LOBs requires that, during the migration, all LOB columns on the target table must be nullable. This is so even if the LOB columns aren't nullable on the source table. If AWS DMS creates the target tables, it sets LOB columns to nullable by default. In some cases, you might create the target tables using some other mechanism, such as import or export. In such cases, make sure that the LOB columns are nullable before you start the migration task.

This requirement has one exception. Suppose that you perform a homogeneous migration from an Oracle source to an Oracle target, and you choose **Limited Lob mode**. In this case, the entire row is populated at once, including any LOB values. For such a case, AWS DMS can create the target table LOB columns with not-nullable constraints, if needed.

## Using limited LOB mode

AWS DMS uses two methods that balance performance and convenience when your migration contains LOB values:

1. **Limited LOB mode** migrates all LOB values up to a user-specified size limit (default is 32 KB). LOB values larger than the size limit must be manually migrated. **Limited LOB mode**, the default for all migration tasks, typically provides the best performance. However, ensure that the **Max LOB size** parameter setting is correct. Set this parameter to the largest LOB size for all your tables.
2. **Full LOB mode** migrates all LOB data in your tables, regardless of size. **Full LOB mode** provides the convenience of moving all LOB data in your tables, but the process can have a significant impact on performance.

For some database engines, such as PostgreSQL, AWS DMS treats JSON data types like LOBs. Make sure that if you chose **Limited LOB mode**, the **Max LOB size** option is set to a value that doesn't cause the JSON data to be truncated.

AWS DMS provides full support for using large object data types (BLOBS, CLOBS, and NCLOBS). The following source endpoints have full LOB support:

- Oracle
- Microsoft SQL Server
- ODBC

The following target endpoints have full LOB support:

- Oracle

- Microsoft SQL Server

The following target endpoint has limited LOB support. You can't use an unlimited LOB size for this target endpoint.

- Amazon Redshift
- Amazon S3

For endpoints that have full LOB support, you can also set a size limit for LOB data types.

## Improved LOB performance

While migrating LOB data, you can specify the following different LOB optimization settings.

### Per table LOB settings

Using per table LOB settings, you can override task-level LOB settings for some or all of your tables. To do this, define the `lob-settings` in your `table-settings` rule. Following is an example table that includes some large LOB values.

```

SET SERVEROUTPUT ON
CREATE TABLE TEST_CLOB
(
ID NUMBER,
C1 CLOB,
C2 VARCHAR2(4000)
);
DECLARE
bigtextstring CLOB := '123';
iINT;
BEGIN
WHILE Length(bigtextstring) <= 60000 LOOP
bigtextstring := bigtextstring || '0000000000000000000000000000000000000000';
END LOOP;
INSERT INTO TEST_CLOB (ID, C1, C2) VALUES (0, bigtextstring,'AnyValue');
END;
/
SELECT * FROM TEST_CLOB;
COMMIT

```

Next, create a migration task and modify the LOB handling for your table using the new `lob-settings` rule. The `bulk-max-siz` value determines the maximum LOB size (KB). It's truncated if it's bigger than the size specified.

```
{
"rules": [
  {
    "rule-type": "selection",
    "rule-id": "1",
    "rule-name": "1",
    "object-locator": {
      "schema-name": "HR",
      "table-name": "TEST_CLOB"
    },
    "rule-action": "include"
  },
  {
    "rule-type": "table-settings",
    "rule-id": "2",
    "rule-name": "2"
  }
]
```

```

    "rule-name": "2",
    "object-locator": {
        "schema-name": "HR",
        "table-name": "TEST_CLOB"
    },
    "lob-settings": {
        "mode": "limited",
        "bulk-max-size": "16"
    }
}
]
}

```

Even if this AWS DMS task is created with `FullLobMode : true`, the per table LOB settings direct AWS DMS to truncate LOB data in this particular table to 16,000. You can check the task logs to confirm this.

```
721331968: 2018-09-11T19:48:46:979532 [SOURCE_UNLOAD] W: The value of column 'C' in table 'HR.TEST_CLOB' was truncated to length 16384
```

## Inline LOB settings

When you create an AWS DMS task, the LOB mode determines how LOBs are handled.

With full LOB mode and limited LOB mode, each has its own benefits and disadvantages. Inline LOB mode combines the advantages of both full LOB mode and limited LOB mode.

You can use inline LOB mode when you need to replicate both small and large LOBs, and most of the LOBs are small. When you choose this option, during full load the AWS DMS task transfers the small LOBs inline, which is more efficient. The AWS DMS task transfers the large LOBs by performing a lookup from the source table.

During change processing, both small and large LOBs are replicated by performing a lookup from the source table.

When you use inline LOB mode, the AWS DMS task checks all of the LOB sizes to determine which ones to transfer inline. LOBs larger than the specified size are replicated using full LOB mode. Therefore, if you know that most of the LOBs are larger than the specified setting, it's better not to use this option. Instead, allow an unlimited LOB size.

You configure this option using an attribute in task settings, `InlineLobMaxSize`, which is only available when `FullLobMode` is set to `true`. The default value for `InlineLobMaxSize` is 0 and the range is 1 – 102400 kilobytes (100 MB).

For example, you might use the following AWS DMS task settings. Here, setting `InlineLobMaxSize` to a value of 5 results in all LOBs smaller than or equal to 5,000 being transferred inline.

```
{
    "TargetMetadata": {
        "TargetSchema": "",
        "SupportLobs": true,
        "FullLobMode": true,
        "LobChunkSize": 64,
        "LimitedSizeLobMode": false,
        "LobMaxSize": 32,
        "InlineLobMaxSize": 5,
        "LoadMaxFileSize": 0,
        "ParallelLoadThreads": 0,
        "ParallelLoadBufferSize": 0,
        "BatchApplyEnabled": false,
        "TaskRecoveryTableEnabled": false},
}
```

} . . .

## Improving performance when migrating large tables using row filtering

To improve the performance when migrating a large table, break the migration into more than one task. To break the migration into multiple tasks using row filtering, use a key or a partition key. For example, if you have an integer primary key ID from 1 to 8,000,000, you can create eight tasks using row filtering to migrate 1 million records each.

### To apply row filtering in the console:

1. Open the AWS Management Console.
2. Choose **Tasks**, and create a new task.
3. Choose the **Table mappings** tab, and expand **Selection rules**.
4. Choose **Add new selection rule**. You can now add a column filter with either a less than or equal to, greater than or equal to, equal to, or a range condition between two values. For more information on column filtering, see [Specifying table selection and transformations rules from the console \(p. 491\)](#).

If you have a large partitioned table that is partitioned by date, you can migrate data based on date. For example, suppose that you have a table partitioned by month, and only the current month's data is updated. In this case, you can create a full load task for each static monthly partition and create a full load plus CDC task for the currently updated partition.

If your table has a single-column primary key or unique index, you can have your AWS DMS task segment the table using a parallel load of the ranges type to load the data in parallel. For more information, see [Table and collection settings rules and operations \(p. 523\)](#).

## Ongoing replication

AWS DMS provides ongoing replication of data, keeping the source and target databases in sync. It replicates only a limited amount of data definition language (DDL) statements. AWS DMS doesn't propagate items such as indexes, users, privileges, stored procedures, and other database changes not directly related to table data.

If you plan to use ongoing replication, set the **Multi-AZ** option when you create your replication instance. By choosing **Multi-AZ**, you get high availability and failover support for the replication instance. However, this option can have an impact on performance and can slow down replication while applying changes to the target system.

Before you upgrade your source or target databases, we recommend that you stop any AWS DMS tasks that are running on these databases. Resume the tasks after your upgrades are complete.

During ongoing replication, it's critical to identify the network bandwidth between your source database system and your AWS DMS replication instance. Make sure that the network doesn't cause any bottlenecks during ongoing replication.

It's also important to identify the rate of change and archive log generation per hour on your source database system. Doing this can help you to understand the throughput that you might get during ongoing replication.

## Reducing the load on your source database

AWS DMS uses some resources on your source database. During a full load task, AWS DMS performs a full table scan of the source table for each table processed in parallel. Also, each task that you create as part of a migration queries the source for changes as part of the CDC process. For AWS DMS to perform CDC for some sources, such as Oracle, you might need to increase the amount of data written to your database's change log.

If you find that you're overburdening your source database, reduce the number of tasks or tables for each task for your migration. Each task gets source changes independently, so consolidating tasks can decrease the change capture workload.

## Reducing the bottlenecks on your target database

During the migration, try to remove any processes that compete for write resources on your target database:

- Turn off unnecessary triggers.
- Turn off secondary indexes during initial load and turn them back on later during ongoing replication.
- With Amazon RDS databases, it's a good idea to turn off backups and Multi-AZ until the cutover.
- While migrating to non-RDS systems, it's a good idea turn off any logging on the target until the cutover.

## Using data validation during migration

To ensure that your data was migrated accurately from the source to the target, we highly recommend that you use data validation. If you turn on data validation for a task, AWS DMS begins comparing the source and target data immediately after a full load is performed for a table.

Data validation works with the following databases wherever AWS DMS supports them as source and target endpoints:

- Oracle
- PostgreSQL
- MySQL
- MariaDB
- Microsoft SQL Server
- Amazon Aurora MySQL-Compatible Edition
- Amazon Aurora PostgreSQL-Compatible Edition
- IBM Db2 LUW

For more information, see [AWS DMS data validation \(p. 584\)](#).

## Monitoring your AWS DMS tasks using metrics

You have several options for monitoring metrics for your tasks using the AWS DMS console:

### Host metrics

You can find host metrics on the **CloudWatch metrics** tab for each particular replication instance. Here, you can monitor whether your replication instance is sized appropriately.

### Replication task metrics

Metrics for replication tasks, including incoming and committed changes, and latency between the replication host and source/target databases can be found on the **CloudWatch metrics** tab for each particular task.

### Table metrics

You can find individual table metrics on the **Table statistics** tab for each individual task. These metrics include these numbers:

- Rows loaded during the full load.
- Inserts, updates, and deletes since the task started.
- DDL operations since the task started.

For more information on metrics monitoring, see [Monitoring AWS DMS tasks \(p. 558\)](#).

## Events and notifications

AWS DMS uses Amazon SNS to provide notifications when an AWS DMS event occurs, for example the creation or deletion of a replication instance. You can work with these notifications in any form supported by Amazon SNS for an AWS Region. These can include email messages, text messages, or calls to an HTTP endpoint.

For more information, see [Working with Amazon SNS events and notifications in AWS Database Migration Service \(p. 576\)](#).

## Using the task log to troubleshoot migration issues

In some cases, AWS DMS can encounter issues for which warnings or error messages appear only in the task log. In particular, data truncation issues or row rejections due to foreign key violations are only written in the task log. Therefore, be sure to review the task log when migrating a database. To view the task log, configure Amazon CloudWatch as part of task creation.

For more information, see [Monitoring replication tasks using Amazon CloudWatch](#).

## Troubleshooting replication tasks with Time Travel

To troubleshoot AWS DMS migration issues, you can work with Time Travel. For more information about Time Travel, see [Time Travel task settings \(p. 455\)](#).

When you work with Time Travel, be aware of the following considerations:

- To avoid overhead on a DMS replication instance, turn on Time Travel only for tasks that need debugging.

- When you use Time Travel to troubleshoot replication tasks that might run for several days, monitor replication instance metrics for resource overheads. This approach applies especially in cases where high transaction loads run on source databases for extended periods of time. For more details, see [Monitoring AWS DMS tasks \(p. 558\)](#).
- When the Time Travel task setting `EnableRawData` is set to `true`, the task memory usage during DMS replication might be higher than when Time Travel isn't turned on. If you turn on Time Travel for extended periods of time, monitor your task.
- Currently, you can turn on Time Travel only at the task level. Changes to all tables are logged in Time Travel logs. If you are troubleshooting for specific tables in a database with high transaction volume, create a separate task .

## Changing the user and schema for an Oracle target

When you use Oracle as a target, AWS DMS migrates the data to the schema owned by the target endpoint's user.

For example, suppose that you're migrating a schema named `PERFDATA` to an Oracle target endpoint, and that the target endpoint user name is `MASTER`. AWS DMS connects to the Oracle target as `MASTER` and populates the `MASTER` schema with database objects from `PERFDATA`.

To override this behavior, provide a schema transformation. For example, to migrate the `PERFDATA` schema objects to a `PERFDATA` schema at the target endpoint, use the following transformation.

```
{  
    "rule-type": "transformation",  
    "rule-id": "2",  
    "rule-name": "2",  
    "object-locator": {  
        "schema-name": "PERFDATA"  
    },  
    "rule-target": "schema",  
    "rule-action": "rename",  
    "value": "PERFDATA"  
}
```

For more information about transformations, see [Specifying table selection and transformations rules using JSON \(p. 494\)](#).

## Changing table and index tablespaces for an Oracle target

When using Oracle as a target, AWS DMS migrates all tables and indexes to the default tablespace in the target. For example, suppose that your source is a database engine other than Oracle. All of the target tables and indexes are migrated to the same default tablespace.

To override this behavior, provide corresponding tablespace transformations. For example, suppose that you want to migrate tables and indexes to table and index tablespaces in the Oracle target that are named after the schema in the source. In this case, you can use transformations similar to the following. Here, the schema in the source is named `INVENTORY` and corresponding table and index tablespaces in the target are named `INVENTORTBL` and `INVENTORYIDX`.

```
{  
    "rule-type": "transformation",
```

```
"rule-id": "3",
"rule-name": "3",
"rule-action": "rename",
"rule-target": "table-tablespace",
"object-locator": {
    "schema-name": "INVENTORY",
    "table-name": "%",
    "table-tablespace-name": "%"
},
"value": "INVENTORYTBL"
},
{
"rule-type": "transformation",
"rule-id": "4",
"rule-name": "4",
"rule-action": "rename",
"rule-target": "index-tablespace",
"object-locator": {
    "schema-name": "INVENTORY",
    "table-name": "%",
    "index-tablespace-name": "%"
},
"value": "INVENTORYIDX"
}
```

For more information about transformations, see [Specifying table selection and transformations rules using JSON \(p. 494\)](#).

When Oracle is both source and target, you can preserve existing table or index tablespace assignments by setting the Oracle source extra connection attribute `enableHomogenousTablespace=true`. For more information, see [Extra connection attributes when using Oracle as a source for AWS DMS \(p. 165\)](#).

## Upgrading a replication instance version

AWS periodically releases new versions of the AWS DMS replication engine software, with new features and performance improvements. Each version of the replication engine software has its own version number. It's critical to test the existing version of your AWS DMS replication instance running a production work load before you upgrade your replication instance to a later version. For more information on available version upgrades, see [AWS DMS release notes \(p. 712\)](#).

## Understanding your migration cost

AWS Database Migration Service helps you migrate databases to AWS easily and securely at a low cost. You only pay for your replication instances and any additional log storage. Each database migration instance includes storage sufficient for swap space, replication logs, and data cache for most replications and inbound data transfer is free.

You might need more resources during initial load or during peak load time. You can closely monitor replication instance resource utilization using cloud watch metrics. You can then scale up and scale down replication instance size based on usage.

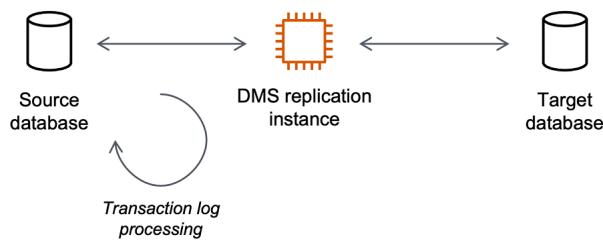
For more information on estimating your migration costs, see:

- [AWS Database Migration Service pricing](#)
- [AWS Pricing Calculator](#)

# Working with an AWS DMS replication instance

When you create an AWS DMS replication instance, AWS DMS creates it on an Amazon EC2 instance in a virtual private cloud (VPC) based on the Amazon VPC service. You use this replication instance to perform your database migration. By using a replication instance, you can get high availability and failover support with a Multi-AZ deployment when you choose the **Multi-AZ** option.

In a Multi-AZ deployment, AWS DMS automatically provisions and maintains a synchronous standby replica of the replication instance in a different Availability Zone. The primary replication instance is synchronously replicated across Availability Zones to a standby replica. This approach provides data redundancy, eliminates I/O freezes, and minimizes latency spikes.



AWS DMS uses a replication instance to connect to your source data store, read the source data, and format the data for consumption by the target data store. A replication instance also loads the data into the target data store. Most of this processing happens in memory. However, large transactions might require some buffering on disk. Cached transactions and log files are also written to disk.

You can create an AWS DMS replication instance in the following AWS Regions.

Region Name	Region	Endpoint	Protocol	
US East (Ohio)	us-east-2	dms.us-east-2.amazonaws.com dms-fips.us-east-2.amazonaws.com dms-fips.us-east-2.amazonaws.com	HTTPS HTTPS HTTPS	
US East (N. Virginia)	us-east-1	dms.us-east-1.amazonaws.com dms-fips.us-east-1.amazonaws.com dms-fips.us-east-1.amazonaws.com	HTTPS HTTPS HTTPS	
US West (N. California)	us-west-1	dms.us-west-1.amazonaws.com dms-fips.us-west-1.amazonaws.com dms-fips.us-west-1.amazonaws.com	HTTPS HTTPS HTTPS	
US West (Oregon)	us-west-2	dms.us-west-2.amazonaws.com dms-fips.us-west-2.amazonaws.com dms-fips.us-west-2.amazonaws.com	HTTPS HTTPS HTTPS	

<b>Region Name</b>	<b>Region</b>	<b>Endpoint</b>	<b>Protocol</b>	
Africa (Cape Town)	af-south-1	dms.af-south-1.amazonaws.com	HTTPS	
Asia Pacific (Hong Kong)	ap-east-1	dms.ap-east-1.amazonaws.com	HTTPS	
Asia Pacific (Hyderabad)	ap-south-2	dms.ap-south-2.amazonaws.com	HTTPS	
Asia Pacific (Jakarta)	ap-southeast-3	dms.ap-southeast-3.amazonaws.com	HTTPS	
Asia Pacific (Melbourne)	ap-southeast-4	dms.ap-southeast-4.amazonaws.com	HTTPS	
Asia Pacific (Mumbai)	ap-south-1	dms.ap-south-1.amazonaws.com	HTTPS	
Asia Pacific (Osaka)	ap-northeast-3	dms.ap-northeast-3.amazonaws.com	HTTPS	
Asia Pacific (Seoul)	ap-northeast-2	dms.ap-northeast-2.amazonaws.com	HTTPS	
Asia Pacific (Singapore)	ap-southeast-1	dms.ap-southeast-1.amazonaws.com	HTTPS	
Asia Pacific (Sydney)	ap-southeast-2	dms.ap-southeast-2.amazonaws.com	HTTPS	
Asia Pacific (Tokyo)	ap-northeast-1	dms.ap-northeast-1.amazonaws.com	HTTPS	
Canada (Central)	ca-central-1	dms.ca-central-1.amazonaws.com	HTTPS	
Europe (Frankfurt)	eu-central-1	dms.eu-central-1.amazonaws.com	HTTPS	
Europe (Ireland)	eu-west-1	dms.eu-west-1.amazonaws.com	HTTPS	
Europe (London)	eu-west-2	dms.eu-west-2.amazonaws.com	HTTPS	

Region Name	Region	Endpoint	Protocol	
Europe (Milan)	eu-south-1	dms.eu-south-1.amazonaws.com	HTTPS	
Europe (Paris)	eu-west-3	dms.eu-west-3.amazonaws.com	HTTPS	
Europe (Spain)	eu-south-2	dms.eu-south-2.amazonaws.com	HTTPS	
Europe (Stockholm)	eu-north-1	dms.eu-north-1.amazonaws.com	HTTPS	
Europe (Zurich)	eu-central-2	dms.eu-central-2.amazonaws.com	HTTPS	
Middle East (Bahrain)	me-south-1	dms.me-south-1.amazonaws.com	HTTPS	
Middle East (UAE)	me-central-1	dms.me-central-1.amazonaws.com	HTTPS	
South America (São Paulo)	sa-east-1	dms.sa-east-1.amazonaws.com	HTTPS	
AWS GovCloud (US-East)	us-gov-east-1	dms.us-gov-east-1.amazonaws.com dms.us-gov-east-1.amazonaws.com dms.us-gov-east-1.amazonaws.com	HTTPS HTTPS HTTPS	
AWS GovCloud (US-West)	us-gov-west-1	dms.us-gov-west-1.amazonaws.com dms.us-gov-west-1.amazonaws.com dms.us-gov-west-1.amazonaws.com	HTTPS HTTPS HTTPS	

AWS DMS supports a special AWS Region called AWS GovCloud (US) that is designed to allow US government agencies and customers to move sensitive workloads into the cloud. AWS GovCloud (US) addresses the US government's specific regulatory and compliance requirements. For more information about AWS GovCloud (US), see [What is AWS GovCloud \(US\)?](#)

Following, you can find out more details about replication instances.

### Topics

- [Choosing the right AWS DMS replication instance for your migration \(p. 103\)](#)
- [Choosing the best size for a replication instance \(p. 107\)](#)
- [Working with replication engine versions \(p. 109\)](#)
- [Public and private replication instances \(p. 110\)](#)
- [IP addressing and network types \(p. 111\)](#)
- [Setting up a network for a replication instance \(p. 112\)](#)
- [Setting an encryption key for a replication instance \(p. 119\)](#)

- [Creating a replication instance \(p. 119\)](#)
- [Modifying a replication instance \(p. 123\)](#)
- [Rebooting a replication instance \(p. 125\)](#)
- [Deleting a replication instance \(p. 127\)](#)
- [Working with the AWS DMS maintenance window \(p. 128\)](#)

## Choosing the right AWS DMS replication instance for your migration

AWS DMS creates the replication instance on an Amazon EC2 instance. AWS DMS currently supports the T2, T3, C4, C5, C6i, R4, R5 and R6i Amazon EC2 instance classes for replication instances:

- T2 instances are burstable performance instances that provide a baseline level of CPU performance with the ability to burst above the baseline. The baseline performance and ability to burst are governed by CPU credits. T2 instances receive CPU credits continuously at a set rate depending on the instance size. They accumulate CPU credits when they are idle and consume CPU credits when they are active.

T2 instances are a good choice for a variety of general-purpose workloads. These include microservices, low-latency interactive applications, small and medium databases, virtual desktops, development, build and stage environments, code repositories, and product prototypes.
- T3 instances are the next-generation burstable general-purpose instance type. This type provides a baseline level of CPU performance with the ability to burst CPU usage at any time for as long as required. T3 instances offer a balance of compute, memory, and network resources and are designed for applications with moderate CPU usage that experience temporary spikes in use. T3 instances accumulate CPU credits when a workload is operating below baseline threshold. Each earned CPU credit provides the T3 instance the opportunity to burst with the performance of a full CPU core for one minute when needed.

T3 instances can burst at any time for as long as required in unlimited mode. For more information on unlimited mode, see [Working with unlimited mode for burstable performance instances \(p. 106\)](#).

- C4 instances are optimized for compute-intensive workloads and deliver very cost-effective high performance at a low price per compute ratio. They achieve significantly higher packet per second (PPS) performance, lower network jitter, and lower network latency. AWS DMS can also be CPU-intensive, especially when performing heterogeneous migrations and replications such as migrating from Oracle to PostgreSQL. C4 instances can be a good choice for these situations.
- C5 instances are the next-generation instance type to deliver cost-effective high performance at a low price per compute ratio for running advanced compute-intensive workloads. This includes workloads such as high-performance web servers, high-performance computing (HPC), batch processing, ad serving, highly scalable multiplayer gaming, and video encoding. Other workloads C5 instances are suited to include scientific modeling, distributed analytics, and machine and deep learning inference. The C5 instances are available with a choice of processors from Intel and AMD.
- C6i instances offer up to 15% better compute price performance over comparable Gen5 instances for a wide variety of workloads, and always-on memory encryption. C6i instances are an ideal fit for compute-intensive workloads such as batch processing, distributed analytics, high performance computing (HPC), ad serving, highly scalable multiplayer gaming, and video encoding.
- R4 instances are memory optimized for memory-intensive workloads. Ongoing migrations or replications of high-throughput transaction systems using AWS DMS can also consume large amounts of CPU and memory. R4 instances include more memory per vCPU than earlier generation instance types.

- R5 instances are the next generation of memory-optimized instance types for Amazon EC2. R5 instances are well-suited for memory-intensive applications such as high performance databases, distributed web scale in-memory caches, midsize in-memory databases, real time big data analytics, and other enterprise applications. Ongoing migrations or replications of high-throughput transaction systems using AWS DMS can also consume large amounts of CPU and memory.
- R6i instances offer up to 15% better compute price performance over comparable Gen5 instances for a wide variety of workloads, and always-on memory encryption. R6i instances are SAP Certified and are ideal for workloads such as SQL and noSQL databases, distributed web scale in-memory caches like Memcached and Redis, in-memory databases like SAP HANA, and real time big data analytics like Hadoop and Spark clusters.

Each replication instance has a specific configuration of memory and vCPU. The following table shows the configuration for each replication instance type. For pricing information, see the [AWS Database Migration Service service pricing page](#).

Replication instance type	vCPU	Memory (GiB)
General Purpose		
dms.t2.micro	1	1
dms.t2.small	1	2
dms.t2.medium	2	4
dms.t2.large	2	8
dms.t3.micro	2	1
dms.t3.small	2	2
dms.t3.medium	2	4
dms.t3.large	2	8
Compute Optimized		
dms.c4.large	2	3.75
dms.c4.xlarge	4	7.5
dms.c4.2xlarge	8	15
dms.c4.4xlarge	16	30
dms.c5.large	2	4
dms.c5.xlarge	4	8
dms.c5.2xlarge	8	16
dms.c5.4xlarge	16	32
dms.c5.9xlarge	36	72
dms.c5.12xlarge	48	96
dms.c5.18xlarge	72	144
dms.c5.24xlarge	96	192

<b>Replication instance type</b>	<b>vCPU</b>	<b>Memory (GiB)</b>
dms.c6i.large	2	4
dms.c6i.xlarge	4	8
dms.c6i.2xlarge	8	16
dms.c6i.4xlarge	16	32
dms.c6i.8xlarge	32	64
dms.c6i.12xlarge	48	96
dms.c6i.16xlarge	64	128
dms.c6i.24xlarge	96	192
dms.c6i.32xlarge	128	256
Memory Optimized		
dms.r4.large	2	15.25
dms.r4.xlarge	4	30.5
dms.r4.2xlarge	8	61
dms.r4.4xlarge	16	122
dms.r4.8xlarge	32	244
dms.r5.large	2	16
dms.r5.xlarge	4	32
dms.r5.2xlarge	8	64
dms.r5.4xlarge	16	128
dms.r5.8xlarge	32	256
dms.r5.12xlarge	48	384
dms.r5.16xlarge	64	512
dms.r5.24xlarge	96	768
dms.r6i.large	2	16
dms.r6i.xlarge	4	32
dms.r6i.2xlarge	8	64
dms.r6i.4xlarge	16	128
dms.r6i.8xlarge	32	256
dms.r6i.12xlarge	48	384
dms.r6i.16xlarge	64	512
dms.r6i.24xlarge	96	768

Replication instance type	vCPU	Memory (GiB)
dms.r6i.32xlarge	128	1024

#### Topics

- [Deciding what instance class to use \(p. 106\)](#)
- [Working with unlimited mode for burstable performance instances \(p. 106\)](#)

## Deciding what instance class to use

To help determine which replication instance class might work best for you, let's look at the change data capture (CDC) process that AWS DMS uses.

Let's assume that you're running a full load plus CDC task (bulk load plus ongoing replication). In this case, the task has its own SQLite repository to store metadata and other information. Before AWS DMS starts a full load, these steps occur:

- AWS DMS starts capturing changes for the tables it's migrating from the source engine's transaction log (we call these *cached changes*). After full load is done, these cached changes are collected and applied on the target. Depending on the volume of cached changes, these changes can directly be applied from memory, where they are collected first, up to a set threshold. Or they can be applied from disk, where changes are written when they can't be held in memory.
- After cached changes are applied, by default AWS DMS starts a transactional apply process on the target instance.

During the applied cached changes phase and ongoing replications phase, AWS DMS uses two stream buffers, one each for incoming and outgoing data. AWS DMS also uses an important component called a *sorter*, which is another memory buffer. Following are two important uses of the sorter component (which has others):

- It tracks all transactions and makes sure that it forwards only relevant transactions to the outgoing buffer.
- It makes sure that transactions are forwarded in the same commit order as on the source.

As you can see, we have three important memory buffers in this architecture for CDC in AWS DMS. If any of these buffers experience memory pressure, the migration can have performance issues that can potentially cause failures.

When you plug heavy workloads with a high number of transactions per second (TPS) into this architecture, you can find the extra memory provided by R5 and R6i instances useful. You can use R5 and R6i instances to hold a large number of transactions in memory and prevent memory-pressure issues during ongoing replications.

## Working with unlimited mode for burstable performance instances

A burstable performance instance configured as **unlimited**, such as a T3 instance, can sustain high CPU utilization for any period of time whenever required. The hourly instance price can automatically cover all CPU usage spikes. It does so if the average CPU utilization of the instance is at or below the baseline over a rolling 24-hour period or the instance lifetime, whichever is shorter.

For the vast majority of general-purpose workloads, instances configured as unlimited give enough performance without any additional charges. If the instance runs at higher CPU utilization for a prolonged period, it can do so for a flat additional rate per vCPU-hour. For information about T3 instance pricing, see "T3 CPU Credits" in [AWS Database Migration Service](#).

For more information on unlimited mode for T3 instances, see [Unlimited mode for burstable performance instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

**Important**

If you use a dms .t3.micro instance under the [AWS Free Tier](#) offer and use it in unlimited mode, charges might apply. In particular, charges might apply if your average utilization over a rolling 24-hour period exceeds the baseline utilization of the instance. For more information, see [Baseline utilization](#) in the *Amazon EC2 User Guide for Linux Instances*.

T3 instances launch as unlimited by default. If the average CPU usage over a 24-hour period exceeds the baseline, you incur charges for surplus credits. In some cases, you might launch T3 Spot Instances as unlimited and plan to use them immediately and for a short duration. If you do so with no idle time for accruing CPU credits, you incur charges for surplus credits. We recommend that you launch your T3 Spot Instances in standard mode to avoid paying higher costs. For more information, see [Surplus credits can incur charges](#), [T3 Spot Instances](#), and [Standard mode for burstable performance instances](#) in the *Amazon EC2 User Guide for Linux Instances*.

## Choosing the best size for a replication instance

Choosing the appropriate replication instance depends on several factors of your use case. To help understand how replication instance resources are used, see the following discussion. It covers the common scenario of a full load + CDC task.

During a full load task, AWS DMS loads tables individually. By default, eight tables are loaded at a time. AWS DMS captures ongoing changes to the source during a full load task so the changes can be applied later on the target endpoint. The changes are cached in memory; if available memory is exhausted, changes are cached to disk. When a full load task completes for a table, AWS DMS immediately applies the cached changes to the target table.

After all outstanding cached changes for a table have been applied, the target endpoint is in a transactionally consistent state. At this point, the target is in-sync with the source endpoint with respect to the last cached changes. AWS DMS then begins ongoing replication between the source and target. To do so, AWS DMS takes change operations from the source transaction logs and applies them to the target in a transactionally consistent manner. (This process assumes batch optimized apply isn't selected). AWS DMS streams ongoing changes through memory on the replication instance, if possible. Otherwise, AWS DMS writes changes to disk on the replication instance until they can be applied on the target.

You have some control over how the replication instance handles change processing, and how memory is used in that process. For more information on how to tune change processing, see [Change processing tuning settings \(p. 465\)](#).

## Factors to consider

Memory and disk space are key factors in selecting an appropriate replication instance for your use case. Following, you can find a discussion of the use case characteristics to analyze to choose a replication instance.

- Database and table size

Data volume helps determine the task configuration to optimize full load performance. For example, for two 1 TB schemas, you can partition tables into four tasks of 500 GB and run them in parallel. The

possible parallelism depends on the CPU resource available in the replication instance. That's why it's a good idea understand the size of your database and tables to optimize full load performance. It helps determine the number of tasks that you can possibly have.

- Large objects

The data types that are present in your migration scope can affect performance. Particularly, large objects (LOBs) impact performance and memory consumption. To migrate a LOB value, AWS DMS performs a two-step process. First, AWS DMS inserts the row into the target without the LOB value. Second, AWS DMS updates the row with the LOB value. This has an impact on the memory, so it's important to identify LOB columns in the source and analyze their size.

- Load frequency and transaction size

Load frequency and transactions per second (TPS) influence memory usage. A high number of TPS or data manipulation language (DML) activities leads to high usage of memory. This happens because DMS caches the changes until they are applied to the target. During CDC, this leads to swapping (writing to the physical disk due to memory overflow), which causes latency.

- Table keys and referential integrity

Information about the keys of the table determines the CDC mode (batch apply or transactional apply) that you use to migrate data. In general, transactional apply is slower than batch apply. For long-running transactions, there can be many changes to migrate. When you use transactional apply, AWS DMS might require more memory to store the changes compared to batch apply. If you migrate tables without primary keys, batch apply will fail and the DMS task moves to transactional apply mode. When referential integrity is active between tables during CDC, AWS DMS uses transactional apply by default. For more information about batch apply compared to transactional apply, see [How can I use the DMS batch apply feature to improve CDC replication performance?](#) .

Use these metrics to determine if you need the replication instance to be compute optimized or memory optimized.

## Common issues

You might face the following common issues that cause resource contention on the replication instance during migration. For information on the replication instance metrics, see [Replication instance metrics \(p. 564\)](#).

- If the memory in a replication instance becomes insufficient, this results in writing data to the disk. Reading from the disk can cause latency, which you can avoid by sizing the replication instance with enough memory.
- The disk size assigned to the replication instance can be smaller than required. The disk size is used when data in memory spills over; it's also used to store the task logs. The maximum IOPS depends on it too.
- Running multiple tasks or tasks with high parallelism affects CPU consumption of the replication instance. This slows down the processing of the tasks and results in latency.

## Best practices

Consider these two most common best practices when sizing a replication instance. For more information, see [Best practices for AWS Database Migration Service \(p. 85\)](#).

1. Size your workload and understand if it's computer-intensive or memory-intensive. Based on this, you can determine the class and size of the replication instance:
  - AWS DMS processes LOBs in memory. This operation requires a fair amount of memory.

- The number of tasks and the number of threads impact CPU consumption. Avoid using more than eight MaxFullLoadSubTasks during the full load operation.
2. Increase the disk space assigned to the replication instance when you have a high workload during full load. Doing this lets the replication instance use the maximum IOPS assigned to it.

The preceding guidelines don't cover all possible scenarios. It's important to consider the specifics of your particular use case when you determine the size of your replication instance.

The preceding tests show CPU and memory vary with different workloads. Particularly, LOBs affect memory, and task count or parallelism affect the CPU. After your migration is running, monitor the CPU, freeable memory, free storage, and IOPS of your replication instance. Based on the data you gather, you can size your replication instance up or down as needed.

## Working with replication engine versions

The *replication engine* is the core AWS DMS software that runs on your replication instance and performs the migration tasks you specify. AWS periodically releases new versions of the AWS DMS replication engine software, with new features and performance improvements. Each version of the replication engine software has its own version number, to distinguish it from other versions.

When you launch a new replication instance, it runs the latest AWS DMS engine version unless you specify otherwise. For more information, see [Working with an AWS DMS replication instance \(p. 100\)](#).

If you have a replication instance that is currently running, you can upgrade it to a more recent engine version. (AWS DMS doesn't support engine version downgrades.) For more information about replication engine versions, see [AWS DMS release notes \(p. 712\)](#).

## Upgrading the engine version using the console

You can upgrade an AWS DMS replication instance using the AWS Management Console.

### To upgrade a replication instance using the console

1. Open the AWS DMS console at <https://console.aws.amazon.com/dms/>.
2. In the navigation pane, choose **Replication instances**.
3. Choose your replication engine, and then choose **Modify**.
4. For **Engine version**, choose the version number you want, and then choose **Modify**.

#### Note

We recommend that you stop all tasks before upgrading the Replication Instance. If you don't stop the task, AWS DMS will stop the task automatically before the upgrade. If you stop the task manually, you will need to start the task manually after the upgrade is complete. Upgrading the replication instance takes several minutes. When the instance is ready, its status changes to **available**.

## Upgrading the engine version using the AWS CLI

You can upgrade an AWS DMS replication instance using the AWS CLI, as follows.

### To upgrade a replication instance using the AWS CLI

1. Determine the Amazon Resource Name (ARN) of your replication instance by using the following command.

```
aws dms describe-replication-instances \
--query "ReplicationInstances[*].
[ReplicationInstanceIdentifier,ReplicationInstanceArn,ReplicationInstanceClass]"
```

In the output, take note of the ARN for the replication instance you want to upgrade, for example:  
`arn:aws:dms:us-east-1:123456789012:rep:6EFQQ06U6EDPRCPKLNPL2SCEY`

2. Determine which replication instance versions are available by using the following command.

```
aws dms describe-orderable-replication-instances \
--query "OrderableReplicationInstances[*].[ReplicationInstanceClass,EngineVersion]"
```

In the output, note the engine version number or numbers that are available for your replication instance class. You should see this information in the output from step 1.

3. Upgrade the replication instance by using the following command.

```
aws dms modify-replication-instance \
--replication-instance-arn arn \
--engine-version n.n.n
```

Replace *arn* in the preceding with the actual replication instance ARN from the previous step.

Replace *n.n.n* with the engine version number that you want, for example: `3.4.5`

#### Note

Upgrading the replication instance takes several minutes. You can view the replication instance status using the following command.

```
aws dms describe-replication-instances \
--query "ReplicationInstances[*].
[ReplicationInstanceIdentifier,ReplicationInstanceStatus]"
```

When the replication instance is ready, its status changes to **available**.

## Public and private replication instances

You can specify if a replication instance has a public or private IP address that the instance uses to connect to the source and target databases.

A *private replication instance* has a private IP address that you can't access outside the replication network. You use a private instance when both source and target databases are in the same network that is connected to the virtual private cloud (VPC) of the replication instance. The network can be connected to the VPC by using a virtual private network (VPN), AWS Direct Connect, or VPC peering.

A *VPC peering* connection is a networking connection between two VPCs. It allows routing using each VPC's private IP addresses as if they were in the same network. For more information about VPC peering, see [VPC peering](#) in the *Amazon VPC User Guide*.

A *public replication instance* can use the VPC security group of the replication instance, and the replication instance's public IP address or the NAT gateway's public IP address. These connections form a network that you use for data migration.

# IP addressing and network types

AWS DMS always creates your replication instance in an Amazon Virtual Private Cloud (VPC). When you create your VPC, you can determine the IP addressing to use: either IPv4 or IPv6, or both. Then, when you create or modify a replication instance, you can specify use of an IPv4 address protocol or an IPv6 address protocol using *dual-stack mode*.

## IPv4 addresses

When you create a VPC, you can specify a range of IPv4 addresses for the VPC in the form of a Classless Inter-Domain Routing (CIDR) block, such as 10.0.0.0/16. A subnet group defines the range of IP addresses in this CIDR block. These IP addresses can be private or public.

A private IPv4 address is an IP address that's not reachable over the internet. You can use private IPv4 addresses for communication between your replication instance and other resources, such as Amazon EC2 instances, in the same VPC. Each replication instance has a private IP address for communication in the VPC.

A public IP address is an IPv4 address that is reachable from the internet. You can use public addresses for communication between your replication instance and resources on the internet. You control whether your replication instance receives a public IP address.

## Dual-stack mode and IPv6 addresses

When you have resources that must communicate with your replication instance over IPv6, use *dual-stack mode*. To use dual-stack mode, make sure that each subnet in the subnet group that you associate with the replication instance has an IPv6 CIDR block associated with it. You can create a new replication subnet group or modify an existing replication subnet group to meet this requirement. Each IPv6 address is globally unique. The IPv6 CIDR block for your VPC is automatically assigned from the Amazon pool of IPv6 addresses. You can't choose the range yourself.

DMS disables Internet Gateway access for IPv6 endpoints of private dual-stack mode replication instances. DMS does this to ensure that your IPv6 endpoints are private and can only be accessed from within your VPC.

You can use the AWS DMS Console to create or modify a replication instance, and specify dual-stack mode in the **Network type** section. The following image shows the **Network type** section in the console.

## Connectivity and security

### Network type - new Info

To use dual-stack mode, make sure that you associate an IPv6 CIDR block with a subnet in the VPC you specify.

#### IPv4

Replication instance with an IPv4 network type that supports IPv4 addressing.

#### Dual-stack mode

Replication instance with a dual network type that supports both IPv4 and IPv6 addressing.

## References

- For mode information about IPv4 and IPv6 addresses, see [IP Addressing](#) in the *Amazon VPC User Guide*.
- For more information about creating a replication instance using dual-stack mode, see [Creating a replication instance \(p. 119\)](#).

- For more information about modifying a replication instance, see [Modifying a replication instance \(p. 123\)](#).

## Setting up a network for a replication instance

AWS DMS always creates the replication instance in a VPC based on Amazon VPC. You specify the VPC where your replication instance is located. You can use your default VPC for your account and AWS Region, or you can create a new VPC.

Make sure that the elastic network interface allocated for your replication instance's VPC is associated with a security group. Also, make sure this security group's rules let all traffic on all ports leave (egress) the VPC. This approach allows communication from the replication instance to your source and target database endpoints, if correct ingress rules are enabled on the endpoints. We recommend that you use the default settings for the endpoints, which allows egress on all ports to all addresses.

The source and target endpoints access the replication instance that is inside the VPC either by connecting to the VPC or by being inside the VPC. The database endpoints must include network access control lists (ACLs) and security group rules (if applicable) that allow incoming access from the replication instance. How you set this up depends on the network configuration that you use. You can use the replication instance VPC security group, the replication instance's private or public IP address, or the NAT gateway's public IP address. These connections form a network that you use for data migration.

### Note

Since an IP address can change as a result of changes to underlying infrastructure, we recommend you use a VPC CIDR range, or route your replication instance outbound traffic through a NAT GW associated Elastic IP. For more information about creating a VPC, including a CIDR block, see [Work with VPCs and subnets](#) in the *Amazon Virtual Private Cloud User Guide*. For more information about Elastic IP addresses, see [Elastic IP addresses](#) in the *Amazon Elastic Compute Cloud User Guide*.

## Network configurations for database migration

You can use several different network configurations with AWS Database Migration Service. The following are common configurations for a network used for database migration.

### Topics

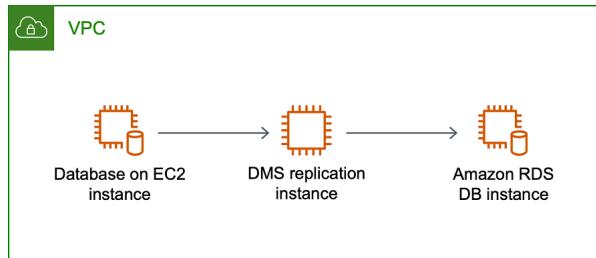
- [Configuration with all database migration components in one VPC \(p. 112\)](#)
- [Configuration with multiple VPCs \(p. 113\)](#)
- [Configuration for a network to a VPC using AWS Direct Connect or a VPN \(p. 113\)](#)
- [Configuration for a network to a VPC using the internet \(p. 114\)](#)
- [Configuration with an RDS DB instance not in a VPC to a DB instance in a VPC using ClassicLink \(p. 114\)](#)

When practical, we recommend that you create a DMS replication instance in the same region as your target endpoint, and in the same VPC or subnet as your target endpoint.

## Configuration with all database migration components in one VPC

The simplest network for database migration is for the source endpoint, the replication instance, and the target endpoint to all be in the same VPC. This configuration is a good one if your source and target endpoints are on an Amazon RDS DB instance or an Amazon EC2 instance.

The following illustration shows a configuration where a database on an Amazon EC2 instance connects to the replication instance and data is migrated to an Amazon RDS DB instance.



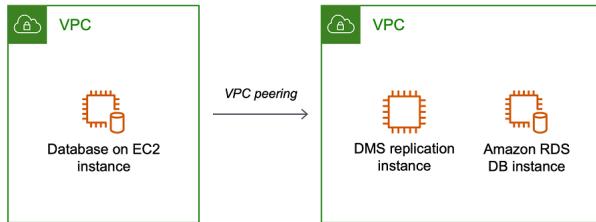
The VPC security group used in this configuration must allow ingress on the database port from the replication instance. You can do this in a couple of ways. You can ensure that the security group used by the replication instance has ingress to the endpoints. Or you can allow the VPC CIDR range, NAT GW Elastic IP, or private IP address of the replication instance if you are using one. But we do not recommend you use the private IP address of the replication instance, because it can break your replication if the replication IP address changes.

## Configuration with multiple VPCs

If your source endpoint and target endpoints are in different VPCs, you can create your replication instance in one of the VPCs. You can then link the two VPCs by using VPC peering.

A VPC peering connection is a networking connection between two VPCs that enables routing using each VPC's private IP addresses as if they were in the same network. You can create a VPC peering connection between your own VPCs, with a VPC in another AWS account, or with a VPC in a different AWS Region. For more information about VPC peering, see [VPC peering](#) in the *Amazon VPC User Guide*.

The following illustration shows an example configuration using VPC peering. Here, the source database on an Amazon EC2 instance in a VPC connects by VPC peering to a VPC. This VPC contains the replication instance and the target database on an Amazon RDS DB instance.



To implement VPC peering, follow the instructions in [Work with VPC peering connections](#) located in the *Amazon Virtual Private Cloud, VPC Peering* documentation. Be sure the route table of one VPC contains the CIDR block of the other. For example, if VPC A is using destination 10.0.0.0/16 and VPC B is using destination 172.31.0.0, the route table of VPC A should contain 172.31.0.0, and route table of VPC B must contain 10.0.0.0/16. For more detailed information, see [Update your route tables for VPC peering connection](#) in the *Amazon Virtual Private Cloud, VPC Peering* documentation.

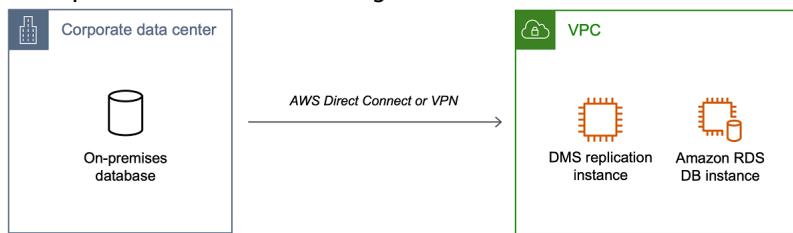
The VPC security groups used in this configuration must allow ingress on the database port from the replication instance, or it should allow ingress on the CIDR block for the VPC being peered.

## Configuration for a network to a VPC using AWS Direct Connect or a VPN

Remote networks can connect to a VPC using several options such as AWS Direct Connect or a software or hardware VPN connection. These options are often used to integrate existing on-site services, such

as monitoring, authentication, security, data, or other systems, by extending an internal network into the AWS cloud. By using this type of network extension, you can seamlessly connect to AWS-hosted resources such as a VPC.

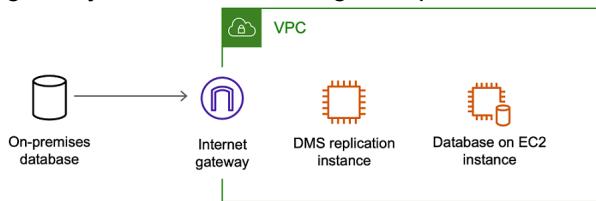
The following illustration shows a configuration where the source endpoint is an on-premises database in a corporate data center. It is connected by using AWS Direct Connect or a VPN to a VPC that contains the replication instance and a target database on an Amazon RDS DB instance.



In this configuration, the VPC security group must include a routing rule that sends traffic destined for a VPC CIDR range or specific IP address to a host. This host must be able to bridge traffic from the VPC into the on-premises VPN. In this case, the NAT host includes its own security group settings. These settings must allow traffic from the replication instance's VPC CIDR range, or private IP address, or security group into the NAT instance. But we do not recommend you use the private IP address of the replication instance, because it can break your replication if the replication IP address changes.

## Configuration for a network to a VPC using the internet

If you don't use a VPN or AWS Direct Connect to connect to AWS resources, you can use the internet to migrate your database. In this case, you can migrate to either an Amazon EC2 instance or an Amazon RDS DB instance. This configuration involves a public replication instance in a VPC with an internet gateway that contains the target endpoint and the replication instance.



To add an internet gateway to your VPC, see [Attaching an internet gateway](#) in the *Amazon VPC User Guide*.

The VPC route table must include routing rules that send traffic not destined for the VPC by default to the internet gateway. In this configuration, the connection to the endpoint appears to come from the public IP address of the replication instance, not the private IP address. For more information, see [VPC Route Tables](#) in the *Amazon VPC User Guide*.

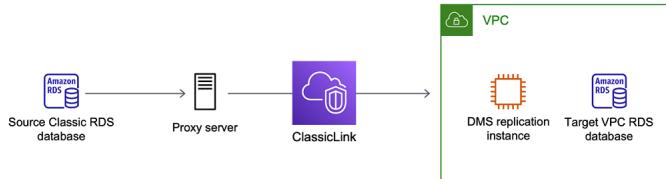
## Configuration with an RDS DB instance not in a VPC to a DB instance in a VPC using ClassicLink

We are retiring EC2-Classic on August 15, 2022. We recommend that you migrate from EC2-Classic to a VPC. For more information, see [Migrate from EC2-Classic to a VPC](#) in the *Amazon EC2 User Guide* and the blog [EC2-Classic Networking is Retiring – Here's How to Prepare](#).

To connect an Amazon RDS DB instance not in a VPC to a DMS replication server and DB instance in a VPC, you can use ClassicLink with a proxy server.

ClassicLink enables you to link an EC2-Classic DB instance to a VPC in your account, within the same AWS Region. After you've created the link, the source DB instance can communicate with the replication instance inside the VPC using their private IP addresses.

Because the replication instance in the VPC can't directly access the source DB instance on the EC2-Classic platform using ClassicLink, you use a proxy server. The proxy server connects the source DB instance to the VPC containing the replication instance and target DB instance. The proxy server uses ClassicLink to connect to the VPC. Port forwarding on the proxy server allows communication between the source DB instance and the target DB instance in the VPC.



## Using ClassicLink with AWS Database Migration Service

You can connect an Amazon RDS DB instance that is not in a VPC to an AWS DMS replication server and DB instance that are in a VPC. To do so, you can use Amazon EC2 ClassicLink with a proxy server.

The following procedure shows how to use ClassicLink for this purpose. This procedure connects an Amazon RDS source DB instance that is not in a VPC to a VPC containing an AWS DMS replication instance and a target DB instance.

- Create an AWS DMS replication instance in a VPC. (All replication instances are created in VPCs.)
- Associate a VPC security group to the replication instance and the target DB instance. When two instances share a VPC security group, they can communicate with each other by default.
- Set up a proxy server on an EC2 Classic instance.
- Create a connection using ClassicLink between the proxy server and the VPC.
- Create AWS DMS endpoints for the source and target databases.
- Create an AWS DMS task.

### To use ClassicLink to migrate a database on a DB instance not in a VPC to a database on a DB instance in a VPC

1. Create an AWS DMS replication instance and assign a VPC security group:

- a. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.

If you're signed in as an AWS Identity and Access Management (IAM) user, make sure that you have the appropriate permissions to access AWS DMS. For more information about the permissions required for database migration, see [IAM permissions needed to use AWS DMS \(p. 628\)](#).

- b. On the **Dashboard** page, choose **Replication Instance**. Follow the instructions at [Step 1: Create a replication instance using the AWS DMS console \(p. 24\)](#) to create a replication instance.
- c. After you have created the AWS DMS replication instance, open the EC2 service console. Click **Network Interfaces** from the navigation pane.
- d. Choose the **DMSNetworkInterface**, and then choose **Change Security Groups** from the **Actions** menu.
- e. Choose the security group you want to use for the replication instance and the target DB instance.

2. Associate the security group from the last step with the target DB instance:

- a. Open the Amazon RDS service console. Choose **Instances** from the navigation pane.
  - b. Choose the target DB instance. For **Instance Actions**, choose **Modify**.
  - c. For the **Security Group** parameter, choose the security group you used in the previous step.
  - d. Choose **Continue**, and then **Modify DB Instance**.
3. Step 3: Set up a proxy server on an EC2 Classic instance using NGINX. Use an AMI of your choice to launch an EC2 Classic instance. The example below is based on the AMI Ubuntu Server 14.04 LTS (HVM).

To set up a proxy server on an EC2 Classic instance

- a. Connect to the EC2 Classic instance and install NGINX using the following commands:

```
Prompt> sudo apt-get update
Prompt> sudo wget http://nginx.org/download/nginx-1.9.12.tar.gz
Prompt> sudo tar -xvzf nginx-1.9.12.tar.gz
Prompt> cd nginx-1.9.12
Prompt> sudo apt-get install build-essential
Prompt> sudo apt-get install libpcre3 libpcre3-dev
Prompt> sudo apt-get install zlib1g-dev
Prompt> ./configure --with-stream
Prompt> sudo make
Prompt> sudo make install
```

- b. Edit the NGINX daemon file, `/etc/init/nginx.conf`, using the following code:

```
# /etc/init/nginx.conf - Upstart file

description "nginx http daemon"
author "email"

start on (filesystem and net-device-up IFACE=lo)
stop on runlevel [!2345]

env DAEMON=/usr/local/nginx/sbin/nginx
env PID=/usr/local/nginx/logs/nginx.pid

expect fork
respawn
respawn limit 10 5

pre-start script
    $DAEMON -t
    if [ $? -ne 0 ]
        then exit $?
    fi
end script

exec $DAEMON
```

- c. Create an NGINX configuration file at `/usr/local/nginx/conf/nginx.conf`. In the configuration file, add the following:

```
# /usr/local/nginx/conf/nginx.conf - NGINX configuration file

worker_processes 1;
```

```
events {
    worker_connections 1024;
}

stream {
    server {
        listen DB instance port number;
        proxy_pass DB instance identifier:DB instance port number;
    }
}
```

- d. From the command line, start NGINX using the following commands:

```
Prompt> sudo initctl reload-configuration
Prompt> sudo initctl list | grep nginx
Prompt> sudo initctl start nginx
```

4. Create a ClassicLink connection between the proxy server and the target VPC that contains the target DB instance and the replication instance:
- Open the EC2 console and choose the EC2 Classic instance that is running the proxy server.
  - For **Actions**, choose **ClassicLink**, then choose **Link to VPC**.
  - Choose the security group that you used earlier in this procedure.
  - Choose **Link to VPC**.
5. Step 5: Create AWS DMS endpoints using the procedure at [Step 2: Specify source and target endpoints \(p. 26\)](#). Make sure to use the internal EC2 DNS hostname of the proxy as the server name when specifying the source endpoint.
6. Create an AWS DMS task using the procedure in [Step 3: Create a task and migrate data \(p. 28\)](#).

## Creating a replication subnet group

As part of the network to use for database migration, you need to specify what subnets in your virtual private cloud (VPC) that you plan to use. This VPC must be based on the Amazon VPC service. A *subnet* is a range of IP addresses in your VPC in a given Availability Zone. These subnets can be distributed among the Availability Zones for the AWS Region where your VPC is located.

You create a replication instance in a subnet that you choose, and you can manage what subnet a source or target endpoint uses by using the AWS DMS console.

You create a replication subnet group to define which subnets to use. You must specify subnets in at least two Availability Zones.

### To create a replication subnet group

- Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.

If you're signed in as an IAM user, make sure that you have the appropriate permissions to access AWS DMS. For more information about the permissions required for database migration, see [IAM permissions needed to use AWS DMS \(p. 628\)](#).

- In the navigation pane, choose **Subnet Groups**.
- Choose **Create Subnet Group**.

- On the **Edit Replication Subnet Group** page, shown following, specify your replication subnet group information. The following table describes the settings.

### Edit Replication Subnet Group

Available Subnets			Subnet Group		
AZ	Subnet	CIDR	AZ	Subnet	CIDR
<input type="checkbox"/>	subnet-4d802a3b	172.30.5.0/28			No records found.
<input checked="" type="checkbox"/>	subnet-6bec2f1d	172.30.0.0/24			
<input type="checkbox"/>	subnet-37b3566f	172.30.1.0/24			
<input checked="" type="checkbox"/>	subnet-3b945863	172.30.6.0/28			
<input checked="" type="checkbox"/>	subnet-d6bc43b3	172.30.2.0/24			
<input type="checkbox"/>	subnet-68ce7755	172.30.3.0/24			
<input type="checkbox"/>	subnet-b38b9f98	172.30.4.0/24			

Add Subnet(s) to this Subnet Group. You may add subnets one at a time or add all the subnets related to this VPC. You may make additions/edits after this group is created.

Add Remove Reset

Option	Action
<b>Identifier</b>	Enter a name for the replication subnet group that contains from 8 to 16 printable ASCII characters (excluding /, ", and @). The name should be unique for your account for the AWS Region you selected. You can choose to add some intelligence to the name such as including the AWS Region and task you are performing, for example <b>DMS-default-VPC</b> .
<b>Description</b>	Enter a brief description of the replication subnet group.
<b>VPC</b>	Choose the VPC you want to use for database migration. Keep in mind that the VPC must have at least one subnet in at least two Availability Zones.
<b>Available Subnets</b>	Choose the subnets you want to include in the replication subnet group. You must choose subnets in at least two Availability Zones.

- Choose **Add** to add the subnets to the replication subnet group.
- Choose **Create**.

## Resolving domain endpoints using DNS

Usually, an AWS DMS replication instance uses the Domain Name System (DNS) resolver in an Amazon EC2 instance to resolve domain endpoints. If you require DNS resolution, you can use the Amazon

Route 53 Resolver. For more information about using Route 53 DNS Resolver, see [Getting started with Route 53 Resolver](#).

For information about how to use your own on-premises name server to resolve certain endpoints using the Amazon Route 53 Resolver, see [Using your own on-premises name server \(p. 90\)](#).

## Setting an encryption key for a replication instance

AWS DMS encrypts the storage used by a replication instance and the endpoint connection information. To encrypt the storage used by a replication instance, AWS DMS uses a AWS KMS key that is unique to your AWS account. You can view and manage this KMS key with AWS Key Management Service (AWS KMS). You can use the default KMS key in your account (aws/dms) or a KMS key that you create. If you have an existing AWS KMS encryption key, you can also use that key for encryption.

You can specify your own encryption key by supplying a KMS key identifier to encrypt your AWS DMS resources. When you specify your own encryption key, the user account used to perform the database migration must have access to that key. For more information on creating your own encryption keys and giving users access to an encryption key, see the [AWS KMS Developer Guide](#).

If you don't specify a KMS key identifier, then AWS DMS uses your default encryption key. KMS creates the default encryption key for AWS DMS for your AWS account. Your AWS account has a different default encryption key for each AWS Region.

To manage the keys used for encrypting your AWS DMS resources, you use AWS KMS. You can find AWS KMS in the AWS Management Console by searching for **KMS** on the navigation pane.

AWS KMS combines secure, highly available hardware and software to provide a key management system scaled for the cloud. Using AWS KMS, you can create encryption keys and define the policies that control how these keys can be used. AWS KMS supports AWS CloudTrail, so you can audit key usage to verify that keys are being used appropriately. Your AWS KMS keys can be used in combination with AWS DMS and other supported AWS services. Supported AWS services include Amazon RDS, Amazon S3, Amazon Elastic Block Store (Amazon EBS), and Amazon Redshift.

When you have created your AWS DMS resources with a specific encryption key, you can't change the encryption key for those resources. Make sure to determine your encryption key requirements before you create your AWS DMS resources.

## Creating a replication instance

Your first task in migrating a database is to create a replication instance. This replication instance requires sufficient storage and processing power to perform the tasks that you assign and migrate data from your source database to the target database. The required size of this instance varies depending on the amount of data you need to migrate and the tasks that you need the instance to perform. For more information about replication instances, see [Working with an AWS DMS replication instance \(p. 100\)](#).

### To create a replication instance by using the AWS console

1. Choose **Replication instances** in the navigation pane of the AWS DMS console and then choose **Create replication instance**.
2. On the **Create replication instance** page, specify your replication instance information. The following table describes the settings you can make.

Option	Action
<b>Name</b>	Enter a name for the replication instance that contains from 8 to 16 printable ASCII characters (excluding /, ;, and @). The name should be unique for your account for the AWS Region you selected. You can choose to add some intelligence to the name, such as including the AWS Region and task you are performing, for example <b>west2-mysql2mysql-instance1</b> .
<b>Descriptive Amazon Resource Name (ARN) - <i>Optional</i></b>	A friendly name to override the default DMS ARN. You can't modify it after creation.
<b>Description</b>	Enter a brief description of the replication instance.
<b>Instance class</b>	Choose an instance class with the configuration you need for your migration. Keep in mind that the instance must have enough storage, network, and processing power to successfully complete your migration. For more information on how to determine which instance class is best for your migration, see <a href="#">Working with an AWS DMS replication instance (p. 100)</a> .
<b>Engine version</b>	In the AWS DMS console, you can choose any supported engine version that you want. From the AWS CLI, the replication instance runs the latest non-beta version of the AWS DMS replication engine unless you specify a different engine version in the AWS CLI.
<b>Multi-AZ</b>	Use this optional parameter to create a standby replica of your replication instance in another Availability Zone for failover support. If you intend to use change data capture (CDC) or ongoing replication, you should turn on this option.

Option	Action
<b>Allocated storage (GiB)</b>	<p>Storage is primarily consumed by log files and cached transactions. For caches transactions, storage is used only when the cached transactions need to be written to disk. Therefore, AWS DMS doesn't use a significant amount of storage. Some exceptions include the following:</p> <ul style="list-style-type: none"> <li>Very large tables that incur a significant transaction load. Loading a large table can take some time, so cached transactions are more likely to be written to disk during a large table load.</li> <li>Tasks that are configured to pause before loading cached transactions. In this case, all transactions are cached until the full load completes for all tables. With this configuration, a fair amount of storage might be consumed by cached transactions.</li> <li>Tasks configured with tables being loaded into Amazon Redshift. However, this configuration isn't an issue when Amazon Aurora is the target.</li> </ul> <p>In most cases, the default allocation of storage is sufficient. However, it's always a good idea to pay attention to storage-related metrics. Make sure to scale up your storage if you find you are consuming more than the default allocation.</p>
<b>Network type</b>	DMS supports the <b>IPv4</b> addressing protocol network type, and supports both IPv4 and IPv6 addressing protocol network types in <b>Dual-stack</b> mode. When you have resources that must communicate with your replication instance using an IPv6 addressing protocol network type, use <b>Dual-stack</b> mode.
<b>VPC</b>	Choose the VPC that you want to use. If your source or your target database is in a VPC, choose that VPC. If your source and your target databases are in different VPCs, ensure that they are both in public subnets and are publicly accessible. Then choose the VPC where the replication instance is to be located. The replication instance must be able to access the data in the source VPC. If neither your source or target database is in a VPC, choose a VPC where the replication instance is to be located.
<b>Replication Subnet Group</b>	Choose the replication subnet group in your selected VPC where you want the replication instance to be created. If your source database is in a VPC, choose the subnet group that contains the source database as the location for your replication instance. For more information about replication subnet groups, see <a href="#">Creating a replication subnet group (p. 117)</a> .
<b>Publicly accessible</b>	Choose this option if you want the replication instance to be accessible from the internet.

3. Choose the **Advanced** tab to set values for network and encryption settings if you need them. The following table describes the settings.

Option	Action
<b>Availability zone</b>	Choose the Availability Zone where your source database is located.
<b>VPC Security group(s)</b>	The replication instance is created in a VPC. If your source database is in a VPC, choose the VPC security group that provides access to the DB instance where the database resides.
<b>KMS key</b>	Choose the encryption key to use to encrypt replication storage and connection information. If you choose <b>(Default) aws/dms</b> , the default AWS Key Management Service (AWS KMS) key associated with your account and AWS Region is used. A description and your account number are shown, along with the key's ARN. For more information on using the encryption key, see <a href="#">Setting an encryption key and specifying AWS KMS permissions (p. 643)</a> .

4. Specify the **Maintenance** settings. The following table describes the settings. For more information about maintenance settings, see [Working with the AWS DMS maintenance window \(p. 128\)](#).

Option	Action
<b>Auto minor version upgrade</b>	<p>Choose to have minor engine upgrades applied automatically to the replication instance during the maintenance window.</p> <p>When <b>Auto minor version upgrade</b> is enabled, DMS uses the current default engine version when you create a replication instance. For example, if you set the replication <b>Engine version</b> to a lower version number than the current default version, DMS uses the default version.</p> <p>If <b>Auto minor version upgrade</b> isn't enabled when you create a replication instance, DMS uses the engine version specified by <b>Engine version</b>.</p>
<b>Maintenance window</b>	<p>Choose a weekly time range during which system maintenance can occur, in Universal Coordinated Time (UTC).</p> <p>Default: A 30-minute window selected at random from an 8-hour block of time per AWS Region, occurring on a random day of the week.</p>

5. Choose **Create replication instance**.

# Modifying a replication instance

You can modify the settings for a replication instance to, for example, change the instance class or to increase storage.

When you modify a replication instance, you can apply the changes immediately. To apply changes immediately, choose the **Apply changes immediately** option in the AWS Management Console. Or use the `--apply-immediately` parameter when calling the AWS CLI, or set the `ApplyImmediately` parameter to `true` when using the DMS API.

If you don't choose to apply changes immediately, the changes are put into the pending modifications queue. During the next maintenance window, any pending changes in the queue are applied.

## Note

If you choose to apply changes immediately, any changes in the pending modifications queue are also applied. If any of the pending modifications require downtime, choosing **Apply changes immediately** can cause unexpected downtime.

## To modify a replication instance by using the AWS console

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. In the navigation pane, choose **Replication instances**.
3. Choose the replication instance you want to modify. The following table describes the modifications you can make.

Option	Action
<b>Name</b>	You can change the name of the replication instance. Enter a name for the replication instance that contains from 8 to 16 printable ASCII characters (excluding /, , and @). The name should be unique for your account for the AWS Region you selected. You can choose to add some intelligence to the name, such as including the AWS Region and task you are performing, for example <b>west2-mysql2mysql-instance1</b> .
<b>Description</b>	Revise or enter a brief description of the replication instance.
<b>Instance class</b>	You can change the instance class. Choose an instance class with the configuration you need for your migration. Changing the instance class causes the replication instance to reboot. This reboot occurs during the next maintenance window or can occur immediately if you choose the <b>Apply changes immediately</b> option.  For more information on how to determine which instance class is best for your migration, see <a href="#">Working with an AWS DMS replication instance (p. 100)</a> .
<b>Engine version</b>	You can upgrade the engine version that is used by the replication instance. Upgrading the replication engine version causes the replication instance to shut down while it is being upgraded.

Option	Action
<b>Multi-AZ</b>	You can change this option to create a standby replica of your replication instance in another Availability Zone for failover support or remove this option. If you intend to use change data capture (CDC), ongoing replication, you should enable this option.
<b>Allocated storage (GiB)</b>	<p>Storage is primarily consumed by log files and cached transactions. For caches transactions, storage is used only when the cached transactions need to be written to disk. Therefore, AWS DMS doesn't use a significant amount of storage. Some exceptions include the following:</p> <ul style="list-style-type: none"> <li>• Very large tables that incur a significant transaction load. Loading a large table can take some time, so cached transactions are more likely to be written to disk during a large table load.</li> <li>• Tasks that are configured to pause before loading cached transactions. In this case, all transactions are cached until the full load completes for all tables. With this configuration, a fair amount of storage might be consumed by cached transactions.</li> <li>• Tasks configured with tables being loaded into Amazon Redshift. However, this configuration isn't an issue when Amazon Aurora is the target.</li> </ul> <p>In most cases, the default allocation of storage is sufficient. However, it's always a good idea to pay attention to storage related metrics and scale up your storage if you find you are consuming more than the default allocation.</p>
<b>Network type</b>	DMS supports the <b>IPv4</b> addressing protocol network type, and supports both IPv4 and IPv6 addressing protocol network types in <b>Dual-stack</b> mode. When you have resources that must communicate to your replication instance using an IPv6 addressing protocol network type, choose <b>Dual-stack</b> mode.
<b>VPC Security Group(s)</b>	The replication instance is created in a VPC. If your source database is in a VPC, choose the VPC security group that provides access to the DB instance where the database resides.

Option	Action
<b>Auto minor version upgrade</b>	<p>Choose this option to have minor engine upgrades applied automatically to the replication instance during the maintenance window or immediately if you choose the <b>Apply changes immediately</b> option.</p> <p>When <b>Auto minor version upgrade</b> is enabled, DMS uses the current default engine version when you modify a replication instance. For example, if you set <b>Engine version</b> to a lower version number than the current default version, DMS uses the default version.</p> <p>If <b>Auto minor version upgrade</b> isn't enabled when you modify a replication instance, DMS uses the engine version specified by <b>Engine version</b>.</p>
<b>Maintenance window</b>	<p>Choose a weekly time range during which system maintenance can occur, in Universal Coordinated Time (UTC).</p> <p>Default: A 30-minute window selected at random from an 8-hour block of time per AWS Region, occurring on a random day of the week.</p>
<b>Apply changes immediately</b>	Choose this option to apply any modifications you made immediately. Depending on the settings you choose, choosing this option could cause an immediate reboot of the replication instance.
<b>Apply changes during next scheduled maintenance window</b>	Choose this option if you want DMS to wait until the next scheduled maintenance window to apply your changes.

## Rebooting a replication instance

You can reboot an AWS DMS replication instance to restart the replication engine. A reboot results in a momentary outage for the replication instance, during which the instance status is set to **Rebooting**. If the AWS DMS instance is configured for Multi-AZ, the reboot can be conducted with a failover. An AWS DMS event is created when the reboot is completed.

If your AWS DMS instance is a Multi-AZ deployment, you can force a planned failover from one AWS Availability Zone to another when you reboot. When you force a planned failover of your AWS DMS instance, AWS DMS closes out active connections on the current instance prior to automatically switching to a standby instance in another Availability Zone. Rebooting with a planned failover helps you simulate a planned failover event of an AWS DMS instance, such as when scaling the replication instance class.

### Note

After a reboot forces a failover from one Availability Zone to another, the Availability Zone change might not be reflected for several minutes. This lag appears in the AWS Management Console, and in calls to the AWS CLI and AWS DMS API.

If migration tasks are running on the replication instance when a reboot occurs, no data loss occurs but the task stops, and the task status changes to an error state.

If the tables in the migration task are in the middle of a bulk load (full load phase) and haven't yet started, they go into an error state. But tables that are complete at that time, remain in a complete state.

When a reboot happens during the full load phase, we recommended that you perform either one of the steps below.

- Remove the tables that are in a complete state from the task, and restart the task with the remaining tables.
- Create a new task with tables in an error state, and with tables that are pending.

If tables in the migration task are in the ongoing replication phase, the task resumes once the reboot is completed.

You can't reboot your AWS DMS replication instance if its status is not in the **Available** state. Your AWS DMS instance can be unavailable for several reasons, such as a previously requested modification or a maintenance-window action. The time required to reboot an AWS DMS replication instance is typically small (under 5 minutes).

## Rebooting a replication instance using the AWS console

To reboot a replication instance, use the AWS console.

### To reboot a replication instance using the AWS console

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. In the navigation pane, choose **Replication instances**.
3. Choose the replication instance you want to reboot.
4. Choose **Reboot**.
5. In the **Reboot replication instance** dialog box, choose **Reboot With Failover?** if you have configured your replication instance for Multi-AZ deployment and you want to fail over to another AWS Availability Zone.
6. Choose **Reboot**.

## Rebooting a replication instance using the CLI

To reboot a replication instance, use the AWS CLI `reboot-replication-instance` command with the following parameter:

- `--replication-instance-arn`

### Example Example simple reboot

The following AWS CLI example reboots a replication instance.

```
aws dms reboot-replication-instance \
--replication-instance-arn arn of my rep instance
```

### Example Example simple reboot with failover

The following AWS CLI example reboots a replication instance with failover.

```
aws dms reboot-replication-instance \
--replication-instance-arn arn of my rep instance \
--force-failover
```

## Rebooting a replication instance using the API

To reboot a replication instance, use the AWS DMS API [RebootReplicationInstance](#) action with the following parameters:

- ReplicationInstanceArn = *arn of my rep instance*

### Example Example simple reboot

The following code example reboots a replication instance.

```
https://dms.us-west-2.amazonaws.com/  
?Action=RebootReplicationInstance  
&DBInstanceArn=arn of my rep instance  
&SignatureMethod=HmacSHA256  
&SignatureVersion=4  
&Version=2014-09-01  
&X-Amz-Algorithm=AWS4-HMAC-SHA256  
&X-Amz-Credential=AKIADQKE4SARGYLE/20140425/us-east-1/dms/aws4_request  
&X-Amz-Date=20140425T192732Z  
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date  
&X-Amz-Signature=1dc9dd716f4855e9bdf188c70f1cf9f6251b070b68b81103b59ec70c3e7854b3
```

### Example Example simple reboot with failover

The following code example reboots a replication instance and fails over to another AWS Availability Zone.

```
https://dms.us-west-2.amazonaws.com/  
?Action=RebootReplicationInstance  
&DBInstanceArn=arn of my rep instance  
&ForceFailover=true  
&SignatureMethod=HmacSHA256  
&SignatureVersion=4  
&Version=2014-09-01  
&X-Amz-Algorithm=AWS4-HMAC-SHA256  
&X-Amz-Credential=AKIADQKE4SARGYLE/20140425/us-east-1/dms/aws4_request  
&X-Amz-Date=20140425T192732Z  
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date  
&X-Amz-Signature=1dc9dd716f4855e9bdf188c70f1cf9f6251b070b68b81103b59ec70c3e7854b3
```

## Deleting a replication instance

You can delete an AWS DMS replication instance when you are finished using it. If you have migration tasks that use the replication instance, you must stop and delete the tasks before deleting the replication instance.

If you close your AWS account, all AWS DMS resources and configurations associated with your account are deleted after two days. These resources include all replication instances, source and target endpoint configuration, replication tasks, and SSL certificates. If after two days you decide to use AWS DMS again, you recreate the resources you need.

## Deleting a replication instance using the AWS console

To delete a replication instance, use the AWS console.

## To delete a replication instance using the AWS console

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. In the navigation pane, choose **Replication instances**.
3. Choose the replication instance you want to delete.
4. Choose **Delete**.
5. In the dialog box, choose **Delete**.

## Deleting a replication instance using the CLI

To delete a replication instance, use the AWS CLI `delete-replication-instance` command with the following parameter:

- `--replication-instance-arn`

### Example Example delete

The following AWS CLI example deletes a replication instance.

```
aws dms delete-replication-instance \
--replication-instance-arn arn of my rep instance
```

## Deleting a replication instance using the API

To delete a replication instance, use the AWS DMS API `DeleteReplicationInstance` action with the following parameters:

- `ReplicationInstanceArn = arn of my rep instance`

### Example Example delete

The following code example deletes a replication instance.

```
https://dms.us-west-2.amazonaws.com/
?Action=DeleteReplicationInstance
&DBInstanceArn=arn of my rep instance
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-09-01
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE/20140425/us-east-1/dms/aws4_request
&X-Amz-Date=20140425T192732Z
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
&X-Amz-Signature=1dc9dd716f4855e9bdf188c70f1cf9f6251b070b68b81103b59ec70c3e7854b3
```

# Working with the AWS DMS maintenance window

Every AWS DMS replication instance has a weekly maintenance window during which any available system changes are applied. You can think of the maintenance window as an opportunity to control when modifications and software patching occurs.

If AWS DMS determines that maintenance is required during a given week, the maintenance occurs during the 30-minute maintenance window you chose when you created the replication instance. AWS DMS completes most maintenance during the 30-minute maintenance window. However, a longer time might be required for larger changes.

## Effect of maintenance on existing migration tasks

When an AWS DMS migration task is running on an instance, the following events occur when a patch is applied:

- If the tables in the migration task are in the replicating ongoing changes phase (CDC), AWS DMS stops the task for a moment and then resumes it after the patch is applied. The migration then continues from where it was interrupted when the patch was applied.
- If AWS DMS is migrating a table as part of a **migrate existing data** or **migrate existing data and replicate ongoing changes** task, DMS stops and then restarts the migration for all tables that are in full load phase while the patch is applied. DMS also stops and resumes all tables that are in CDC phase while the patch is applied.

## Changing the maintenance window setting

You can change the maintenance window time frame using the AWS Management Console, the AWS CLI, or the AWS DMS API.

### Changing the maintenance window setting using the console

You can change the maintenance window time frame using the AWS Management Console.

#### To change the preferred maintenance window using the console

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. In the navigation pane, choose **Replication instances**.
3. Choose the replication instance you want to modify and choose **Modify**.
4. Expand the **Maintenance** tab and choose a date and time for your maintenance window.
5. Choose **Apply changes immediately**.
6. Choose **Modify**.

### Changing the maintenance window setting using the CLI

To adjust the preferred maintenance window, use the AWS CLI `modify-replication-instance` command with the following parameters.

- `--replication-instance-identifier`
- `--preferred-maintenance-window`

#### Example

The following AWS CLI example sets the maintenance window to Tuesdays from 4:00–4:30 a.m. UTC.

```
aws dms modify-replication-instance \
--replication-instance-identifier myrepinstance \
--preferred-maintenance-window Tue:04:00-Tue:04:30
```

## Changing the maintenance window setting using the API

To adjust the preferred maintenance window, use the AWS DMS API [ModifyReplicationInstance](#) action with the following parameters.

- ReplicationInstanceIdentifier = *myrepinstance*
- PreferredMaintenanceWindow = *Tue:04:00-Tue:04:30*

### Example

The following code example sets the maintenance window to Tuesdays from 4:00–4:30 a.m. UTC.

```
https://dms.us-west-2.amazonaws.com/  
?Action=ModifyReplicationInstance  
&DBInstanceIdentifier=myrepinstance  
&PreferredMaintenanceWindow=Tue:04:00-Tue:04:30  
&SignatureMethod=HmacSHA256  
&SignatureVersion=4  
&Version=2014-09-01  
&X-Amz-Algorithm=AWS4-HMAC-SHA256  
&X-Amz-Credential=AKIADQKE4SARGYLE/20140425/us-east-1/dms/aws4_request  
&X-Amz-Date=20140425T192732Z  
&X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date  
&X-Amz-Signature=1dc9dd716f4855e9bdf188c70f1cf9f6251b070b68b81103b59ec70c3e7854b3
```

# Working with AWS DMS endpoints

An endpoint provides connection, data store type, and location information about your data store. AWS Database Migration Service uses this information to connect to a data store and migrate data from a source endpoint to a target endpoint. You can specify additional connection attributes for an endpoint by using extra connection attributes. These attributes can control logging, file size, and other parameters; for more information about extra connection attributes, see the documentation section for your data store.

Following, you can find out more details about endpoints.

## Topics

- [Creating source and target endpoints \(p. 131\)](#)
- [Sources for data migration \(p. 134\)](#)
- [Targets for data migration \(p. 293\)](#)
- [Configuring VPC endpoints as AWS DMS source and target endpoints \(p. 437\)](#)
- [DDL statements supported by AWS DMS \(p. 438\)](#)

## Creating source and target endpoints

You can create source and target endpoints when you create your replication instance or you can create endpoints after your replication instance is created. The source and target data stores can be on an Amazon Elastic Compute Cloud (Amazon EC2) instance, an Amazon Relational Database Service (Amazon RDS) DB instance, or an on-premises database. (Note that one of your endpoints must be on an AWS service. You can't use AWS DMS to migrate from an on-premises database to another on-premises database.)

The following procedure assumes that you have chosen the AWS DMS console wizard. Note that you can also do this step by selecting **Endpoints** from the AWS DMS console's navigation pane and then selecting **Create endpoint**. When using the console wizard, you create both the source and target endpoints on the same page. When not using the console wizard, you create each endpoint separately.

### To specify source or target database endpoints using the AWS console

1. On the **Connect source and target database endpoints** page, specify your connection information for the source or target database. The following table describes the settings.

For this option	Do this
<b>Endpoint type</b>	Choose whether this endpoint is the source or target endpoint.
<b>Select RDS DB Instance</b>	Choose this option if the endpoint is an Amazon RDS DB instance.
<b>Endpoint identifier</b>	Type the name you want to use to identify the endpoint. You might want to include in the name the type of endpoint, such as <b>oracle-source</b> or <b>PostgreSQL-target</b> . The name must be unique for all replication instances.
<b>Source engine and Target engine</b>	Choose the type of database engine that is the endpoint.

For this option	Do this
<b>Access to endpoint database</b>	<p>Choose the option you want to use to specify endpoint database credentials:</p> <ul style="list-style-type: none"> <li>• <a href="#">Choose AWS Secrets Manager (p. 132)</a> – Use secrets defined in AWS Secrets Manager to secretly provide your credentials as shown following. For more information on creating these secrets and the secret access roles that enable AWS DMS to access them, see <a href="#">Using secrets to access AWS Database Migration Service endpoints (p. 615)</a>.</li> <li>• <a href="#">Provide access information manually (p. 132)</a> – Use clear-text credentials that you enter directly as shown following.</li> </ul>
<b>Choose AWS Secrets Manager</b>	Set the following secret credentials.
<b>Secret ID</b>	Type the full Amazon Resource Name (ARN), partial ARN, or friendly name of a secret that you have created in the AWS Secrets Manager for endpoint database access.
<b>IAM role</b>	Type the ARN of a secret access role that you have created in IAM to provide AWS DMS access on your behalf to the secret identified by <b>Secret ID</b> .
<b>Secret ID for Oracle automatic storage management (ASM)</b>	(For Oracle source endpoints using Oracle ASM only) Type the full Amazon Resource Name (ARN), partial ARN, or friendly name of a secret that you have created in the AWS Secrets Manager for Oracle ASM access. This secret is typically created to access Oracle ASM on the same server as the secret identified by <b>Secret ID</b> .
<b>IAM role for Oracle ASM</b>	(For Oracle source endpoints using Oracle ASM only) Type the ARN of a secret access role that you have created in IAM to provide AWS DMS access on your behalf to the secret identified by <b>Secret ID for Oracle automatic storage management (ASM)</b> .
<b>Provide access information manually</b>	Set the following clear-text credentials.
<b>Server name</b>	Type the server name. For an on-premises database, this can be the IP address or the public hostname. For an Amazon RDS DB instance, this can be the endpoint (also called the DNS name) for the DB instance, such as <code>mysqlsvrinst.abcd12345678.us-west-2.rds.amazonaws.com</code> .
<b>Port</b>	Type the port used by the database.
<b>Secure Socket Layer (SSL) mode</b>	Choose an SSL mode if you want to enable connection encryption for this endpoint. Depending on the mode you select, you might be asked to provide certificate and server certificate information.
<b>User name</b>	Type the user name with the permissions required to allow data migration. For information on the permissions required, see the security section for the source or target database engine in this user guide.

For this option	Do this
<b>Password</b>	Type the password for the account with the required permissions. Passwords for AWS DMS source and target endpoints have character restrictions, depending on the database engine. For more information, see the following table.
<b>Database name</b>	For certain database engines, the name of the database you want to use as the endpoint database.

The following table lists the unsupported characters in endpoint passwords for the listed database engines. If you want to use commas (,) in your endpoint passwords, use the Secrets Manager support provided in AWS DMS to authenticate access to your AWS DMS instances. For more information, see [Using secrets to access AWS Database Migration Service endpoints \(p. 615\)](#).

For this database engine	The following characters are unsupported in an endpoint password
Microsoft Azure, as a source only	;
Microsoft SQL Server	, ;
MySQL-compatible, including MySQL, MariaDB, and Amazon Aurora MySQL	;
Oracle	,
PostgreSQL, Amazon Aurora PostgreSQL-Compatible Edition, and Amazon Aurora Serverless as a target only for Aurora PostgreSQL-Compatible Edition	; + %
Amazon Redshift, as a target only	, ;

- Choose the **Advanced** tab to set values for **Extra connection attributes** and **AWS KMS key** if you need them. You can test the endpoint connection by choosing **Run test**. The following table describes the settings.

For this option	Do this
<b>Extra connection attributes</b>	<p>Type any additional connection parameters here. For more information about extra connection attributes, see the documentation section for your <b>Source engine</b> or <b>Target engine</b> (specified in step 1).</p> <p>For an Oracle source endpoint that uses Oracle ASM, if you choose <b>Provide access information manually</b> in step 1, you might also need to type in extra connection attributes to specify Oracle ASM user credentials. For more information on these Oracle ASM extra connection attributes, see <a href="#">Using Oracle LogMiner or AWS DMS Binary Reader for CDC (p. 135)</a>.</p>

For this option	Do this
<b>AWS KMS key</b>	Choose the encryption key to use to encrypt replication storage and connection information. If you choose <b>(Default) aws/dms</b> , the default AWS Key Management Service (AWS KMS) key associated with your account and AWS Region is used. For more information on using the encryption key, see <a href="#">Setting an encryption key and specifying AWS KMS permissions (p. 643)</a> .
<b>Test endpoint connection (optional)</b>	Add the VPC and replication instance name. To test the connection, choose <b>Run test</b> .

## Sources for data migration

AWS Database Migration Service (AWS DMS) can use many of the most popular data engines as a source for data replication. The database source can be a self-managed engine running on an Amazon EC2 instance or an on-premises database. Or it can be a data source on an AWS service such as Amazon RDS or Amazon S3.

For a comprehensive list of valid sources, see [Sources for AWS DMS \(p. 134\)](#).

### Topics

- [Using an Oracle database as a source for AWS DMS \(p. 134\)](#)
- [Using a Microsoft SQL Server database as a source for AWS DMS \(p. 175\)](#)
- [Using Microsoft Azure SQL database as a source for AWS DMS \(p. 195\)](#)
- [Using Microsoft Azure SQL Managed Instance as a source for AWS DMS \(p. 195\)](#)
- [Using Google Cloud for MySQL as a source for AWS DMS \(p. 195\)](#)
- [Using a PostgreSQL database as an AWS DMS source \(p. 195\)](#)
- [Using a MySQL-compatible database as a source for AWS DMS \(p. 216\)](#)
- [Using an SAP ASE database as a source for AWS DMS \(p. 225\)](#)
- [Using MongoDB as a source for AWS DMS \(p. 231\)](#)
- [Using Amazon DocumentDB \(with MongoDB compatibility\) as a source for AWS DMS \(p. 242\)](#)
- [Using Amazon S3 as a source for AWS DMS \(p. 254\)](#)
- [Using IBM Db2 for Linux, Unix, and Windows database \(Db2 LUW\) as a source for AWS DMS \(p. 261\)](#)
- [Using IBM Db2 for z/OS databases as a source for AWS DMS \(p. 265\)](#)

## Using an Oracle database as a source for AWS DMS

You can migrate data from one or many Oracle databases using AWS DMS. With an Oracle database as a source, you can migrate data to any of the targets supported by AWS DMS.

AWS DMS supports the following Oracle database editions:

- Oracle Enterprise Edition
- Oracle Standard Edition
- Oracle Express Edition

- Oracle Personal Edition

For self-managed Oracle databases, AWS DMS supports all Oracle database editions for versions 10.2 and later (for versions 10.x), 11g and up to 12.2, 18c, and 19c. For Amazon RDS for Oracle databases that AWS manages, AWS DMS supports all Oracle database editions for versions 11g (versions 11.2.0.4 and later) and up to 12.2, 18c, and 19c.

You can use Secure Sockets Layer (SSL) to encrypt connections between your Oracle endpoint and your replication instance. For more information on using SSL with an Oracle endpoint, see [SSL support for an Oracle endpoint \(p. 154\)](#). AWS DMS also supports the use of Oracle transparent data encryption (TDE) to encrypt data at rest in the source database. For more information on using Oracle TDE with an Oracle source endpoint, see [Supported encryption methods for using Oracle as a source for AWS DMS \(p. 159\)](#).

Follow these steps to configure an Oracle database as an AWS DMS source endpoint:

1. Create an Oracle user with the appropriate permissions for AWS DMS to access your Oracle source database.
2. Create an Oracle source endpoint that conforms with your chosen Oracle database configuration. To create a full-load-only task, no further configuration is needed.
3. To create a task that handles change data capture (a CDC-only or full-load and CDC task), choose Oracle LogMiner or AWS DMS Binary Reader to capture data changes. Choosing LogMiner or Binary Reader determines some of the later permissions and configuration options. For a comparison of LogMiner and Binary Reader, see the following section.

#### Note

For more information on full-load tasks, CDC-only tasks, and full-load and CDC tasks, see [Creating a task \(p. 442\)](#)

For additional details on working with Oracle source databases and AWS DMS, see the following sections.

#### Topics

- [Using Oracle LogMiner or AWS DMS Binary Reader for CDC \(p. 135\)](#)
- [Workflows for configuring a self-managed or AWS-managed Oracle source database for AWS DMS \(p. 138\)](#)
- [Working with a self-managed Oracle database as a source for AWS DMS \(p. 139\)](#)
- [Working with an AWS-managed Oracle database as a source for AWS DMS \(p. 147\)](#)
- [Limitations on using Oracle as a source for AWS DMS \(p. 152\)](#)
- [SSL support for an Oracle endpoint \(p. 154\)](#)
- [Supported encryption methods for using Oracle as a source for AWS DMS \(p. 159\)](#)
- [Supported compression methods for using Oracle as a source for AWS DMS \(p. 162\)](#)
- [Replicating nested tables using Oracle as a source for AWS DMS \(p. 163\)](#)
- [Storing REDO on Oracle ASM when using Oracle as a source for AWS DMS \(p. 164\)](#)
- [Extra connection attributes when using Oracle as a source for AWS DMS \(p. 165\)](#)
- [Source data types for Oracle \(p. 172\)](#)

## Using Oracle LogMiner or AWS DMS Binary Reader for CDC

In AWS DMS, there are two methods for reading the redo logs when doing change data capture (CDC) for Oracle as a source: Oracle LogMiner and AWS DMS Binary Reader. LogMiner is an Oracle API to read the online redo logs and archived redo log files. Binary Reader is an AWS DMS method that reads and parses the raw redo log files directly. These methods have the following features.

Feature	LogMiner	Binary Reader
Easy to configure	Yes	No
Lower impact on source system I/O and CPU	No	Yes
Better CDC performance	No	Yes
Supports Oracle table clusters	Yes	No
Supports all types of Oracle Hybrid Columnar Compression (HCC)	Yes	Partially  Binary Reader doesn't support QUERY LOW for tasks with CDC. All other HCC types are fully supported.
LOB column support in Oracle 12c only	No	Yes
Supports UPDATE statements that affect only LOB columns	No	Yes
Supports Oracle transparent data encryption (TDE)	Partially  When using Oracle LogMiner, AWS DMS doesn't support TDE encryption on column level for Amazon RDS for Oracle.	Partially  Binary Reader supports TDE only for self-managed Oracle databases.
Supports all Oracle compression methods	Yes	No
Supports XA transactions	No	Yes
RAC	Yes  Not recommended	Yes  Highly recommended

#### Note

By default, AWS DMS uses Oracle LogMiner for (CDC).

The main advantages of using LogMiner with AWS DMS include the following:

- LogMiner supports most Oracle options, such as encryption options and compression options. Binary Reader doesn't support all Oracle options, particularly compression and most options for encryption.
- LogMiner offers a simpler configuration, especially compared to Binary Reader direct-access setup or when the redo logs are managed using Oracle Automatic Storage Management (ASM).
- LogMiner supports table clusters for use by AWS DMS. Binary Reader doesn't.

The main advantages of using Binary Reader with AWS DMS include the following:

- For migrations with a high volume of changes, LogMiner might have some I/O or CPU impact on the computer hosting the Oracle source database. Binary Reader has less chance of having I/O or CPU impact because logs are mined directly rather than making multiple database queries.
- For migrations with a high volume of changes, CDC performance is usually much better when using Binary Reader compared with using Oracle LogMiner.
- Binary Reader supports CDC for LOBs in Oracle version 12c. LogMiner doesn't.

In general, use Oracle LogMiner for migrating your Oracle database unless you have one of the following situations:

- You need to run several migration tasks on the source Oracle database.
- The volume of changes or the redo log volume on the source Oracle database is high, or you have changes and are also using Oracle ASM.

**Note**

If you change between using Oracle LogMiner and AWS DMS Binary Reader, make sure to restart the CDC task.

## Configuration for CDC on an Oracle source database

For an Oracle source endpoint to connect to the database for a change data capture (CDC) task, you might need to specify extra connection attributes. This can be true for either a full-load and CDC task or for a CDC-only task. The extra connection attributes that you specify depend on the method you use to access the redo logs: Oracle LogMiner or AWS DMS Binary Reader.

You specify extra connection attributes when you create a source endpoint. If you have multiple connection attribute settings, separate them from each other by semicolons with no additional white space (for example, oneSetting;thenAnother).

AWS DMS uses LogMiner by default. You don't have to specify additional extra connection attributes to use it.

To use Binary Reader to access the redo logs, add the following extra connection attributes.

```
useLogMinerReader=N;useBfile=Y;
```

Use the following format for the extra connection attributes to access a server that uses ASM with Binary Reader.

```
useLogMinerReader=N;useBfile=Y;asm_user=asm_username;asm_server=RAC_server_ip_address:port_number;  
+ASM;
```

Set the source endpoint Password request parameter to both the Oracle user password and the ASM password, separated by a comma as follows.

```
oracle_user_password,asm_user_password
```

Where the Oracle source uses ASM, you can work with high-performance options in Binary Reader for transaction processing at scale. These options include extra connection attributes to specify the number of parallel threads (`parallelASMReadThreads`) and the number of read-ahead buffers (`readAheadBlocks`). Setting these attributes together can significantly improve the performance of the CDC task. The following settings provide good results for most ASM configurations.

```
useLogMinerReader=N;useBfile=Y;asm_user=asm_username;asm_server=RAC_server_ip_address:port_number/
+ASM;
parallelASMReadThreads=6;readAheadBlocks=150000;
```

For more information on values that extra connection attributes support, see [Extra connection attributes when using Oracle as a source for AWS DMS \(p. 165\)](#).

In addition, the performance of a CDC task with an Oracle source that uses ASM depends on other settings that you choose. These settings include your AWS DMS extra connection attributes and the SQL settings to configure the Oracle source. For more information on extra connection attributes for an Oracle source using ASM, see [Extra connection attributes when using Oracle as a source for AWS DMS \(p. 165\)](#)

You also need to choose an appropriate CDC start point. Typically when you do this, you want to identify the point of transaction processing that captures the earliest open transaction to begin CDC from. Otherwise, the CDC task can miss earlier open transactions. For an Oracle source database, you can choose a CDC native start point based on the Oracle system change number (SCN) to identify this earliest open transaction. For more information, see [Performing replication starting from a CDC start point \(p. 482\)](#).

For more information on configuring CDC for a self-managed Oracle database as a source, see [Account privileges required when using Oracle LogMiner to access the redo logs \(p. 143\)](#), [Account privileges required when using AWS DMS Binary Reader to access the redo logs \(p. 144\)](#), and [Additional account privileges required when using Binary Reader with Oracle ASM \(p. 144\)](#).

For more information on configuring CDC for an AWS-managed Oracle database as a source, see [Configuring a CDC task to use Binary Reader with an RDS for Oracle source for AWS DMS \(p. 150\)](#) and [Using an Amazon RDS Oracle Standby \(read replica\) as a source with Binary Reader for CDC in AWS DMS \(p. 151\)](#).

## Workflows for configuring a self-managed or AWS-managed Oracle source database for AWS DMS

To configure a self-managed source database instance, use the following workflow steps , depending on how you perform CDC.

For this workflow step	If you perform CDC using LogMiner, do this	If you perform CDC using Binary Reader, do this
Grant Oracle account privileges.	See <a href="#">User account privileges required on a self-managed Oracle source for AWS DMS (p. 140)</a> .	See <a href="#">User account privileges required on a self-managed Oracle source for AWS DMS (p. 140)</a> .
Prepare the source database for replication using CDC.	See <a href="#">Preparing an Oracle self-managed source database for CDC using AWS DMS (p. 141)</a> .	See <a href="#">Preparing an Oracle self-managed source database for CDC using AWS DMS (p. 141)</a> .
Grant additional Oracle user privileges required for CDC.	See <a href="#">Account privileges required when using Oracle LogMiner to access the redo logs (p. 143)</a> .	See <a href="#">Account privileges required when using AWS DMS Binary Reader to access the redo logs (p. 144)</a> .
For an Oracle instance with ASM, grant additional user account	No additional action. AWS DMS supports Oracle ASM without additional account privileges.	See <a href="#">Additional account privileges required when using</a>

<b>For this workflow step</b>	<b>If you perform CDC using LogMiner, do this</b>	<b>If you perform CDC using Binary Reader, do this</b>
privileges required to access ASM for CDC.		Binary Reader with Oracle ASM (p. 144).
If you haven't already done so, configure the task to use LogMiner or Binary Reader for CDC.	See <a href="#">Using Oracle LogMiner or AWS DMS Binary Reader for CDC (p. 135)</a> .	See <a href="#">Using Oracle LogMiner or AWS DMS Binary Reader for CDC (p. 135)</a> .
Configure Oracle Standby as a source for CDC.	AWS DMS doesn't support Oracle Standby as a source.	See <a href="#">Using a self-managed Oracle Standby as a source with Binary Reader for CDC in AWS DMS (p. 145)</a> .

Use the following workflow steps to configure an AWS-managed Oracle source database instance.

<b>For this workflow step</b>	<b>If you perform CDC using LogMiner, do this</b>	<b>If you perform CDC using Binary Reader, do this</b>
Grant Oracle account privileges.	For more information, see <a href="#">User account privileges required on an AWS-managed Oracle source for AWS DMS (p. 147)</a> .	For more information, see <a href="#">User account privileges required on an AWS-managed Oracle source for AWS DMS (p. 147)</a> .
Prepare the source database for replication using CDC.	For more information, see <a href="#">Configuring an AWS-managed Oracle source for AWS DMS (p. 149)</a> .	For more information, see <a href="#">Configuring an AWS-managed Oracle source for AWS DMS (p. 149)</a> .
Grant additional Oracle user privileges required for CDC.	No additional account privileges are required.	For more information, see <a href="#">Configuring a CDC task to use Binary Reader with an RDS for Oracle source for AWS DMS (p. 150)</a> .
If you haven't already done so, configure the task to use LogMiner or Binary Reader for CDC.	For more information, see <a href="#">Using Oracle LogMiner or AWS DMS Binary Reader for CDC (p. 135)</a> .	For more information, see <a href="#">Using Oracle LogMiner or AWS DMS Binary Reader for CDC (p. 135)</a> .
Configure Oracle Standby as a source for CDC.	AWS DMS doesn't support Oracle Standby as a source.	For more information, see <a href="#">Using an Amazon RDS Oracle Standby (read replica) as a source with Binary Reader for CDC in AWS DMS (p. 151)</a> .

## Working with a self-managed Oracle database as a source for AWS DMS

A *self-managed database* is a database that you configure and control, either a local on-premises database instance or a database on Amazon EC2. Following, you can find out about the privileges and configurations you need when using a self-managed Oracle database with AWS DMS.

## User account privileges required on a self-managed Oracle source for AWS DMS

To use an Oracle database as a source in AWS DMS, grant the following privileges to the Oracle user specified in the Oracle endpoint connection settings.

### Note

When granting privileges, use the actual name of objects, not the synonym for each object. For example, use V\_\$OBJECT including the underscore, not V\$OBJECT without the underscore.

```
GRANT CREATE SESSION TO db_user;
GRANT SELECT ANY TRANSACTION TO db_user;
GRANT SELECT ON V_$ARCHIVED_LOG TO db_user;
GRANT SELECT ON V_$LOG TO db_user;
GRANT SELECT ON V_$LOGFILE TO db_user;
GRANT SELECT ON V_$LOGMNR_LOGS TO db_user;
GRANT SELECT ON V_$LOGMNR_CONTENTS TO db_user;
GRANT SELECT ON V_$DATABASE TO db_user;
GRANT SELECT ON V_$THREAD TO db_user;
GRANT SELECT ON V_$PARAMETER TO db_user;
GRANT SELECT ON V_$NLS_PARAMETERS TO db_user;
GRANT SELECT ON V_$TIMEZONE_NAMES TO db_user;
GRANT SELECT ON V_$TRANSACTION TO db_user;
GRANT SELECT ON V_$CONTAINERS TO db_user;
GRANT SELECT ON ALL_INDEXES TO db_user;
GRANT SELECT ON ALL_OBJECTS TO db_user;
GRANT SELECT ON ALL_TABLES TO db_user;
GRANT SELECT ON ALL_USERS TO db_user;
GRANT SELECT ON ALL_CATALOG TO db_user;
GRANT SELECT ON ALL_CONSTRAINTS TO db_user;
GRANT SELECT ON ALL_CONS_COLUMNS TO db_user;
GRANT SELECT ON ALL_TAB_COLS TO db_user;
GRANT SELECT ON ALL_IND_COLUMNS TO db_user;
GRANT SELECT ON ALL_ENCRYPTED_COLUMNS TO db_user;
GRANT SELECT ON ALL_LOG_GROUPS TO db_user;
GRANT SELECT ON ALL_TAB_PARTITIONS TO db_user;
GRANT SELECT ON SYS.DBA_REGISTRY TO db_user;
GRANT SELECT ON SYS.OBJ$ TO db_user;
GRANT SELECT ON DBA_TABLESPACES TO db_user;
GRANT SELECT ON DBA_OBJECTS TO db_user; -- Required if the Oracle version is earlier than 11.2.0.3.
GRANT SELECT ON SYS.ENC$ TO db_user; -- Required if transparent data encryption (TDE) is enabled. For more information on using Oracle TDE with AWS DMS, see Supported encryption methods for using Oracle as a source for AWS DMS.
GRANT SELECT ON GV_$TRANSACTION TO db_user; -- Required if the source database is Oracle RAC in AWS DMS versions 3.4.6 and higher.
GRANT SELECT ON V_$DATAGUARD_STATS TO db_user ; -- Required if the source database is Oracle Data Guard and Oracle Standby is used in the latest release of DMS version 3.4.6, version 3.4.7, and higher.
```

Grant the additional following privilege for each replicated table when you are using a specific table list.

```
GRANT SELECT on any-replicated-table to db_user;
```

Grant the additional following privilege to validate LOB columns with the validation feature.

```
GRANT EXECUTE ON SYS.DBMS_CRYPTO TO db_user;
```

Grant the additional following privilege if you use binary reader instead of LogMiner.

```
GRANT SELECT ON SYS.DBA_DIRECTORIES TO db_user;
```

Grant the additional following privilege to expose views.

```
GRANT SELECT on ALL_VIEWS to dms_user;
```

To expose views, you must also add the exposeViews=true extra connection attribute to your source endpoint.

### Prerequisites for handling open transactions for Oracle Standby

When using AWS DMS versions 3.4.6 and higher, perform the following steps to handle open transactions for Oracle Standby.

1. Create a database link named, AWSDMS\_DBLINK on the primary database. *DMS\_USER* will use the database link to connect to the primary database. Note that the database link is executed from the standby instance to query the open transactions running on the primary database. See the following example.

```
CREATE PUBLIC DATABASE LINK AWSDMS_DBLINK
  CONNECT TO DMS_USER IDENTIFIED BY DMS_USER_PASSWORD
  USING '(DESCRIPTION=
    (ADDRESS=(PROTOCOL=TCP)(HOST=PRIMARY_HOST_NAME_OR_IP)(PORT=PORT))
    (CONNECT_DATA=(SERVICE_NAME=SID)))
  )';
```

2. Verify the connection to the database link using *DMS\_USER* is established, as shown in the following example.

```
select 1 from dual@AWSDMS_DBLINK
```

## Preparing an Oracle self-managed source database for CDC using AWS DMS

Prepare your self-managed Oracle database as a source to run a CDC task by doing the following:

- [Verifying that AWS DMS supports the source database version \(p. 141\)](#).
- [Making sure that ARCHIVELOG mode is on \(p. 142\)](#).
- [Setting up supplemental logging \(p. 142\)](#).

### Verifying that AWS DMS supports the source database version

Run a query like the following to verify that the current version of the Oracle source database is supported by AWS DMS.

```
SELECT name, value, description FROM v$parameter WHERE name = 'compatible';
```

Here, name, value, and description are columns somewhere in the database that are being queried based on the value of name. If this query runs without error, AWS DMS supports the current version of the database and you can continue with the migration. If the query raises an error, AWS DMS doesn't support the current version of the database. To proceed with migration, first convert the Oracle database to a version supported by AWS DMS.

## Making sure that ARCHIVELOG mode is on

You can run Oracle in two different modes: the ARCHIVELOG mode and the NOARCHIVELOG mode. To run a CDC task, run the database in ARCHIVELOG mode. To know if the database is in ARCHIVELOG mode, execute the following query.

```
SQL> SELECT log_mode FROM v$database;
```

If NOARCHIVELOG mode is returned, set the database to ARCHIVELOG per Oracle instructions.

## Setting up supplemental logging

To capture ongoing changes, AWS DMS requires that you enable minimal supplemental logging on your Oracle source database. In addition, you need to enable supplemental logging on each replicated table in the database.

By default, AWS DMS adds PRIMARY KEY supplemental logging on all replicated tables. To allow AWS DMS to add PRIMARY KEY supplemental logging, grant the following privilege for each replicated table.

```
ALTER on any-replicated-table;
```

You can disable the default PRIMARY KEY supplemental logging added by AWS DMS using the extra connection attribute `addSupplementalLogging`. For more information, see [Extra connection attributes when using Oracle as a source for AWS DMS \(p. 165\)](#).

Make sure to turn on supplemental logging if your replication task updates a table using a WHERE clause that doesn't reference a primary key column.

## To manually set up supplemental logging

1. Run the following query to verify if supplemental logging is already enabled for the database.

```
SELECT supplemental_log_data_min FROM v$database;
```

If the result returned is YES or IMPLICIT, supplemental logging is enabled for the database.

If not, enable supplemental logging for the database by running the following command.

```
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
```

2. Make sure that the required supplemental logging is added for each replicated table.

Consider the following:

- If ALL COLUMNS supplemental logging is added to the table, you don't need to add more logging.
- If a primary key exists, add supplemental logging for the primary key. You can do this either by using the format to add supplemental logging on the primary key itself, or by adding supplemental logging on the primary key columns on the database.

```
ALTER TABLE Tablename ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;  
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA (PRIMARY KEY) COLUMNS;
```

- If no primary key exists and the table has a single unique index, add all of the unique index's columns to the supplemental log.

```
ALTER TABLE TableName ADD SUPPLEMENTAL LOG GROUP LogGroupName  
(UniqueIndexColumn1, UniqueIndexColumn2 ...) ALWAYS;
```

Using SUPPLEMENTAL LOG DATA (UNIQUE INDEX) COLUMNS doesn't add the unique index columns to the log.

- If no primary key exists and the table has multiple unique indexes, AWS DMS selects the first unique index in an alphabetically ordered ascending list. You need to add supplemental logging on the selected index's columns as in the previous item.

Using SUPPLEMENTAL LOG DATA (UNIQUE INDEX) COLUMNS doesn't add the unique index columns to the log.

- If no primary key exists and there is no unique index, add supplemental logging on all columns.

```
ALTER TABLE TableName ADD SUPPLEMENTAL LOG DATA (ALL) COLUMNS;
```

In some cases, the target table primary key or unique index is different than the source table primary key or unique index. In such cases, add supplemental logging manually on the source table columns that make up the target table primary key or unique index.

Also, if you change the target table primary key, add supplemental logging on the target unique index's columns instead of the columns of the source primary key or unique index.

If a filter or transformation is defined for a table, you might need to enable additional logging.

Consider the following:

- If ALL COLUMNS supplemental logging is added to the table, you don't need to add more logging.
- If the table has a unique index or a primary key, add supplemental logging on each column that is involved in a filter or transformation. However, do so only if those columns are different from the primary key or unique index columns.
- If a transformation includes only one column, don't add this column to a supplemental logging group. For example, for a transformation A+B, add supplemental logging on both columns A and B. However, for a transformation substring(A,10) don't add supplemental logging on column A.
- To set up supplemental logging on primary key or unique index columns and other columns that are filtered or transformed, you can set up USER\_LOG\_GROUP supplemental logging. Add this logging on both the primary key or unique index columns and any other specific columns that are filtered or transformed.

For example, to replicate a table named TEST.LOGGING with primary key ID and a filter by the column NAME, you can run a command similar to the following to create the log group supplemental logging.

```
ALTER TABLE TEST.LOGGING ADD SUPPLEMENTAL LOG GROUP TEST_LOG_GROUP (ID, NAME) ALWAYS;
```

## Account privileges required when using Oracle LogMiner to access the redo logs

To access the redo logs using the Oracle LogMiner, grant the following privileges to the Oracle user specified in the Oracle endpoint connection settings.

```
GRANT EXECUTE on DBMS_LOGMNR to db_user;
GRANT SELECT on V_$LOGMNR_LOGS to db_user;
GRANT SELECT on V_$LOGMNR_CONTENTS to db_user;
GRANT LOGMINING to db_user; -- Required only if the Oracle version is 12c or later.
```

## Account privileges required when using AWS DMS Binary Reader to access the redo logs

To access the redo logs using the AWS DMS Binary Reader, grant the following privileges to the Oracle user specified in the Oracle endpoint connection settings.

```
GRANT SELECT on v_$transportable_platform to db_user; -- Grant this privilege if the redo logs are stored in Oracle Automatic Storage Management (ASM) and AWS DMS accesses them from ASM.  
GRANT CREATE ANY DIRECTORY to db_user; -- Grant this privilege to allow AWS DMS to use Oracle BFILE read file access in certain cases. This access is required when the replication instance doesn't have file-level access to the redo logs and the redo logs are on non-ASM storage.  
GRANT EXECUTE on DBMS_FILE_TRANSFER to db_user; -- Grant this privilege to copy the redo log files to a temporary folder using the CopyToTempFolder method.  
GRANT EXECUTE on DBMS_FILE_GROUP to db_user;
```

Binary Reader works with Oracle file features that include Oracle directories. Each Oracle directory object includes the name of the folder containing the redo log files to process. These Oracle directories aren't represented at the file system level. Instead, they are logical directories that are created at the Oracle database level. You can view them in the Oracle ALL\_DIRECTORIES view.

If you want AWS DMS to create these Oracle directories, grant the CREATE ANY DIRECTORY privilege specified preceding. AWS DMS creates the directory names with the DMS\_ prefix. If you don't grant the CREATE ANY DIRECTORY privilege, create the corresponding directories manually. In some cases when you create the Oracle directories manually, the Oracle user specified in the Oracle source endpoint isn't the user that created these directories. In these cases, also grant the READ on DIRECTORY privilege.

If the Oracle source endpoint is in Active Dataguard Standby (ADG), see the [How to use Binary Reader with ADG](#) post on the AWS Database Blog.

### Note

AWS DMS CDC doesn't support Active Dataguard Standby that is not configured to use automatic redo transport service.

In some cases, you might use Oracle Managed Files (OMF) for storing the logs. Or your source endpoint is in ADG and thus you can't grant the CREATE ANY DIRECTORY privilege. In these cases, manually create the directories with all the possible log locations before starting the AWS DMS replication task. If AWS DMS doesn't find a precreated directory that it expects, the task stops. Also, AWS DMS doesn't delete the entries it has created in the ALL\_DIRECTORIES view, so manually delete them.

## Additional account privileges required when using Binary Reader with Oracle ASM

To access the redo logs in Automatic Storage Management (ASM) using Binary Reader, grant the following privileges to the Oracle user specified in the Oracle endpoint connection settings.

```
SELECT ON v_$transportable_platform  
SYSASM -- To access the ASM account with Oracle 11g Release 2 (version 11.2.0.2) and later, grant the Oracle endpoint user the SYSASM privilege. For older supported Oracle versions, it's typically sufficient to grant the Oracle endpoint user the SYSDBA privilege.
```

You can validate ASM account access by opening a command prompt and invoking one of the following statements, depending on your Oracle version as specified preceding.

If you need the SYSDBA privilege, use the following.

```
sqlplus asmuser/asmpassword@+asmserver as sysdba
```

If you need the SYSASM privilege, use the following.

```
sqlplus asmuser/asmpassword@asmserver as sysasm
```

## Using a self-managed Oracle Standby as a source with Binary Reader for CDC in AWS DMS

To configure an Oracle Standby instance as a source when using Binary Reader for CDC, start with the following prerequisites:

- AWS DMS currently supports only Oracle Active Data Guard Standby.
- Make sure that the Oracle Data Guard configuration uses:
  - Redo transport services for automated transfers of redo data.
  - Apply services to automatically apply redo to the standby database.

To confirm those requirements are met, execute the following query.

```
SQL> select open_mode, database_role from v$database;
```

From the output of that query, confirm that the standby database is opened in READ ONLY mode and redo is being applied automatically. For example:

OPEN_MODE	DATABASE_ROLE
READ ONLY WITH APPLY	PHYSICAL STANDBY

### To configure an Oracle Standby instance as a source when using Binary Reader for CDC

1. Grant additional privileges required to access standby log files.

```
GRANT SELECT ON v_$standby_log TO db_user;
```

2. Create a source endpoint for the Oracle Standby by using the AWS Management Console or AWS CLI. When creating the endpoint, specify the following extra connection attributes.

```
useLogminerReader=N;useBfile=Y
```

#### Note

In AWS DMS, you can use extra connection attributes to specify if you want to migrate from the archive logs instead of the redo logs. For more information, see [Extra connection attributes when using Oracle as a source for AWS DMS \(p. 165\)](#).

3. Configure archived log destination.

DMS binary reader for Oracle source without ASM uses Oracle Directories to access archived redo logs. If your database is configured to use Fast Recovery Area (FRA) as an archive log destination, the location of archive redo files isn't constant. Each day that archived redo logs are generated results in a new directory being created in the FRA, using the directory name format YYYY\_MM\_DD. For example:

```
DB_RECOVERY_FILE_DEST/SID/archivelog/YYYY_MM_DD
```

When DMS needs access to archived redo files in the newly created FRA directory and the primary read-write database is being used as a source, DMS creates a new or replaces an existing Oracle directory, as follows.

```
CREATE OR REPLACE DIRECTORY dmsrep_taskid AS 'DB_RECOVERY_FILE_DEST/SID/archivelog/  
YYYY_MM_DD';
```

When the standby database is being used as a source, DMS is unable to create or replace the Oracle directory because the database is in read-only mode. But, you can choose to perform one of these additional steps:

- a. Modify `log_archive_dest_1` to use an actual path instead of FRA in such a configuration that Oracle won't create daily subdirectories:

```
ALTER SYSTEM SET log_archive_dest_1='LOCATION=full directory path'
```

Then, create an Oracle directory object to be used by DMS:

```
CREATE OR REPLACE DIRECTORY dms_archived_logs AS 'full directory path';
```

- b. Create an additional archive log destination and an Oracle directory object pointing to that destination. For example:

```
ALTER SYSTEM SET log_archive_dest_3='LOCATION=full directory path';  
CREATE DIRECTORY dms_archived_log AS 'full directory path';
```

Then add an extra connection attribute to the task source endpoint:

```
archivedLogDestId=3
```

- c. Manually pre-create Oracle directory objects to be used by DMS.

```
CREATE DIRECTORY dms_archived_log_20210301 AS 'DB_RECOVERY_FILE_DEST/SID/  
archivelog/2021_03_01';  
CREATE DIRECTORY dms_archived_log_20210302 AS 'DB_RECOVERY_FILE_DEST>/SID>/  
archivelog/2021_03_02';  
...
```

- d. Create an Oracle scheduler job that runs daily and creates the required directory.

## Using a user-managed database on Oracle Cloud Infrastructure (OCI) as a source for CDC in AWS DMS

A user-managed database is a database that you configure and control, such as an Oracle database created on a virtual machine (VM), bare metal, or Exadata server. Or, databases that you configure and control that run on dedicated infrastructure, like Oracle Cloud Infrastructure (OCI). The following information describes the privileges and configurations you need when using an Oracle user-managed database on OCI as a source for change data capture (CDC) in AWS DMS.

### To configure an OCI hosted user-managed Oracle database as a source for change data capture

1. Grant required user account privileges for a user-managed Oracle source database on OCI. For more information, see [Account privileges for a self-managed Oracle source endpoint \(p. 140\)](#).

2. Grant account privileges required when using Binary Reader to access the redo logs. For more information, see [Account privileges required when using Binary Reader \(p. 144\)](#).
3. Add account privileges that are required when using Binary Reader with Oracle Automatic Storage Management (ASM). For more information, see [Additional account privileges required when using Binary Reader with Oracle ASM \(p. 144\)](#).
4. Set-up supplemental logging. For more information, see [Setting up supplemental logging \(p. 142\)](#).
5. Set-up TDE encryption. For more information, see [Encryption methods when using an Oracle database as a source endpoint \(p. 159\)](#).

The following limitations apply when replicating data from an Oracle source database on Oracle Cloud Infrastructure (OCI).

### Limitations

- DMS doesn't support using Oracle LogMiner to access the redo logs.
- DMS doesn't support Autonomous DB.

## Working with an AWS-managed Oracle database as a source for AWS DMS

An AWS-managed database is a database that is on an Amazon service such as Amazon RDS, Amazon Aurora, or Amazon S3. Following, you can find the privileges and configurations that you need to set up when using an AWS-managed Oracle database with AWS DMS.

### User account privileges required on an AWS-managed Oracle source for AWS DMS

Grant the following privileges to the Oracle user account specified in the Oracle source endpoint definition.

#### Important

For all parameter values such as `db_user` and `any-replicated-table`, Oracle assumes the value is all uppercase unless you specify the value with a case-sensitive identifier. For example, suppose that you create a `db_user` value without using quotation marks, as in `CREATE USER myuser` or `CREATE USER MYUSER`. In this case, Oracle identifies and stores the value as all uppercase (MYUSER). If you use quotation marks, as in `CREATE USER "MyUser"` or `CREATE USER 'MyUser'`, Oracle identifies and stores the case-sensitive value that you specify (`MyUser`).

```
GRANT CREATE SESSION to db_user;
GRANT SELECT ANY TRANSACTION to db_user;
GRANT SELECT on DBA_TABLESPACES to db_user;
GRANT SELECT ON any-replicated-table to db_user;
GRANT EXECUTE on rdsadmin.rdsadmin_util to db_user;
-- For Oracle 12c or later:
GRANT LOGMINING to db_user; - Required only if the Oracle version is 12c or later.
```

In addition, grant SELECT and EXECUTE permissions on SYS objects using the Amazon RDS procedure `rdsadmin.rdsadmin_util.grant_sys_object` as shown. For more information, see [Granting SELECT or EXECUTE privileges to SYS objects](#).

```
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_VIEWS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_TAB_PARTITIONS', 'db_user', 'SELECT');
```

```

exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_INDEXES', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_OBJECTS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_TABLES', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_USERS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_CATALOG', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_CONSTRAINTS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_CONS_COLUMNS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_TAB_COLS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_IND_COLUMNS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_LOG_GROUPS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$ARCHIVED_LOG', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$LOG', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$LOGFILE', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$DATABASE', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$THREAD', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$PARAMETER', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$NLS_PARAMETERS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$TIMEZONE_NAMES', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$TRANSACTION', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$CONTAINERS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_REGISTRY', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('OBJ$', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('ALL_ENCRYPTED_COLUMNS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$LOGMNR_LOGS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$LOGMNR_CONTENTS', 'db_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_LOGMNR', 'db_user', 'EXECUTE');

-- (as of Oracle versions 12.1 and later)
exec rdsadmin.rdsadmin_util.grant_sys_object('REGISTRY$SQLPATCH', 'db_user', 'SELECT');

-- (for Amazon RDS Active Dataguard Standby (ADG))
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$STANDBY_LOG', 'db_user', 'SELECT');

-- (for transparent data encryption (TDE))

exec rdsadmin.rdsadmin_util.grant_sys_object('ENC$', 'db_user', 'SELECT');

-- (for validation with LOB columns)
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMS_CRYPTO', 'db_user', 'EXECUTE');

-- (for binary reader)
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_DIRECTORIES', 'db_user', 'SELECT');

-- Required when the source database is Oracle Data guard, and Oracle Standby is used in
-- the latest release of DMS version 3.4.6, version 3.4.7, and higher.

exec rdsadmin.rdsadmin_util.grant_sys_object('V_$DATAGUARD_STATS', 'db_user', 'SELECT');

```

For more information on using Amazon RDS Active Dataguard Standby (ADG) with AWS DMS see [Using an Amazon RDS Oracle Standby \(read replica\) as a source with Binary Reader for CDC in AWS DMS \(p. 151\)](#).

For more information on using Oracle TDE with AWS DMS, see [Supported encryption methods for using Oracle as a source for AWS DMS \(p. 159\)](#).

#### [Prerequisites for handling open transactions for Oracle Standby](#)

When using AWS DMS versions 3.4.6 and higher, perform the following steps to handle open transactions for Oracle Standby.

1. Create a database link named, AWSDMS\_DBLINK on the primary database. **DMS\_USER** will use the database link to connect to the primary database. Note that the database link is executed from the

standby instance to query the open transactions running on the primary database. See the following example.

```
CREATE PUBLIC DATABASE LINK AWS_DMS_DBLINK
CONNECT TO DMS_USER IDENTIFIED BY DMS_USER_PASSWORD
USING '(DESCRIPTION=
    (ADDRESS=(PROTOCOL=TCP)(HOST=PRIMARY_HOST_NAME_OR_IP)(PORT=PORT))
    (CONNECT_DATA=(SERVICE_NAME=SID)))
)';
```

2. Verify the connection to the database link using *DMS\_USER* is established, as shown in the following example.

```
select 1 from dual@AWS_DMS_DBLINK
```

## Configuring an AWS-managed Oracle source for AWS DMS

Before using an AWS-managed Oracle database as a source for AWS DMS, perform the following tasks for the Oracle database:

- Enable automatic backups. For more information about enabling automatic backups, see [Enabling automated backups](#) in the *Amazon RDS User Guide*.
- Set up supplemental logging.
- Set up archiving. Archiving the redo logs for your Amazon RDS for Oracle DB instance allows AWS DMS to retrieve the log information using Oracle LogMiner or Binary Reader.

### To set up archiving

1. Run the `rdsadmin.rdsadmin_util.set_configuration` command to set up archiving.

For example, to retain the archived redo logs for 24 hours, run the following command.

```
exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours',24);
commit;
```

#### Note

The commit is required for a change to take effect.

2. Make sure that your storage has enough space for the archived redo logs during the specified retention period. For example, if your retention period is 24 hours, calculate the total size of your accumulated archived redo logs over a typical hour of transaction processing and multiply that total by 24. Compare this calculated 24-hour total with your available storage space and decide if you have enough storage space to handle a full 24 hours transaction processing.

### To set up supplemental logging

1. Run the following command to enable supplemental logging at the database level.

```
exec rdsadmin.rdsadmin_util.alter_supplemental_logging('ADD');
```

2. Run the following command to enable primary key supplemental logging.

```
exec rdsadmin.rdsadmin_util.alter_supplemental_logging('ADD','PRIMARY KEY');
```

3. (Optional) Enable key-level supplemental logging at the table level.

Your source database incurs a small bit of overhead when key-level supplemental logging is enabled. Therefore, if you are migrating only a subset of your tables, you might want to enable key-level supplemental logging at the table level. To enable key-level supplemental logging at the table level, run the following command.

```
alter table table_name add supplemental log data (PRIMARY KEY) columns;
```

## Configuring a CDC task to use Binary Reader with an RDS for Oracle source for AWS DMS

You can configure AWS DMS to access the source Amazon RDS for Oracle instance redo logs using Binary Reader for CDC.

### Note

To use Oracle LogMiner, the minimum required user account privileges are sufficient. For more information, see [User account privileges required on an AWS-managed Oracle source for AWS DMS \(p. 147\)](#).

To use AWS DMS Binary Reader, specify additional settings and extra connection attributes for the Oracle source endpoint, depending on your AWS DMS version.

Binary Reader support is available in the following versions of Amazon RDS for Oracle:

- Oracle 11.2 – Versions 11.2.0.4V11 and later.
- Oracle 12.1 – Versions 12.1.0.2.V7 and later.
- Oracle 12.2 – All versions.
- Oracle 18.0 – All versions.
- Oracle 19.0 – All versions.

### To configure CDC using Binary Reader

1. Log in to your Amazon RDS for Oracle source database as the master user and run the following stored procedures to create the server-level directories.

```
exec rdsadmin.rdsadmin_master_util.create_archivelog_dir;
exec rdsadmin.rdsadmin_master_util.create_onlinelog_dir;
```

2. Grant the following privileges to the Oracle user account that is used to access the Oracle source endpoint.

```
GRANT READ ON DIRECTORY ONLINELOG_DIR TO db_user;
GRANT READ ON DIRECTORY ARCHIVELOG_DIR TO db_user;
```

3. Set the following extra connection attributes on the Amazon RDS Oracle source endpoint:

- For RDS Oracle versions 11.2 and 12.1, set the following.

```
useLogminerReader=N;useBfile=Y;accessAlternateDirectly=false;useAlternateFolderForOnline=true;
oraclePathPrefix=/rdsdbdata/db/{$DATABASE_NAME}_A/;usePathPrefix=/rdsdbdata/
log/;replacePathPrefix=true;
```

- For RDS Oracle versions 12.2, 18.0, and 19.0, set the following.

```
useLogminerReader=N;useBfile=Y
```

**Note**

Make sure there's no white space following the semicolon separator (;) for multiple attribute settings, for example oneSetting;thenAnother.

For more information configuring a CDC task, see [Configuration for CDC on an Oracle source database \(p. 137\)](#).

## Using an Amazon RDS Oracle Standby (read replica) as a source with Binary Reader for CDC in AWS DMS

Verify the following prerequisites for using Amazon RDS for Oracle Standby as a source when using Binary Reader for CDC in AWS DMS:

- Use the Oracle master user to set up Binary Reader.
- Make sure that AWS DMS currently supports using only Oracle Active Data Guard Standby.

After you do so, use the following procedure to use RDS for Oracle Standby as a source when using Binary Reader for CDC.

### To configure an RDS for Oracle Standby as a source when using Binary Reader for CDC

1. Sign in to RDS for Oracle primary replica as the master user.
2. Run the following stored procedures as documented in the Amazon RDS User Guide to create the server level directories.

```
exec rdsadmin.rdsadmin_master_util.create_archivelog_dir;
exec rdsadmin.rdsadmin_master_util.create_onlinelog_dir;
```

3. Identify the directories created in step 2.

```
SELECT directory_name, directory_path FROM all_directories
WHERE directory_name LIKE ( 'ARCHIVELOG_DIR_%' )
      OR directory_name LIKE ( 'ONLINELOG_DIR_%' )
```

For example, the preceding code displays a list of directories like the following.

DIRECTORY_NAME	DIRECTORY_PATH
ARCHIVELOG_DIR_A	/rdsdbdata/db/ORCL_A/arch
ARCHIVELOG_DIR_B	/rdsdbdata/db/ORCL_B/arch
ONLINELOG_DIR_A	/rdsdbdata/db/ORCL_A/onlinelog
ONLINELOG_DIR_B	/rdsdbdata/db/ORCL_B/onlinelog

4. Grant the Read privilege on the preceding directories to the Oracle user account that is used to access the Oracle Standby.

```
GRANT READ ON DIRECTORY ARCHIVELOG_DIR_A TO db_user;
```

```
GRANT READ ON DIRECTORY ARCHIVELOG_DIR_B TO db_user;
GRANT READ ON DIRECTORY ONLINELOG_DIR_A TO db_user;
GRANT READ ON DIRECTORY ONLINELOG_DIR_B TO db_user;
```

5. Perform an archive log switch on the primary instance. Doing this makes sure that the changes to ALL\_DIRECTORIES are also ported to the Oracle Standby.
6. Run an ALL\_DIRECTORIES query on the Oracle Standby to confirm that the changes were applied.
7. Create a source endpoint for the Oracle Standby by using the AWS DMS Management Console or AWS Command Line Interface (AWS CLI). While creating the endpoint, specify the following extra connection attributes.

```
useLogminerReader=N;useBfile=Y;archivedLogDestId=1;additionalArchivedLogDestId=2
```

8. After creating the endpoint, use **Test endpoint connection** on the **Create endpoint** page of the console or the AWS CLI `test-connection` command to verify that connectivity is established.

## Limitations on using Oracle as a source for AWS DMS

The following limitations apply when using an Oracle database as a source for AWS DMS:

- AWS DMS doesn't support Oracle Extended data types at this time.
- AWS DMS doesn't support long object names (over 30 bytes).
- AWS DMS doesn't support function-based indexes.
- If you manage supplemental logging and carry out transformations on any of the columns, make sure that supplemental logging is activated for all fields and columns. For more information on setting up supplemental logging, see the following topics:
  - For a self-managed Oracle source database, see [Setting up supplemental logging \(p. 142\)](#).
  - For an AWS-managed Oracle source database, see [Configuring an AWS-managed Oracle source for AWS DMS \(p. 149\)](#).
- AWS DMS doesn't support the multi-tenant container root database (CDB\$ROOT). It does support a PDB using the Binary Reader.
- AWS DMS doesn't support deferred constraints.
- Secure LOBs are supported using Full LOB mode only by performing LOB lookup.
- AWS DMS supports the `rename table table-name to new-table-name` syntax for all supported Oracle versions 11 and later. This syntax isn't supported for any Oracle version 10 source databases.
- AWS DMS doesn't replicate results of the DDL statement `ALTER TABLE ADD column data_type DEFAULT default_value`. Instead of replicating `default_value` to the target, it sets the new column to NULL.
- When using AWS DMS version 3.4.7 or later, to replicate changes that result from partition or subpartition operations, do the following before starting a DMS task.
  - Manually create the partitioned table structure (DDL);
  - Make sure the DDL is the same on both Oracle source and Oracle target;
  - Set the extra connection attribute `enableHomogenousPartitionOps=true`.

For more information about `enableHomogenousPartitionOps`, see [Extra connection attributes when using Oracle as a source for AWS DMS \(p. 165\)](#). Also, note that on FULL+CDC tasks, DMS doesn't replicate data changes captured as part of the cached changes. In that use case, recreate the table structure on the Oracle target and reload the tables in question.

Prior to AWS DMS version 3.4.7:

DMS doesn't replicate data changes that result from partition or subpartition operations (ADD, DROP, EXCHANGE, and TRUNCATE). Such updates might cause the following errors during replication:

- For ADD operations, updates and deletes on the added data might raise a "0 rows affected" warning.
- For DROP and TRUNCATE operations, new inserts might raise "duplicates" errors.
- EXCHANGE operations might raise both a "0 rows affected" warning and "duplicates" errors.

To replicate changes that result from partition or subpartition operations, reload the tables in question. After adding a new empty partition, operations on the newly added partition are replicated to the target as normal.

- AWS DMS versions prior to 3.4 don't support data changes on the target that result from running the CREATE TABLE AS statement on the source. However, the new table is created on the target.
- AWS DMS doesn't capture changes made by the Oracle DBMS\_REDEFINITION package, for example the table metadata and the OBJECT\_ID field.
- AWS DMS maps empty BLOB and CLOB columns to NULL on the target.
- When capturing changes with Oracle 11 LogMiner, an update on a CLOB column with a string length greater than 1982 is lost, and the target is not updated.
- During change data capture (CDC), AWS DMS doesn't support batch updates to numeric columns defined as a primary key.
- AWS DMS doesn't support certain UPDATE commands. The following example is an unsupported UPDATE command.

```
UPDATE TEST_TABLE SET KEY=KEY+1;
```

Here, TEST\_TABLE is the table name and KEY is a numeric column defined as a primary key.

- AWS DMS doesn't support full LOB mode for loading LONG and LONG RAW columns. Instead, you can use limited LOB mode for migrating these datatypes to an Oracle target. In limited LOB mode, AWS DMS truncates any data to 64 KB that you set to LONG or LONG RAW columns longer than 64 KB.
- AWS DMS doesn't support full LOB mode for loading XMLTYPE columns. Instead, you can use limited LOB mode for migrating XMLTYPE columns to an Oracle target. In limited LOB mode, DMS truncates any data larger than the user defined 'Maximum LOB size' variable. The maximum recommended value for 'Maximum LOB size' is 100MB.
- AWS DMS doesn't replicate tables whose names contain apostrophes.
- AWS DMS supports CDC from materialized views. But DMS doesn't support CDC from any other views.
- AWS DMS doesn't support CDC for index-organized tables with an overflow segment.
- When you use Oracle LogMiner to access the redo logs, AWS DMS has the following limitations:
  - For Oracle 12 only, AWS DMS doesn't replicate any changes to LOB columns.
  - For all Oracle versions, AWS DMS doesn't replicate the result of UPDATE operations on XMLTYPE and LOB columns.
  - AWS DMS doesn't support XA transactions in replication while using Oracle LogMiner.
  - Oracle LogMiner doesn't support connections to a pluggable database (PDB). To connect to a PDB, access the redo logs using Binary Reader.
  - SHRINK SPACE operations aren't supported.
- When you use Binary Reader, AWS DMS has these limitations:
  - It doesn't support table clusters.
  - It supports only table-level SHRINK SPACE operations. This level includes the full table, partitions, and sub-partitions.
  - It doesn't support changes to index-organized tables with key compression.
  - It doesn't support implementing online redo logs on raw devices.
  - Binary Reader supports TDE only for self-managed Oracle databases since RDS for Oracle doesn't support wallet password retrieval for TDE encryption keys.
  - AWS DMS doesn't support connections to an Amazon RDS Oracle source using an Oracle Automatic Storage Management (ASM) proxy.

- AWS DMS doesn't support virtual columns.
- AWS DMS doesn't support the ROWID data type or materialized views based on a ROWID column.

AWS DMS has partial support for Oracle Materialized Views. For full-loads, DMS can do a full-load copy of an Oracle Materialized View. DMS copies the Materialized View as a base table to the target system and ignores any ROWID columns in the Materialized View. For ongoing replication (CDC), DMS tries to replicate changes to the Materialized View data but the results might not be ideal. Specifically, if the Materialized View is completely refreshed, DMS replicates individual deletes for all the rows, followed by individual inserts for all the rows. That is a very resource intensive exercise and might perform poorly for materialized views with large numbers of rows. For ongoing replication where the materialized views do a fast refresh, DMS tries to process and replicate the fast refresh data changes. In either case, DMS skips any ROWID columns in the materialized view.

- AWS DMS doesn't load or capture global temporary tables.
- For S3 targets using replication, enable supplemental logging on every column so source row updates can capture every column value. An example follows: `alter table yourtablename add supplemental log data (all) columns;`.
- An update for a row with a composite unique key that contains null can't be replicated at the target.
- AWS DMS doesn't support use of multiple Oracle TDE encryption keys on the same source endpoint. Each endpoint can have only one attribute for TDE encryption Key Name "securityDbEncryptionName", and one TDE password for this key.
- When replicating from Amazon RDS for Oracle, TDE is supported only with encrypted tablespace and using Oracle LogMiner.
- AWS DMS does not support multiple table rename operations in quick succession.
- When using Oracle 19.0 as source, AWS DMS doesn't support the following features:
  - Data-guard DML redirect
  - Partitioned hybrid tables
  - Schema-only Oracle accounts
- AWS DMS doesn't support migration of tables or views of type BIN\$ or DR\$.
- Beginning with Oracle 18.x, AWS DMS doesn't support change data capture (CDC) from Oracle Express Edition (Oracle Database XE).
- When migrating data from a CHAR column, DMS truncates any trailing spaces.
- AWS DMS doesn't support a virtual private database (VPD).
- AWS DMS doesn't support replication from application containers.
- AWS DMS doesn't support performing Oracle Flashback Database and restore points, as these operations affect the consistency of Oracle Redo Log files.

## SSL support for an Oracle endpoint

AWS DMS Oracle endpoints support SSL V3 for the none and verify-ca SSL modes. To use SSL with an Oracle endpoint, upload the Oracle wallet for the endpoint instead of .pem certificate files.

### Topics

- [Using an existing certificate for Oracle SSL \(p. 154\)](#)
- [Using a self-signed certificate for Oracle SSL \(p. 155\)](#)

## Using an existing certificate for Oracle SSL

To use an existing Oracle client installation to create the Oracle wallet file from the CA certificate file, do the following steps.

## To use an existing oracle client installation for Oracle SSL with AWS DMS

1. Set the ORACLE\_HOME system variable to the location of your dbhome\_1 directory by running the following command.

```
prompt>export ORACLE_HOME=/home/user/app/user/product/12.1.0/dbhome_1
```

2. Append \$ORACLE\_HOME/lib to the LD\_LIBRARY\_PATH system variable.

```
prompt>export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
```

3. Create a directory for the Oracle wallet at \$ORACLE\_HOME/ssl\_wallet.

```
prompt>mkdir $ORACLE_HOME/ssl_wallet
```

4. Put the CA certificate .pem file in the ssl\_wallet directory. If you use Amazon RDS, you can download the rds-ca-2015-root.pem root CA certificate file hosted by Amazon RDS. For more information about downloading this file, see [Using SSL/TLS to encrypt a connection to a DB instance](#) in the *Amazon RDS User Guide*.
5. Run the following commands to create the Oracle wallet.

```
prompt>orapki wallet create -wallet $ORACLE_HOME/ssl_wallet -auto_login_only
prompt>orapki wallet add -wallet $ORACLE_HOME/ssl_wallet -trusted_cert -cert
$ORACLE_HOME/ssl_wallet/ca-cert.pem -auto_login_only
```

When you have completed the steps previous, you can import the wallet file with the ImportCertificate API call by specifying the certificate-wallet parameter. You can then use the imported wallet certificate when you select verify-ca as the SSL mode when creating or modifying your Oracle endpoint.

**Note**

Oracle wallets are binary files. AWS DMS accepts these files as-is.

## Using a self-signed certificate for Oracle SSL

To use a self-signed certificate for Oracle SSL, do the steps following, assuming an Oracle wallet password of oracle123.

### To use a self-signed certificate for Oracle SSL with AWS DMS

1. Create a directory you will use to work with the self-signed certificate.

```
mkdir -p /u01/app/oracle/self_signed_cert
```

2. Change into the directory you created in the previous step.

```
cd /u01/app/oracle/self_signed_cert
```

3. Create a root key.

```
openssl genrsa -out self-rootCA.key 2048
```

4. Self-sign a root certificate using the root key you created in the previous step.

```
openssl req -x509 -new -nodes -key self-rootCA.key  
-sha256 -days 3650 -out self-rootCA.pem
```

Use input parameters like the following.

- Country Name (2 letter code) [XX], for example: AU
- State or Province Name (full name) [], for example: NSW
- Locality Name (e.g., city) [Default City], for example: Sydney
- Organization Name (e.g., company) [Default Company Ltd], for example: AmazonWebService
- Organizational Unit Name (e.g., section) [], for example: DBeng
- Common Name (e.g., your name or your server's hostname) [], for example: aws
- Email Address [], for example: abcd.efgh@amazonwebservices.com

5. Create an Oracle wallet directory for the Oracle database.

```
mkdir -p /u01/app/oracle/wallet
```

6. Create a new Oracle wallet.

```
orapki wallet create -wallet "/u01/app/oracle/wallet" -pwd oracle123 -auto_login_local
```

7. Add the root certificate to the Oracle wallet.

```
orapki wallet add -wallet "/u01/app/oracle/wallet" -pwd oracle123 -trusted_cert  
-cert /u01/app/oracle/self_signed_cert/self-rootCA.pem
```

8. List the contents of the Oracle wallet. The list should include the root certificate.

```
orapki wallet display -wallet /u01/app/oracle/wallet -pwd oracle123
```

For example, this might display similar to the following.

```
Requested Certificates:  
User Certificates:  
Trusted Certificates:  
Subject: CN=aws,OU=DBeng,O= AmazonWebService,L=Sydney,ST=NSW,C=AU
```

9. Generate the Certificate Signing Request (CSR) using the ORAPKI utility.

```
orapki wallet add -wallet "/u01/app/oracle/wallet" -pwd oracle123  
-dn "CN=aws" -keysize 2048 -sign_alg sha256
```

10. Run the following command.

```
openssl pkcs12 -in /u01/app/oracle/wallet/ewallet.p12 -nodes -out /u01/app/oracle/  
wallet/nonoracle_wallet.pem
```

This has output like the following.

```
Enter Import Password:  
MAC verified OK  
Warning unsupported bag type: secretBag
```

11. Put 'dms' as the common name.

```
openssl req -new -key /u01/app/oracle/wallet/nonoracle_wallet.pem -out certdms.csr
```

Use input parameters like the following.

- Country Name (2 letter code) [XX], for example: AU
- State or Province Name (full name) [], for example: NSW
- Locality Name (e.g., city) [Default City], for example: Sydney
- Organization Name (e.g., company) [Default Company Ltd], for example: AmazonWebService
- Organizational Unit Name (e.g., section) [], for example: aws
- Common Name (e.g., your name or your server's hostname) [], for example: aws
- Email Address [], for example: abcd.efgh@amazonwebservices.com

Make sure this is not same as step 4. You can do this, for example, by changing Organizational Unit Name to a different name as shown.

Enter the additional attributes following to be sent with your certificate request.

- A challenge password [], for example: oracle123
- An optional company name [], for example: aws

12. Get the certificate signature.

```
openssl req -noout -text -in certdms.csr | grep -i signature
```

The signature key for this post is sha256WithRSAEncryption.

13. Run the command following to generate the certificate (.crt) file.

```
openssl x509 -req -in certdms.csr -CA self-rootCA.pem -CAkey self-rootCA.key  
-CAcreateserial -out certdms.crt -days 365 -sha256
```

This displays output like the following.

```
Signature ok  
subject=/C=AU/ST=NSW/L=Sydney/O=awsweb/OU=DBeng/CN=aws  
Getting CA Private Key
```

14. Add the certificate to the wallet.

```
orapki wallet add -wallet /u01/app/oracle/wallet -pwd oracle123 -user_cert -cert  
certdms.crt
```

15. View the wallet. It should have two entries. See the code following.

```
orapki wallet display -wallet /u01/app/oracle/wallet -pwd oracle123
```

16. Configure the sqlnet.ora file (\$ORACLE\_HOME/network/admin/sqlnet.ora).

```
WALLET_LOCATION =  
  (SOURCE =  
    (METHOD = FILE)  
    (METHOD_DATA =
```

```

        (DIRECTORY = /u01/app/oracle/wallet/
    )
)

SQLNET.AUTHENTICATION_SERVICES = (NONE)
SSL_VERSION = 1.0
SSL_CLIENT_AUTHENTICATION = FALSE
SSL_CIPHER_SUITES = (SSL_RSA_WITH_AES_256_CBC_SHA)

```

17. Stop the Oracle listener.

```
lsnrctl stop
```

18. Add entries for SSL in the `listener.ora` file (`$ORACLE_HOME/network/admin/listener.ora`).

```

SSL_CLIENT_AUTHENTICATION = FALSE
WALLET_LOCATION =
(SOURCE =
(METHOD = FILE)
(METHOD_DATA =
(DIRECTORY = /u01/app/oracle/wallet/
)
)
)

SID_LIST_LISTENER =
(SID_LIST =
(SID_DESC =
(GLOBAL_DBNAME = SID)
(ORACLE_HOME = ORACLE_HOME)
(SID_NAME = SID)
)
)
)

LISTENER =
(DESCRIPTION_LIST =
(DESCRIPTION =
(ADDRESS = (PROTOCOL = TCP)(HOST = localhost.localdomain)(PORT = 1521))
(ADDRESS = (PROTOCOL = TCPS)(HOST = localhost.localdomain)(PORT = 1522))
(ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC1521))
)
)
)
```

19. Configure the `tnsnames.ora` file (`$ORACLE_HOME/network/admin/tnsnames.ora`).

```

<SID>=
(DESCRIPTION=
(ADDRESS_LIST =
(ADDRESS=(PROTOCOL = TCP)(HOST = localhost.localdomain)(PORT = 1521))
)
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = <SID>)
)
)

<SID>_ssl=
(DESCRIPTION=
(ADDRESS_LIST =
(ADDRESS=(PROTOCOL = TCPS)(HOST = localhost.localdomain)(PORT = 1522))
)
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = <SID>)
)
)
```

```
)
```

20. Restart the Oracle listener.

```
lsnrctl start
```

21. Show the Oracle listener status.

```
lsnrctl status
```

22. Test the SSL connection to the database from localhost using sqlplus and the SSL tnsnames entry.

```
sqlplus -L ORACLE_USER@SID_ssl
```

23. Verify that you successfully connected using SSL.

```
SELECT SYS_CONTEXT('USERENV', 'network_protocol') FROM DUAL;
SYS_CONTEXT('USERENV', 'NETWORK_PROTOCOL')
-----
tcps
```

24. Change directory to the directory with the self-signed certificate.

```
cd /u01/app/oracle/self_signed_cert
```

25. Create a new client Oracle wallet for AWS DMS to use.

```
orapki wallet create -wallet ./ -auto_login_only
```

26. Add the self-signed root certificate to the Oracle wallet.

```
orapki wallet add -wallet ./ -trusted_cert -cert self-rootCA.pem -auto_login_only
```

27. List the contents of the Oracle wallet for AWS DMS to use. The list should include the self-signed root certificate.

```
orapki wallet display -wallet ./
```

This has output like the following.

```
Trusted Certificates:
Subject: CN=aws,OU=DBeng,O=AmazonWebService,L=Sydney,ST=NSW,C=AU
```

28. Upload the Oracle wallet that you just created to AWS DMS.

## Supported encryption methods for using Oracle as a source for AWS DMS

In the following table, you can find the transparent data encryption (TDE) methods that AWS DMS supports when working with an Oracle source database.

Redo logs access method	TDE tablespace	TDE column
Oracle LogMiner	Yes	Yes

Redo logs access method	TDE tablespace	TDE column
Binary Reader	Yes	Yes

AWS DMS supports Oracle TDE when using Binary Reader, on both the column level and the tablespace level. To use TDE encryption with AWS DMS, first identify the Oracle wallet location where the TDE encryption key and TDE password are stored. Then identify the correct TDE encryption key and password for your Oracle source endpoint.

### To identify and specify encryption key and password for TDE encryption

- Run the following query to find the Oracle encryption wallet on the Oracle database host.

```
SQL> SELECT WRL_PARAMETER FROM V$ENCRYPTION_WALLET;
WRL_PARAMETER
-----
/u01/oracle/product/12.2.0/dbhome_1/data/wallet/
```

Here, /u01/oracle/product/12.2.0/dbhome\_1/data/wallet/ is the wallet location.

- Get the master key ID using one of the following encryption options, depending on which one returns this value.
  - For table or column-level encryption, run the following queries.

```
SQL> SELECT OBJECT_ID FROM ALL_OBJECTS
WHERE OWNER='DMS_USER' AND OBJECT_NAME='TEST_TDE_COLUMN' AND OBJECT_TYPE='TABLE';
OBJECT_ID
-----
81046
SQL> SELECT MKEYID FROM SYS.ENC$ WHERE OBJ#=81046;
MKEYID
-----
AWGDC9g1Sk8Xv+3bVveiVSgAAAAAAAAAAAAAAAAAAAAAA
```

Here, AWGDC9g1Sk8Xv+3bVveiVSg is the master key ID (MKEYID). If you get a value for MKEYID, you can continue with Step 3. Otherwise, continue with Step 2.2.

#### Note

The trailing string 'A' characters (AAA...) is not part of the value.

- For tablespace-level encryption, run the following queries.

```
SQL> SELECT TABLESPACE_NAME, ENCRYPTED FROM dba tablespaces;
TABLESPACE_NAME          ENC
-----
SYSTEM                  NO
SYSAUX                 NO
UNDOTBS1                NO
TEMP                   NO
USERS                  NO
TEST_ENCRYT              YES
SQL> SELECT name,utl_raw.cast_to_varchar2( utl_encode.base64_encode('01'||substr(mkeyid,1,4)) ||
utl_raw.cast_to_varchar2( utl_encode.base64_encode(substr(mkeyid,5,length(mkeyid))) )|| masterkeyid_base64
```

```

FROM (SELECT t.name, RAWTOHEX(x.mkid) mkeyid FROM v$tablespace t, x$kcbtex WHERE
      t.ts#=x.ts#)
WHERE name = 'TEST_ENCRYT';

NAME          MASTERKEYID_BASE64
-----
TEST_ENCRYT   AWGDC9g1Sk8Xv+3bVveiVSg=

```

Here, AWGDC9g1Sk8Xv+3bVveiVSg is the master key ID (TEST\_ENCRYT). If both steps 2.1 and 2.2 return a value, they are always identical.

The trailing '=' character is not part of the value.

3. From the command line, list the encryption wallet entries on the source Oracle database host.

```

$ mkstore -wrl /u01/oracle/product/12.2.0/dbhome_1/data/wallet/ -list
Oracle Secret Store entries:
ORACLE.SECURITY.DB.ENCRYPTION.AWGDC9g1Sk8Xv+3bVveiVSgAAAAAAAAAAAAAAA
ORACLE.SECURITY.DB.ENCRYPTION.AY1mRA80XU9Qvzo3idU40H4AAAAAAAAAAAAAAA
ORACLE.SECURITY.DB.ENCRYPTION.MASTERKEY
ORACLE.SECURITY.ID.ENCRYPTION.
ORACLE.SECURITY.KB.ENCRYPTION.
ORACLE.SECURITY.KM.ENCRYPTION.AY1mRA80XU9Qvzo3idU40H4AAAAAAAAAAAAAAA

```

Find the entry containing the master key ID that you found in step 2 (AWGDC9g1Sk8Xv+3bVveiVSg). This entry is the TDE encryption key name.

4. View the details of the entry that you found in the previous step.

```

$ mkstore -wrl /u01/oracle/product/12.2.0/dbhome_1/data/wallet/ -viewEntry
ORACLE.SECURITY.DB.ENCRYPTION.AWGDC9g1Sk8Xv+3bVveiVSgAAAAAAAAAAAAAAA
Oracle Secret Store Tool : Version 12.2.0.1.0
Copyright (c) 2004, 2016, Oracle and/or its affiliates. All rights reserved.
Enter wallet password:
ORACLE.SECURITY.DB.ENCRYPTION.AWGDC9g1Sk8Xv+3bVveiVSgAAAAAAAAAAAAAAA
= AEMAASAASGYs0phWHfNt9J5mEMkkegGFID4LLfQszDojgDzbfoYDEACv0x3pJC+UGD/
PdtE2jLICBQcAeHgJChQGLA==

```

Enter the wallet password to see the result.

Here, the value to the right of '=' is the TDE password.

5. Specify the TDE encryption key name for the Oracle source endpoint by setting the securityDbEncryptionName extra connection attribute.

```
securityDbEncryptionName=ORACLE.SECURITY.DB.ENCRYPTION.AWGDC9g1Sk8Xv
+3bVveiVSgAAAAAAAAAAAAAAA
```

6. Provide the associated TDE password for this key on the console as part of the Oracle source's **Password** value. Use the following order to format the comma-separated password values, ended by the TDE password value.

```
Oracle_db_password,ASM_Password,AEMAASAASGYs0phWHfNt9J5mEMkkegGFID4LLfQszDojgDzbfoYDEACv0x3pJC
+UGD/PdtE2jLICBQcAeHgJChQGLA==
```

Specify the password values in this order regardless of your Oracle database configuration. For example, if you're using TDE but your Oracle database isn't using ASM, specify password values in the following comma-separated order.

```
Oracle_db_password, ,AEMAASAASGYs0phWHfNt9J5mEMkegGFID4LLfQszDojgDzbfoYDEACv0x3pJC+UGD/PdtE2jLICBQcAeHgJChQGLA==
```

If the TDE credentials you specify are incorrect, the AWS DMS migration task doesn't fail. However, the task also doesn't read or apply ongoing replication changes to the target database. After starting the task, monitor **Table statistics** on the console migration task page to make sure changes are replicated.

If a DBA changes the TDE credential values for the Oracle database while the task is running, the task fails. The error message contains the new TDE encryption key name. To specify new values and restart the task, use the preceding procedure.

**Important**

You can't manipulate a TDE wallet created in an Oracle Automatic Storage Management (ASM) location because OS level commands like `cp`, `mv`, `orapki`, and `mkstore` corrupt the wallet files stored in an ASM location. This restriction is specific to TDE wallet files stored in an ASM location only, but not for TDE wallet files stored in a local OS directory.

To manipulate a TDE wallet stored in ASM with OS level commands, create a local keystore and merge the ASM keystore into the local keystore as follows:

1. Create a local keystore.

```
ADMINISTER KEY MANAGEMENT create keystore file system wallet location
identified by wallet password;
```

2. Merge the ASM keystore into the local keystore.

```
ADMINISTER KEY MANAGEMENT merge keystore ASM wallet location identified
by wallet password into existing keystore file system wallet location
identified by wallet password with backup;
```

Then, to list the encryption wallet entries and TDE password, run steps 3 and 4 against the local keystore.

## Supported compression methods for using Oracle as a source for AWS DMS

In the following table, you can find which compression methods AWS DMS supports when working with an Oracle source database. As the table shows, compression support depends both on your Oracle database version and whether DMS is configured to use Oracle LogMiner to access the redo logs.

Version	Basic	OLTP	HCC (from Oracle 11g R2 or newer)	Others
Oracle 10	No	N/A	N/A	No
Oracle 11 or newer – Oracle LogMiner	Yes	Yes	Yes	Yes – Any compression method supported by Oracle LogMiner.

Version	Basic	OLTP	HCC (from Oracle 11g R2 or newer)	Others
Oracle 11 or newer – Binary Reader	Yes	Yes	Yes – For more information, see the following note .	Yes

#### Note

When the Oracle source endpoint is configured to use Binary Reader, the Query Low level of the HCC compression method is supported for full-load tasks only.

## Replicating nested tables using Oracle as a source for AWS DMS

AWS DMS supports the replication of Oracle tables containing columns that are nested tables or defined types. To enable this functionality, add the following extra connection attribute setting to the Oracle source endpoint.

```
allowSelectNestedTables=true;
```

AWS DMS creates the target tables from Oracle nested tables as regular parent and child tables on the target without a unique constraint. To access the correct data on the target, join the parent and child tables. To do this, first manually create a nonunique index on the NESTED\_TABLE\_ID column in the target child table. You can then use the NESTED\_TABLE\_ID column in the join ON clause together with the parent column that corresponds to the child table name. In addition, creating such an index improves performance when the target child table data is updated or deleted by AWS DMS. For an example, see [Example join for parent and child tables on the target \(p. 164\)](#).

We recommend that you configure the task to stop after a full load completes. Then, create these nonunique indexes for all the replicated child tables on the target and resume the task.

If a captured nested table is added to an existing parent table (captured or not captured), AWS DMS handles it correctly. However, the nonunique index for the corresponding target table isn't created. In this case, if the target child table becomes extremely large, performance might be affected. In such a case, we recommend that you stop the task, create the index, then resume the task.

After the nested tables are replicated to the target, have the DBA run a join on the parent and corresponding child tables to flatten the data.

### Prerequisites for replicating Oracle nested tables as a source

Ensure that you replicate parent tables for all the replicated nested tables. Include both the parent tables (the tables containing the nested table column) and the child (that is, nested) tables in the AWS DMS table mappings.

### Supported Oracle nested table types as a source

AWS DMS supports the following Oracle nested table types as a source:

- Data type
- User defined object

### Limitations of AWS DMS support for Oracle nested tables as a source

AWS DMS has the following limitations in its support of Oracle nested tables as a source:

- AWS DMS supports only one level of table nesting.
- AWS DMS table mapping doesn't check that both the parent and child table or tables are selected for replication. That is, it's possible to select a parent table without a child or a child table without a parent.

## How AWS DMS replicates Oracle nested tables as a source

AWS DMS replicates parent and nested tables to the target as follows:

- AWS DMS creates the parent table identical to the source. It then defines the nested column in the parent as RAW(16) and includes a reference to the parent's nested tables in its NESTED\_TABLE\_ID column.
- AWS DMS creates the child table identical to the nested source, but with an additional column named NESTED\_TABLE\_ID. This column has the same type and value as the corresponding parent nested column and has the same meaning.

## Example join for parent and child tables on the target

To flatten the parent table, run a join between the parent and child tables, as shown in the following example:

1. Create the Type table.

```
CREATE OR REPLACE TYPE NESTED_TEST_T AS TABLE OF VARCHAR(50);
```

2. Create the parent table with a column of type NESTED\_TEST\_T as defined preceding.

```
CREATE TABLE NESTED_PARENT_TEST (ID NUMBER(10,0) PRIMARY KEY, NAME NESTED_TEST_T) NESTED  
TABLE NAME STORE AS NAME_KEY;
```

3. Flatten the table NESTED\_PARENT\_TEST using a join with the NAME\_KEY child table where CHILD.NESTED\_TABLE\_ID matches PARENT.NAME.

```
SELECT ... FROM NESTED_PARENT_TEST PARENT, NAME_KEY CHILD WHERE CHILD.NESTED_  
TABLE_ID = PARENT.NAME;
```

## Storing REDO on Oracle ASM when using Oracle as a source for AWS DMS

For Oracle sources with high REDO generation, storing REDO on Oracle ASM can benefit performance, especially in a RAC configuration since you can configure DMS to distribute ASM REDO reads across all ASM nodes.

To utilize this configuration, use the `asmServer` connection attribute. For example, the following connection string distributes DMS REDO reads across 3 ASM nodes:

```
asmServer=(DESCRIPTION=  
(ADDRESS_LIST=(CONNECT_TIMEOUT=8)(ENABLE=BROKEN)(LOAD_BALANCE=ON)(FAILOVER=ON)  
(ADDRESS=(PROTOCOL=tcp)(HOST=asm_node1_ip_address)(PORT=asm_node1_port_number))  
(ADDRESS=(PROTOCOL=tcp)(HOST=asm_node2_ip_address)(PORT=asm_node2_port_number))  
(ADDRESS=(PROTOCOL=tcp)(HOST=asm_node3_ip_address)(PORT=asm_node3_port_number)))  
(CONNECT_DATA=(SERVICE_NAME=+ASM)))
```

When using NFS to store Oracle REDO, it's important to make sure that applicable DNFS (direct NFS) client patches are applied, specifically any patch addressing Oracle bug 25224242. For additional information, review the following Oracle Publication regarding Direct NFS client related patches, [Recommended Patches for Direct NFS Client](#).

Additionally, to improve NFS read performance, we recommended you increase the value of `rsize` and `wsize` in `fstab` for the the NFS volume, as shown in the following example.

```
NAS_name_here:/ora_DATA1_archive /u09/oradata/DATA1 nfs
rw,bg,hard,nointr,tcp,nfsvers=3,_netdev,
timeo=600,rsize=262144,wsize=262144
```

Also, adjust the `tcp-max-xfer-size` value as follows:

```
vserver nfs modify -vserver vserver -tcp-max-xfer-size 262144
```

## Extra connection attributes when using Oracle as a source for AWS DMS

You can use extra connection attributes to configure your Oracle source. You specify these settings when you create the source endpoint. If you have multiple connection attribute settings, separate them from each other by semicolons with no additional white space (for example, `oneSetting=theValue;thenAnother=anotherValue`).

The following table shows the extra connection attributes that you can use to configure an Oracle database as a source for AWS DMS.

Name	Description
escapeCharacter	<p>Set this attribute to an escape character. This escape character allows you to make a single wildcard character behave as a normal character in table mapping expressions. For more information, see <a href="#">Wildcards in table mapping (p. 499)</a>.</p> <p>Default value: Null</p> <p>Valid values: Any character other than a wildcard character</p> <p>Example: <code>escapeCharacter=#</code></p>
addSupplementalLogging	<p>Set this attribute to set up table-level supplemental logging for the Oracle database. This attribute enables PRIMARY KEY supplemental logging on all tables selected for a migration task.</p> <p>Default value: N</p> <p>Valid values: Y/N</p> <p>Example: <code>addSupplementalLogging=Y;</code></p> <p><b>Note</b> If you use this option, you still need to enable database-level supplemental logging as discussed previously.</p>
additionalArchivedLogDestId	<p>Set this attribute with <code>archivedLogDestId</code> in a primary-Standby setup. This attribute is useful in a switchover when Oracle Data Guard database is used as a source. In this case, AWS DMS needs to know which destination</p>

Name	Description
	<p>to get archive redo logs from to read changes. This is because the previous primary is now a Standby instance after switchover.</p> <p>Although AWS DMS supports the use of the Oracle RESETLOGS option to open the database, never use RESETLOGS unless necessary. For additional information about RESETLOGS, see <a href="#">RMAN Data Repair Concepts</a> in the <i>Oracle® Database Backup and Recovery User's Guide</i>.</p> <p>Default value: N/A</p> <p>Valid values : Archive destination Ids</p> <p>Example: additionalArchivedLogDestId=2;</p>
allowSelectNestedTables	<p>Set this attribute to true to enable replication of Oracle tables containing columns that are nested tables or defined types. For more information, see <a href="#">Replicating nested tables using Oracle as a source for AWS DMS (p. 163)</a>.</p> <p>Default value: false</p> <p>Valid values: true/false</p> <p>Example: allowSelectNestedTables=true;</p>
ExtraArchivedLogDestIds	<p>Specifies the IDs of one or more destinations for one or more archived redo logs. These IDs are the values of the dest_id column in the v\$archived_log view. Use this setting with the archivedLogDestId extra connection attribute in a primary-to-single setup or a primary-to-multiple-standby setup.</p> <p>This setting is useful in a switchover when you use an Oracle Data Guard database as a source. In this case, AWS DMS needs information about what destination to get archive redo logs from to read changes. AWS DMS needs this because after the switchover the previous primary is a standby instance.</p> <p>Defalut value: N/A</p> <p>Valid values: Archive destination Ids</p> <p>Example:</p> <pre>archivedLogDestId=1;ExtraArchivedLogDestIds=[2,3,4]</pre>
useLogminerReader	<p>Set this attribute to Y to capture change data using the LogMiner utility (the default). Set this option to N if you want AWS DMS to access the redo logs as a binary file. When you set this option to N, also add the setting useBfile=Y. For more information on this setting and using Oracle Automatic Storage Management (ASM), see <a href="#">Using Oracle LogMiner or AWS DMS Binary Reader for CDC (p. 135)</a>.</p> <p>Default value: Y</p> <p>Valid values: Y/N</p> <p>Example: useLogminerReader=N;useBfile=Y;</p>

Name	Description
useBfile	<p>Set this attribute to Y in order to capture change data using the Binary Reader utility. Set useLogminerReader to N to set this attribute to Y. To use the Binary Reader with an Amazon RDS for Oracle as the source, you set additional attributes. For more information on this setting and using Oracle Automatic Storage Management (ASM), see <a href="#">Using Oracle LogMiner or AWS DMS Binary Reader for CDC (p. 135)</a>.</p> <p>Default value: N</p> <p>Valid values: Y/N</p> <p>Example: useLogminerReader=N;useBfile=Y;</p>
parallelASMReadThreads	<p>Set this attribute to change the number of threads that DMS configures to perform change data capture (CDC) using Oracle Automatic Storage Management (ASM). You can specify an integer value between 2 (the default) and 8 (the maximum). Use this attribute together with the readAheadBlocks attribute. For more information, see <a href="#">Configuring a CDC task to use Binary Reader with an RDS for Oracle source for AWS DMS (p. 150)</a>.</p> <p>Default value: 2</p> <p>Valid values: An integer from 2 to 8</p> <p>Example: parallelASMReadThreads=6;readAheadBlocks=150000;</p>
readAheadBlocks	<p>Set this attribute to change the number of read-ahead blocks that DMS configures to perform CDC using Oracle Automatic Storage Management (ASM) and non-ASM NAS storage. You can specify an integer value between 1000 (the default) and 200,000 (the maximum). Use this attribute together with the parallelASMReadThreads attribute. For more information, see <a href="#">Configuring a CDC task to use Binary Reader with an RDS for Oracle source for AWS DMS (p. 150)</a>.</p> <p>Default value: 1000</p> <p>Valid values: An integer from 1000 to 200,000</p> <p>Example: parallelASMReadThreads=6;readAheadBlocks=150000;</p>
accessAlternateDirectly	<p>Set this attribute to false in order to use the Binary Reader to capture change data for an Amazon RDS for Oracle as the source. This tells the DMS instance to not access redo logs through any specified path prefix replacement using direct file access. For more information, see <a href="#">Configuring a CDC task to use Binary Reader with an RDS for Oracle source for AWS DMS (p. 150)</a>.</p> <p>Default value: true</p> <p>Valid values: true/false</p> <p>Example: useLogminerReader=N;useBfile=Y;accessAlternateDirectly=false;</p>

Name	Description
useAlternateFolderForOnline	<p>Set this attribute to true in order to use the Binary Reader to capture change data for an Amazon RDS for Oracle as the source. This tells the DMS instance to use any specified prefix replacement to access all online redo logs. For more information, see <a href="#">Configuring a CDC task to use Binary Reader with an RDS for Oracle source for AWS DMS (p. 150)</a>.</p> <p>Default value: false</p> <p>Valid values: true/false</p> <p>Example:  <code>useLogminerReader=N;useBfile=Y;accessAlternateDirectly=false;useAlternateFolderForOnline=true;</code></p>
oraclePathPrefix	<p>Set this string attribute to the required value in order to use the Binary Reader to capture change data for an Amazon RDS for Oracle as the source. This value specifies the default Oracle root used to access the redo logs. For more information, see <a href="#">Configuring a CDC task to use Binary Reader with an RDS for Oracle source for AWS DMS (p. 150)</a>.</p> <p>Default value: none</p> <p>Valid value: /rdsdbdata/db/ORCL_A/</p> <p>Example:  <code>useLogminerReader=N;useBfile=Y;accessAlternateDirectly=false;useAlternateFolderForOnline=true;oraclePathPrefix=/rdsdbdata/db/ORCL_A/;</code></p>
usePathPrefix	<p>Set this string attribute to the required value in order to use the Binary Reader to capture change data for an Amazon RDS for Oracle as the source. This value specifies the path prefix used to replace the default Oracle root to access the redo logs. For more information, see <a href="#">Configuring a CDC task to use Binary Reader with an RDS for Oracle source for AWS DMS (p. 150)</a>.</p> <p>Default value: none</p> <p>Valid value: /rdsdbdata/log/</p> <p>Example:  <code>useLogminerReader=N;useBfile=Y;accessAlternateDirectly=false;useAlternateFolderForOnline=true;oraclePathPrefix=/rdsdbdata/db/ORCL_A/; usePathPrefix=/rdsdbdata/log/;</code></p>

Name	Description
replacePathPrefix	<p>Set this attribute to true in order to use the Binary Reader to capture change data for an Amazon RDS for Oracle as the source. This setting tells DMS instance to replace the default Oracle root with the specified usePathPrefix setting to access the redo logs. For more information, see <a href="#">Configuring a CDC task to use Binary Reader with an RDS for Oracle source for AWS DMS (p. 150)</a>.</p> <p>Default value: false</p> <p>Valid values: true/false</p> <p>Example:  <code>useLogminerReader=N;useBfile=Y;accessAlternateDirectly=false;useAlternateFolderForOnline=true;oraclePathPrefix=/rdsdbdata/db/ORCL_A/; usePathPrefix=/rdsdbdata/log/;replacePathPrefix=true;</code></p>
retryInterval	<p>Specifies the number of seconds that the system waits before resending a query.</p> <p>Default value: 5</p> <p>Valid values: Numbers starting from 1</p> <p>Example: <code>retryInterval=6;</code></p>
archivedLogDestId	<p>Specifies the ID of the destination for the archived redo logs. This value should be the same as a number in the dest_id column of the v\$archived_log view. If you work with an additional redo log destination, we recommend that you use the additionalArchivedLogDestId attribute to specify the additional destination ID. Doing this improves performance by ensuring that the correct logs are accessed from the outset.</p> <p>Default value: 1</p> <p>Valid values: Number</p> <p>Example: <code>archivedLogDestId=1;</code></p>
archivedLogsOnly	<p>When this field is set to Y, AWS DMS only accesses the archived redo logs. If the archived redo logs are stored on Oracle ASM only, the AWS DMS user account needs to be granted ASM privileges.</p> <p>Default value: N</p> <p>Valid values: Y/N</p> <p>Example: <code>archivedLogsOnly=Y;</code></p>

Name	Description
numberDataTypeScale	<p>Specifies the number scale. You can select a scale up to 38, or you can select -1 for FLOAT, or -2 for VARCHAR. By default, the NUMBER data type is converted to precision 38, scale 10.</p> <p>Default value: 10</p> <p>Valid values: -2 to 38 (-2 for VARCHAR, -1 for FLOAT)</p> <p>Example: <code>numberDataTypeScale=12</code></p> <p><b>Note</b>            Select a precision-scale combination, -1 (FLOAT) or -2 (VARCHAR). DMS supports any precision-scale combination supported by Oracle. If precision is 39 or above, select -2 (VARCHAR). The numberDataTypeScale setting for the Oracle database is used for the NUMBER data type only (without the explicit precision and scale definition).</p>
failTasksOnLobTruncation	<p>When set to true, this attribute causes a task to fail if the actual size of an LOB column is greater than the specified LobMaxSize.</p> <p>If a task is set to limited LOB mode and this option is set to true, the task fails instead of truncating the LOB data.</p> <p>Default value: false</p> <p>Valid values: Boolean</p> <p>Example: <code>failTasksOnLobTruncation=true;</code></p>
readTableSpaceName	<p>When set to true, this attribute supports tablespace replication.</p> <p>Default value: false</p> <p>Valid values: Boolean</p> <p>Example: <code>readTableSpaceName=true;</code></p>
enableHomogenousTablespaces	<p>Set this attribute to enable homogenous tablespace replication and create existing tables or indexes under the same tablespace on the target.</p> <p>Default value: false</p> <p>Valid values: true/false</p> <p>Example: <code>enableHomogenousTablespaces=true</code></p>

Name	Description
standbyDelayTime	<p>Use this attribute to specify a time in minutes for the delay in standby sync. If the source is an Active Data Guard standby database, use this attribute to specify the time lag between primary and standby databases.</p> <p>In AWS DMS, you can create an Oracle CDC task that uses an Active Data Guard standby instance as a source for replicating ongoing changes. Doing this eliminates the need to connect to an active database that might be in production.</p> <p>Default value:0</p> <p>Valid values: Number</p> <p>Example: <code>standbyDelayTime=1;</code></p> <p><b>Note:</b> When using DMS 3.4.6, 3.4.7 and higher, use of this connection setting is optional. In the latest version of DMS 3.4.6 and version 3.4.7, <i>dms_user</i> should have <code>select</code> permission on <code>V_\$DATAGUARD_STATS</code>, allowing DMS to calculate standby delay time.</p>
securityDbEncryptionName	<p>Specifies the name of a key used for the transparent data encryption (TDE) of the columns and tablespace in the Oracle source database. For more information on setting this attribute and its associated password on the Oracle source endpoint, see <a href="#">Supported encryption methods for using Oracle as a source for AWS DMS (p. 159)</a>.</p> <p>Default value: ""</p> <p>Valid values: String</p> <p><code>securityDbEncryptionName=ORACLE.SECURITY.DB.ENCRYPTION.Adg8m2dhkU/0v/m5QUaaNJEAAAAAAAAAAAAAA</code></p>
spatialSdo2GeoJsonFunctionName	<p><b>For Oracle Database 12.1 or earlier sources migrating to PostgreSQL targets,</b> use this attribute to convert SDO_GEOmetry to GEOJSON format.</p> <p>By default, AWS DMS calls the SD02GEOJSON custom function which must be present and accessible to the AWS DMS user. Or you can create your own custom function that mimics the operation of SD0GEOJSON and set <code>spatialSdo2GeoJsonFunctionName</code> to call it instead.</p> <p>Default value: SDO2GEOJSON</p> <p>Valid values: String</p> <p>Example: <code>spatialSdo2GeoJsonFunctionName=myCustomSD02GEOJSONFunction;</code></p>
exposeViews	<p>Use this attribute to pull data once from a view; you can't use it for ongoing replication. When you extract data from a view, the view is shown as a table on the target schema.</p> <p>Default value: false</p> <p>Valid values: true/false</p> <p>Example: <code>exposeViews=true</code></p>

Name	Description
enableHomogenousPartitionOps	<p>Set this attribute to true to enable replication of Oracle Partition and subPartition DDL Operations for Oracle <i>Homogenous</i> migration.</p> <p>Note that this feature and enhancement was introduced in AWS DMS version 3.4.7.</p> <p>Default value: false</p> <p>Valid values: true/false</p> <p>Example: enableHomogenousPartitionOps=true</p>

## Source data types for Oracle

The Oracle endpoint for AWS DMS supports most Oracle data types. The following table shows the Oracle source data types that are supported when using AWS DMS and the default mapping to AWS DMS data types.

### Note

With the exception of the LONG and LONG RAW data types, when replicating from an Oracle source to an Oracle target (a *homogeneous replication*), all of the source and target data types will be identical. But the LONG data type will be mapped to CLOB and the LONG RAW data type will be mapped to BLOB.

For information on how to view the data type that is mapped in the target, see the section for the target endpoint you are using.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

Oracle data type	AWS DMS data type
BINARY_FLOAT	REAL4
BINARY_DOUBLE	REAL8
BINARY	BYTES
FLOAT (P)	If precision is less than or equal to 24, use REAL4. If precision is greater than 24, use REAL8.
NUMBER (P,S)	When scale is greater than 0, use REAL8
NUMBER according to the numberDataTypeScale extra connection attribute.	When scale is 0: <ul style="list-style-type: none"><li>• And precision is less than or equal to 2, use INT1.</li><li>• And precision is greater than 2 and less than or equal to 4, use INT2.</li><li>• And precision is greater than 4 and less than or equal to 9, use INT4.</li><li>• And precision is greater than 9, use NUMERIC.</li><li>• And precision is greater than or equal to scale, use NUMERIC.</li></ul> In all other cases, use REAL8.
DATE	DATETIME

Oracle data type	AWS DMS data type
INTERVAL_YEAR TO MONTH	STRING (with interval year_to_month indication)
INTERVAL_DAY TO SECOND	STRING (with interval day_to_second indication)
TIMESTAMP	DATETIME
TIMESTAMP WITH TIME ZONE	STRING (with timestamp_with_timezone indication)
TIMESTAMP WITH LOCAL TIME ZONE	STRING (with timestamp_with_local_timezone indication)
CHAR	STRING
VARCHAR2	STRING
NCHAR	WSTRING
NVARCHAR2	WSTRING
RAW	BYTES
REAL	REAL8
BLOB	<p>BLOB</p> <p>To use this data type with AWS DMS, you must enable the use of BLOB data types for a specific task. AWS DMS supports BLOB data types only in tables that include a primary key.</p>
CLOB	<p>CLOB</p> <p>To use this data type with AWS DMS, you must enable the use of CLOB data types for a specific task. During CDC, AWS DMS supports CLOB data types only in tables that include a primary key.</p>
NCLOB	<p>NCLOB</p> <p>To use this data type with AWS DMS, you must enable the use of NCLOB data types for a specific task. During CDC, AWS DMS supports NCLOB data types only in tables that include a primary key.</p>
LONG	<p>CLOB</p> <p>The LONG data type isn't supported in batch-optimized apply mode (TurboStream CDC mode).</p> <p>To use this data type with AWS DMS, enable the use of LOBs for a specific task.</p> <p>During CDC or full load, AWS DMS supports LOB data types only in tables that have a primary key.</p> <p>Also, AWS DMS doesn't support full LOB mode for loading LONG columns. Instead, you can use limited LOB mode for migrating LONG columns to an Oracle target. In limited LOB mode, AWS DMS truncates any data to 64 KB that you set to LONG columns longer than 64 KB.</p> <p>For more information about LOB support in AWS DMS, see <a href="#">Setting LOB support for source databases in an AWS DMS task (p. 479)</a></p>

Oracle data type	AWS DMS data type
LONG RAW	<p>BLOB</p> <p>The LONG RAW data type isn't supported in batch-optimized apply mode (TurboStream CDC mode).</p> <p>To use this data type with AWS DMS, enable the use of LOBs for a specific task.</p> <p>During CDC or full load, AWS DMS supports LOB data types only in tables that have a primary key.</p> <p>Also, AWS DMS doesn't support full LOB mode for loading LONG RAW columns. Instead, you can use limited LOB mode for migrating LONG RAW columns to an Oracle target. In limited LOB mode, AWS DMS truncates any data to 64 KB that you set to LONG RAW columns longer than 64 KB. For more information about LOB support in AWS DMS, see <a href="#">Setting LOB support for source databases in an AWS DMS task (p. 479)</a></p>
XMLTYPE	CLOB
SDO_GEOGRAPHY	<p>BLOB (when an Oracle to Oracle migration)</p> <p>CLOB (when an Oracle to PostgreSQL migration)</p>

Oracle tables used as a source with columns of the following data types aren't supported and can't be replicated. Replicating columns with these data types result in a null column.

- BFILE
- ROWID
- REF
- UROWID
- User-defined data types
- ANYDATA
- VARRAY

**Note**

Virtual columns aren't supported.

## Migrating Oracle spatial data types

*Spatial data* identifies the geometry information for an object or location in space. In an Oracle database, the geometric description of a spatial object is stored in an object of type SDO\_GEOGRAPHY. Within this object, the geometric description is stored in a single row in a single column of a user-defined table.

AWS DMS supports migrating the Oracle type SDO\_GEOGRAPHY from an Oracle source to either an Oracle or PostgreSQL target.

When you migrate Oracle spatial data types using AWS DMS, be aware of these considerations:

- When migrating to an Oracle target, make sure to manually transfer USER\_SDO\_GEO\_METADATA entries that include type information.

- When migrating from an Oracle source endpoint to a PostgreSQL target endpoint, AWS DMS creates target columns. These columns have default geometry and geography type information with a 2D dimension and a spatial reference identifier (SRID) equal to zero (0). An example is GEOMETRY, 2, 0.
- For Oracle version 12.1 or earlier sources migrating to PostgreSQL targets, convert SDO\_GEOGRAPHY objects to GEOJSON format by using the SDO2GEOJSON function, or the spatialSdo2GeoJsonFunctionName extra connection attribute. For more information, see [Extra connection attributes when using Oracle as a source for AWS DMS \(p. 165\)](#).
- AWS DMS supports Oracle Spatial Column migrations for Full LOB mode only. AWS DMS doesn't support Limited LOB or Inline LOB modes. For more information about LOB mode, see [Setting LOB support for source databases in an AWS DMS task \(p. 479\)](#).
- Because AWS DMS only supports Full LOB mode for migrating Oracle Spatial Columns, the columns' table needs a primary key and a unique key. If the table doesn't have a primary key and a unique key, the table is skipped from migration.

## Using a Microsoft SQL Server database as a source for AWS DMS

Migrate data from one or many Microsoft SQL Server databases using AWS DMS. With a SQL Server database as a source, you can migrate data to another SQL Server database, or to one of the other AWS DMS supported databases. The following lists SQL Server editions you can use as a source with on-premises databases.

SQL Server version	Full load	Ongoing replication (CDC)
2005, 2008, 2008R2, 2012, 2014, 2016, 2017, 2019	Enterprise Standard Workgroup Developer Web	Enterprise Edition Developer Standard Edition (version 2016 SP1 and later)

When using SQL Server 2005 as a source, only Full Load is supported.

The following lists SQL Server editions you can use as a source with Amazon RDS databases.

SQL Server version	Full load	Ongoing replication (CDC)
2012, 2014, 2016, 2017, and 2019	Enterprise Standard	Enterprise Edition Standard Edition (version 2016 SP1 and later)

### Note

Support for Microsoft SQL Server version 2019 as a source is available.

The source SQL Server database can be installed on any computer in your network. A SQL Server account with appropriate access privileges to the source database for the type of task you chose is required for use with AWS DMS. This account must have the `view definition` and `view server state` permissions. You add this permission using the following command:

```
grant view definition to [user]
grant view server state to [user]
```

AWS DMS supports migrating data from named instances of SQL Server. You can use the following notation in the server name when you create the source endpoint.

```
IPAddress\InstanceName
```

For example, the following is a correct source endpoint server name. Here, the first part of the name is the IP address of the server, and the second part is the SQL Server instance name (in this example, SQLTest).

```
10.0.0.25\SQLTest
```

Also, obtain the port number that your named instance of SQL Server listens on, and use it to configure your AWS DMS source endpoint.

**Note**

Port 1433 is the default for Microsoft SQL Server. But dynamic ports that change each time SQL Server is started, and specific static port numbers used to connect to SQL Server through a firewall are also often used. So, you want to know the actual port number of your named instance of SQL Server when you create the AWS DMS source endpoint.

You can use SSL to encrypt connections between your SQL Server endpoint and the replication instance. For more information on using SSL with a SQL Server endpoint, see [Using SSL with AWS Database Migration Service \(p. 647\)](#).

For additional details on working with SQL Server source databases and AWS DMS, see the following.

**Topics**

- [Limitations on using SQL Server as a source for AWS DMS \(p. 176\)](#)
- [Permissions for full load only tasks \(p. 178\)](#)
- [Prerequisites for using ongoing replication \(CDC\) from a SQL Server source \(p. 179\)](#)
- [Capturing data changes for self-managed SQL Server on-premises or on Amazon EC2 \(p. 179\)](#)
- [Setting up ongoing replication on a cloud SQL Server DB instance \(p. 183\)](#)
- [Recommended settings when using Amazon RDS for SQL Server as a source for AWS DMS \(p. 185\)](#)
- [Supported compression methods for SQL Server \(p. 186\)](#)
- [Working with self-managed SQL Server AlwaysOn availability groups \(p. 186\)](#)
- [Endpoint settings when using SQL Server as a source for AWS DMS \(p. 188\)](#)
- [Source data types for SQL Server \(p. 192\)](#)

## Limitations on using SQL Server as a source for AWS DMS

The following limitations apply when using a SQL Server database as a source for AWS DMS:

- The identity property for a column isn't migrated to a target database column.
- The SQL Server endpoint doesn't support the use of sparse tables.
- Windows Authentication isn't supported.
- Changes to computed fields in a SQL Server aren't replicated.
- Temporal tables aren't supported.

- SQL Server partition switching isn't supported.
- When using the WRITETEXT and UPDATETEXT utilities, AWS DMS doesn't capture events applied on the source database.
- The following data manipulation language (DML) pattern isn't supported.

```
SELECT * INTO new_table FROM existing_table
```

- When using SQL Server as a source, column-level encryption isn't supported.
- Transparent Data Encryption (TDE) enabled at the database level is supported.
- AWS DMS doesn't support server level audits on SQL Server 2008 or SQL Server 2008 R2 as sources. This is because of a known issue with SQL Server 2008 and 2008 R2. For example, running the following command causes AWS DMS to fail.

```
USE [master]
GO
ALTER SERVER AUDIT [my_audit_test-20140710] WITH (STATE=on)
GO
```

- Geometry columns aren't supported in full lob mode when using SQL Server as a source. Instead, use limited lob mode or set the InlineLobMaxSize task setting to use inline lob mode.
- When using a Microsoft SQL Server source database in a replication task, the SQL Server Replication Publisher definitions aren't removed if you remove the task. A Microsoft SQL Server system administrator must delete those definitions from Microsoft SQL Server.
- Migrating data from schema-bound and non-schema-bound views is supported for full-load only tasks.
- Renaming tables using sp\_rename isn't supported (for example, sp\_rename 'Sales.SalesRegion', 'SalesReg;')
- Renaming columns using sp\_rename isn't supported (for example, sp\_rename 'Sales.Sales.Region', 'RegID', 'COLUMN';)
- AWS DMS doesn't support change processing to set and unset column default values (using the ALTER COLUMN SET DEFAULT clause with ALTER TABLE statements).
- AWS DMS doesn't support change processing to set column nullability (using the ALTER COLUMN [SET|DROP] NOT NULL clause with ALTER TABLE statements).
- With SQL Server 2012 and SQL Server 2014, when using DMS replication with Availability Groups, the distribution database can't be placed in an availability group. SQL 2016 supports placing the distribution database into an availability group, except for distribution databases used in merge, bidirectional, or peer-to-peer replication topologies.
- For partitioned tables, AWS DMS doesn't support different data compression settings for each partition.
- When inserting a value into SQL Server spatial data types (GEOGRAPHY and GEOMETRY), you can either ignore the spatial reference system identifier (SRID) property or specify a different number. When replicating tables with spatial data types, AWS DMS replaces the SRID with the default SRID (0 for GEOMETRY and 4326 for GEOGRAPHY).
- If your database isn't configured for MS-REPLICATION or MS-CDC, you can still capture tables that do not have a Primary Key, but only INSERT/DELETE DML events are captured. UPDATE and TRUNCATE TABLE events are ignored.
- Columnstore indexes aren't supported.
- Memory-optimized tables (using In-Memory OLTP) aren't supported.
- When replicating a table with a primary key that consists of multiple columns, updating the primary key columns during full load isn't supported.
- Delayed durability isn't supported.

- The `readBackupOnly=Y` endpoint setting (extra connection attribute) doesn't work on RDS for SQL Server source instances because of the way RDS performs backups.
- `EXCLUSIVE_AUTOMATIC_TRUNCATION` doesn't work on Amazon RDS SQL Server source instances because RDS users don't have access to run the SQL Server stored procedure, `sp_repldone`.
- AWS DMS doesn't capture truncate commands.
- AWS DMS doesn't support replication from databases with accelerated database recovery (ADR) turned on.
- AWS DMS doesn't support capturing data definition language (DDL) and data manipulation language (DML) statements within a single transaction.
- AWS DMS doesn't support the replication of data-tier application packages (DACPAC).
- UPDATE statements that involve primary keys or unique indexes and update multiple data rows, can cause conflicts when you apply changes to the target database. This might happen, for example, when the target database applies updates as INSERT and DELETE statements instead of a single UPDATE statement. With the batch optimized apply mode, the table might be ignored. With the transactional apply mode, the UPDATE operation might result in constraint violations. To avoid this issue, reload the relevant table. Alternatively, locate the problematic records in the Apply Exceptions control table (`dmslogs.awsdms_apply_exceptions`) and edit them manually in the target database. For more information, see [Change processing tuning settings \(p. 465\)](#).
- AWS DMS doesn't support the replication of tables and schemas, where the name includes a special character from the following set.

```
\\" -- \n \" \b \r ' \t ;
```

- Data masking isn't supported. AWS DMS migrates masked data without masking.
- AWS DMS replicates up to 32,767 tables with primary keys and up to 1,000 columns for each table. This is because AWS DMS creates a SQL Server replication article for each replicated table, and SQL Server replication articles have these limitations.
- When using Change Data Capture (CDC), you must define all columns that make up a unique index as NOT NULL. If this requirement is not met, SQL Server system error 22838 will result.

The following limitations apply when accessing the backup transaction logs:

- Encrypted backups aren't supported.
- Backups stored at a URL or on Windows Azure aren't supported.

The following limitations apply when accessing the backup transaction logs at file level:

- The backup transaction logs must reside in a shared folder with the appropriate permissions and access rights.
- Active transaction logs are accessed through the Microsoft SQL Server API (and not at file-level).
- UNIX platforms aren't supported.
- Reading the backup logs from multiple stripes isn't supported.
- Microsoft SQL Server backup to multiple disks (for example, MIRROR TO DISK) isn't supported.

## Permissions for full load only tasks

The following permissions are required to perform full load only tasks.

```
USE db_name;

CREATE USER dms_user FOR LOGIN dms_user;
ALTER ROLE [db_datareader] ADD MEMBER dms_user;
```

```
GRANT VIEW DATABASE STATE to dms_user ;  
USE master;  
GRANT VIEW SERVER STATE TO dms_user;
```

## Prerequisites for using ongoing replication (CDC) from a SQL Server source

You can use ongoing replication (change data capture, or CDC) for a self-managed SQL Server database on-premises or on Amazon EC2, or a cloud database such as Amazon RDS or a Microsoft Azure SQL managed instance.

The following requirements apply specifically when using ongoing replication with a SQL Server database as a source for AWS DMS:

- SQL Server must be configured for full backups, and you must perform a backup before beginning to replicate data.
- The recovery model must be set to **Bulk logged** or **Full**.
- SQL Server backup to multiple disks isn't supported. If the backup is defined to write the database backup to multiple files over different disks, AWS DMS can't read the data and the AWS DMS task fails.
- For self-managed SQL Server sources, SQL Server Replication Publisher definitions for the source used in a DMS CDC task aren't removed when you remove the task. A SQL Server system administrator must delete these definitions from SQL Server for self-managed sources.
- During CDC, AWS DMS needs to look up SQL Server transaction log backups to read changes. AWS DMS doesn't support SQL Server transaction log backups created using third-party backup software that *aren't* in native format. To support transaction log backups that *are* in native format and created using third-party backup software, add the `use3rdPartyBackupDevice=Y` connection attribute to the source endpoint.
- For self-managed SQL Server sources, be aware that SQL Server doesn't capture changes on newly created tables until they've been published. When tables are added to a SQL Server source, AWS DMS manages creating the publication. However, this process might take several minutes. Operations made to newly created tables during this delay aren't captured or replicated to the target.
- AWS DMS change data capture requires full transaction logging to be turned on in SQL Server. To turn on full transaction logging in SQL Server, either enable MS-REPLICATION or CHANGE DATA CAPTURE (CDC).
- SQL Server *tlog* entries won't be marked for re-use until the MS CDC capture job processes those changes.
- CDC operations aren't supported on memory-optimized tables. This limitation applies to SQL Server 2014 (when the feature was first introduced) and later.

## Capturing data changes for self-managed SQL Server on-premises or on Amazon EC2

To capture changes from a source Microsoft SQL Server database, make sure that the database is configured for full backups. Configure the database either in full recovery mode or bulk-logged mode.

For a self-managed SQL Server source, AWS DMS uses the following:

### MS-Replication

To capture changes for tables with primary keys. You can configure this automatically by giving sysadmin privileges to the AWS DMS endpoint user on the source SQL Server instance. Or you can

follow the steps in this section to prepare the source and use a user that doesn't have sysadmin privileges for the AWS DMS endpoint.

#### MS-CDC

To capture changes for tables without primary keys. Enable MS-CDC at the database level and for all of the tables individually.

When setting up a SQL Server database for ongoing replication (CDC), you can do one of the following:

- Set up ongoing replication using the sysadmin role.
- Set up ongoing replication to not use the sysadmin role.

#### Setting up ongoing replication using the sysadmin role with self-managed SQL Server

AWS DMS ongoing replication for SQL Server uses native SQL Server replication for tables with primary keys, and change data capture (CDC) for tables without primary keys.

Before setting up ongoing replication, see [Prerequisites for using ongoing replication \(CDC\) from a SQL Server source \(p. 179\)](#).

For tables with primary keys, AWS DMS can generally configure the required artifacts on the source. However, for SQL Server source instances that are self-managed, make sure to first configure the SQL Server distribution manually. After you do so, AWS DMS source users with sysadmin permission can automatically create the publication for tables with primary keys.

To check if distribution has already been configured, run the following command.

```
sp_get_distributor
```

If the result is NULL for column distribution, distribution isn't configured. You can use the following procedure to set up distribution.

#### To set up distribution

1. Connect to your SQL Server source database using the SQL Server Management Studio (SSMS) tool.
2. Open the context (right-click) menu for the **Replication** folder, and choose **Configure Distribution**. The Configure Distribution wizard appears.
3. Follow the wizard to enter the default values and create the distribution.

#### To set up CDC

AWS DMS version 3.4.7 and greater can set up MS CDC for your database and all of your tables automatically if you aren't using a read-only replica. To use this feature, set the `SetUpMsCdcForTables` ECA to true. For information about ECAs, see [Endpoint settings \(p. 188\)](#).

For versions of AWS DMS earlier than 3.4.7, or for a read-only replica as a source, perform the following steps:

1. For tables without primary keys, set up MS-CDC for the database. To do so, use an account that has the sysadmin role assigned to it, and run the following command.

```
use [DBname]
```

```
EXEC sys.sp_cdc_enable_db
```

2. Next, set up MS-CDC for each of the source tables. For each table with unique keys but no primary key, run the following query to set up MS-CDC.

```
exec sys.sp_cdc_enable_table
@source_schema = N'schema_name',
@source_name = N'table_name',
@index_name = N'unique_index_name',
@role_name = NULL,
@supports_net_changes = 1
GO
```

3. For each table with no primary key or no unique keys, run the following query to set up MS-CDC.

```
exec sys.sp_cdc_enable_table
@source_schema = N'schema_name',
@source_name = N'table_name',
@role_name = NULL
GO
```

For more information on setting up MS-CDC for specific tables, see the [SQL Server documentation](#).

## Setting up ongoing replication without the sysadmin role on self-managed SQL Server

You can set up ongoing replication for a SQL Server database source that doesn't require the user account to have sysadmin privileges. You still need a user with sysadmin privileges to configure your SQL Server database for ongoing replication.

### Note

You can perform this procedure while the DMS task is running. If the DMS task is stopped, you can perform this procedure only if there are no transaction log or database backups in progress. This is because SQL Server requires the SYSADMIN privilege to query the backups for the log sequence number (LSN) position.

Before setting up ongoing replication, see [Prerequisites for using ongoing replication \(CDC\) from a SQL Server source \(p. 179\)](#).

For tables with primary keys, perform the following procedure.

### To set up a SQL Server database source for ongoing replication without using the sysadmin role

1. Create a new SQL Server account with password authentication using SQL Server Management Studio (SSMS). In this example, we use an account called dmstest.
2. In the **User Mappings** section of SSMS, choose the MSDB and MASTER databases (which gives public permission) and assign the DB\_OWNER role for the database you want to use for ongoing replication.

```
USE [master]
CREATE LOGIN [dmstest] WITH PASSWORD=N'xxxxxxxx';
CREATE USER [dmstest] FOR LOGIN [dmstest];

USE [msdb];
CREATE USER [dmstest] FOR LOGIN [dmstest];

USE [DB name used by DMS];
```

```
CREATE USER [dmstest] FOR LOGIN [dmstest];
ALTER ROLE [db_owner] ADD MEMBER [dmstest];
```

3. Open the context (right-click) menu for the new account, choose **Security** and explicitly grant the Connect SQL privilege.
4. Run the following grant commands.

```
GRANT SELECT ON FN_DBLOG TO dmstest;
GRANT VIEW SERVER STATE TO dmstest;
use msdb;
GRANT EXECUTE ON MSDB.DBO.SP_STOP_JOB TO dmstest;
GRANT EXECUTE ON MSDB.DBO.SP_START_JOB TO dmstest;
GRANT SELECT ON MSDB.DBO.BACKUPSET TO dmstest;
GRANT SELECT ON MSDB.DBO.BACKUPMEDIAFAMILY TO dmstest;
GRANT SELECT ON MSDB.DBO.BACKUPFILE TO dmstest;
```

5. In SSMS, open the context (right-click) menu for the **Replication** folder, and then choose **Configure Distribution**. Follow all default steps and configure this SQL Server instance for distribution. A distribution database is created under databases.
6. Create a publication for SQL Server ongoing replication as follows:
  - a. Log in to SSMS using the SYSADMIN user account.
  - b. Expand **Replication**.
  - c. Open the context (right-click) menu for **Local Publications**.
  - d. In the New Publication wizard, choose **Next**.
  - e. Choose the database where you want to create the publication.
  - f. Choose **Transactional publication**, and then choose **Next**.
  - g. Expand **Tables** and choose the tables with PK and the tables that you want to publish. Choose **Next**.
  - h. Choose **Next**, because you don't need to create a filter.
  - i. In the **Snapshot Agent** screen, choose the first option to **Create a snapshot immediately and keep the snapshot available to initialize subscriptions**. Choose **Next**.
  - j. Choose **Security Settings**, and then choose **Run under the SQL Server Agent service account**. Make sure to choose **By impersonating the process account** for a publisher connection. Choose **OK**.
  - k. Choose **Next**.
  - l. Choose **Create the publication**.
  - m. Provide a name of the publication in the format AR\_PUBLICATION\_000*DBID*.

For example, if your *DBID* is less than 10, name the publication AR\_PUBLICATION\_0000*DBID* (4 zeros). If your *DBID* is greater than or equal to 10, name the publication AR\_PUBLICATION\_000*DBID* (3 zeros). You can also use the DB\_ID function in SQL Server. For more information on the DB\_ID function, see the [SQL Server documentation](#).

7. Create a new AWS DMS task with SQL Server as the source endpoint using the user account that you created.

For tables without primary keys, set up MS-CDC for the database. To do so, use an account that has the sysadmin role assigned to it, and run the following command.

```
use [DBname]
EXEC sys.sp_cdc_enable_db
```

Next, set up MS-CDC for each of the source tables. For each table with unique keys but no primary key, run the following query to set up MS-CDC.

```
exec sys.sp_cdc_enable_table
@source_schema = N'schema_name',
@source_name = N'table_name',
@index_name = N'unique_index_name',
@role_name = NULL,
@supports_net_changes = 1
GO
```

For each table with no primary key or no unique keys, run the following query to set up MS-CDC.

```
exec sys.sp_cdc_enable_table
@source_schema = N'schema_name',
@source_name = N'table_name',
@role_name = NULL
GO
```

For more information on setting up MS-CDC for specific tables, see the [SQL Server documentation](#).

## Setting up ongoing replication on a cloud SQL Server DB instance

This section describes how to set up CDC on a cloud-hosted SQL Server database instance. A cloud-hosted SQL server instance is an instance running on Amazon RDS for SQL Server, an Azure SQL Managed Instance, or an Azure SQL instance.

Before setting up ongoing replication, see [Prerequisites for using ongoing replication \(CDC\) from a SQL Server source \(p. 179\)](#).

Unlike self-managed Microsoft SQL Server sources, Amazon RDS for SQL Server doesn't support MS-Replication. Therefore, AWS DMS needs to use MS-CDC for tables with or without primary keys.

Amazon RDS doesn't grant sysadmin privileges for setting replication artifacts that AWS DMS uses for ongoing changes in a source SQL Server instance. Make sure to turn on MS-CDC for the Amazon RDS instance (using master user privileges) as in the following procedure.

### To turn on MS-CDC for a cloud SQL Server DB instance

1. Run one of the following queries at the database level.

For an RDS for SQL Server DB instance, use this query.

```
exec msdb.dbo.rds_cdc_enable_db 'DB_name'
```

For an Azure SQL managed DB instance, use this query.

```
USE DB_name
GO
EXEC sys.sp_cdc_enable_db
GO
```

2. For each table with a primary key, run the following query to turn on MS-CDC.

```
exec sys.sp_cdc_enable_table
@source_schema = N'schema_name',
@source_name = N'table_name',
@role_name = NULL,
@supports_net_changes = 1
```

```
GO
```

For each table with unique keys but no primary key, run the following query to turn on MS-CDC.

```
exec sys.sp_cdc_enable_table
@source_schema = N'schema_name',
@source_name = N'table_name',
@index_name = N'unique_index_name',
@role_name = NULL,
@supports_net_changes = 1
GO
```

For each table with no primary key nor unique keys, run the following query to turn on MS-CDC.

```
exec sys.sp_cdc_enable_table
@source_schema = N'schema_name',
@source_name = N'table_name',
@role_name = NULL
GO
```

3. Set the retention period for changes to be available on the source using the following commands.

```
use dbname
EXEC sys.sp_cdc_change_job @job_type = 'capture' ,@pollinginterval = 86399
exec sp_cdc_stop_job 'capture'
exec sp_cdc_start_job 'capture'
```

The parameter `@pollinginterval` is measured in seconds with a recommended value set to 86399. This means that the transaction log retains changes for 86,399 seconds (one day) when `@pollinginterval = 86399`. The procedure `exec sp_cdc_start_job 'capture'` initiates the settings.

**Note**

With some versions of SQL Server, if the value of `pollinginterval` is set to more than 3599 seconds, the value resets to the default five seconds. When this happens, T-Log entries are purged before AWS DMS can read them. To determine which SQL Server versions are affected by this known issue, see [this Microsoft KB article](#).

If you are using Amazon RDS with Multi-AZ, make sure that you also set your secondary to have the right values in case of failover.

```
exec rdsadmin..rds_set_configuration 'cdc_capture_pollinginterval' , 86399
```

If an AWS DMS replication task that captures ongoing changes to your SQL Server source stops for more than one hour, use the following procedure.

**To maintain the retention period during an AWS DMS replication task**

1. Stop the job truncating the transaction logs by using the following command.

```
exec sp_cdc_stop_job 'capture'
```

2. Find your task on the AWS DMS console and resume the task.
3. Choose the **Monitoring** tab, and check the `CDCLatencySource` metric.
4. After the `CDCLatencySource` metric equals 0 (zero) and stays there, restart the job truncating the transaction logs using the following command.

```
exec sp_cdc_start_job 'capture'
```

Remember to start the job that truncates SQL Server transaction logs. Otherwise, storage on your SQL Server instance might fill up.

### Limitations for ongoing replication on a cloud SQL Server DB instance

- AWS DMS supports ongoing replication (CDC) with the active transaction log only. You can't use the backup log with CDC.
- You may lose events if you move them from the active transaction log to the backup log, or truncate them from the active transaction log.

## Recommended settings when using Amazon RDS for SQL Server as a source for AWS DMS

When you work with Amazon RDS for SQL Server as a source, the capture job relies on the parameters `maxscans` and `maxtrans`. These parameters govern the maximum number of scans that the capture does on the transaction log and the number of transactions that are processed for each scan.

For databases, where a number of transactions is greater than `maxtrans * maxscans`, increasing the `polling_interval` value can cause an accumulation of active transaction log records. In turn, this accumulation can lead to an increase in the size of the transaction log.

Note that AWS DMS does not rely on the MS-CDC capture job. The MS-CDC capture job marks the transaction log entries as having been processed. This allows the transaction log backup job to remove the entries from the transaction log.

We recommend that you monitor the size of the transaction log and the success of the MS-CDC jobs. If the MS-CDC jobs fail, the transaction log could grow excessively and cause AWS DMS replication failures. You can monitor MS-CDC capture job errors using the `sys.dm_cdc_errors` dynamic management view in the source database. You can monitor the transaction log size using the `DBCC SQLPERF(LOGSPACE)` management command.

### To address the transaction log increase that is caused by MS-CDC

1. Check the Log Space Used % for the database AWS DMS is replicating from and validate that it increases continuously.

```
DBCC SQLPERF(LOGSPACE)
```

2. Identify what is blocking the transaction log backup process.

```
Select log_reuse_wait, log_reuse_wait_desc, name from sys.databases where name = db_name();
```

If the `log_reuse_wait_desc` value equals REPLICATION, the log backup retention is caused by the latency in MS-CDC.

3. Increase the number of events processed by the capture job by increasing the `maxtrans` and `maxscans` parameter values.

```
EXEC sys.sp_cdc_change_job @job_type = 'capture' ,@maxtrans = 5000, @maxscans = 20
exec sp_cdc_stop_job 'capture'
```

```
exec sp_cdc_start_job 'capture'
```

To address this issue, set the values of `maxscans` and `maxtrans` so that `maxtrans*maxscans` is equal to the average number of events generated for tables that AWS DMS replicates from the source database for each day.

If you set these parameters higher than the recommended value, the capture jobs process all events in the transaction logs. If you set these parameters below the recommended value, MS-CDC latency increases and your transaction log grows.

Identifying appropriate values for `maxscans` and `maxtrans` can be difficult because changes in workload produce varying number of events. In this case, we recommend that you set up monitoring on MS-CDC latency. For more information, see [Monitor the process](#) in SQL Server documentation. Then configure `maxtrans` and `maxscans` dynamically based on the monitoring results.

If the AWS DMS task is unable to find the log sequence numbers (LSNs) needed to resume or continue the task, the task may fail and require a complete reload.

## Supported compression methods for SQL Server

The following table shows the compression methods that AWS DMS supports for each SQL Server version.

SQL Server version	Row/Page compression (at partition level)	Vardecimal storage format
2005	No	No
2008	Yes	No
2012	Yes	No
2014	Yes	No

### Note

Sparse columns and columnar structure compression aren't supported.

## Working with self-managed SQL Server AlwaysOn availability groups

SQL Server Always On availability groups provide high availability and disaster recovery as an enterprise-level alternative to database mirroring.

In AWS DMS, you can migrate changes from a single primary or secondary availability group replica.

### Working with the primary availability group replica

#### To use the primary availability group as a source in AWS DMS, do the following:

1. Turn on the distribution option for all SQL Server instances in your availability replicas. For more information, see [Setting up ongoing replication using the sysadmin role with self-managed SQL Server \(p. 180\)](#).
2. In the AWS DMS console, open the SQL Server source database settings. For **Server Name**, specify the Domain Name Service (DNS) name or IP address that was configured for your availability group listener.

When you start an AWS DMS task for the first time, it might take longer than usual to start. This slowness occurs because the creation of the table articles is being duplicated by the availability group server.

## Working with a secondary availability group replica

### To use a secondary availability group as a source in AWS DMS, do the following:

1. Use the same credentials for connecting to individual replicas as those used by the AWS DMS source endpoint user.
2. Ensure that your AWS DMS replication instance can resolve DNS names for all existing replicas, and connect to them. You can use the following SQL query to get DNS names for all of your replicas.

```
select ar.replica_server_name, ar.endpoint_url from sys.availability_replicas ar
JOIN sys.availability_databases_cluster adc
ON adc.group_id = ar.group_id AND adc.database_name = '<source_database_name>';
```

3. When you create the source endpoint, specify the DNS name of the availability group listener for the endpoint's **Server name** or for the endpoint secret's **Server address**. For more information about availability group listeners, see [What is an availability group listener?](#) in the SQL Server documentation.

You can use either a public DNS server or an on-premises DNS server to resolve the availability group listener, the primary replica, and the secondary replicas. To use an on-premises DNS server, configure the Amazon Route 53 Resolver. For more information, see [Using your own on-premises name server \(p. 90\)](#).

4. Add the following extra connection attributes to your source endpoint.

Extra connection attribute	Value	Notes
applicationIntent	ReadOnly	Without this ODBC setting, the replication task is routed to the primary availability group replica. For more information, see <a href="#">SQL Server Native Client Support for High Availability, Disaster Recovery</a> in the SQL Server documentation.
multiSubnetFailover	yes	For more information, see <a href="#">SQL Server Native Client Support for High Availability, Disaster Recovery</a> in the SQL Server documentation.
alwaysOnSharedSyncedBackup.IsEnabled	False	For more information, see <a href="#">Endpoint settings when using SQL Server as a source for AWS DMS (p. 188)</a> .
activateSafeguard	false	For more information, see <a href="#">Limitations (p. 188)</a> following.
setUpMsCdcForTables	false	For more information, see <a href="#">Limitations (p. 188)</a> following.

5. Enable the distribution option on all replicas in your availability group. Add all nodes to the distributors list. For more information, see [To set up distribution \(p. 180\)](#).
6. Run the following query on the primary read-write replica to enable publication of your database. You run this query only once for your database.

```
sp_replicationdboption @dbname = N'<source DB name>', @optname = N'publish', @value = N'true';
```

## Limitations

Following are limitations for working with a secondary availability group replica:

- AWS DMS doesn't support Safeguard when using a read-only availability group replica as a source. For more information, see [Endpoint settings when using SQL Server as a source for AWS DMS \(p. 188\)](#).
- AWS DMS doesn't support the `setUpMsCdcForTables` extra connection attribute when using a read-only availability group replica as a source. For more information, see [Endpoint settings when using SQL Server as a source for AWS DMS \(p. 188\)](#).
- AWS DMS can use a self-managed secondary availability group replica as a source database for ongoing replication (change data capture, or CDC) starting from version 3.4.7. Cloud SQL Server Multi-AZ read replicas are not supported. If you use previous versions of AWS DMS, make sure that you use the primary availability group replica as a source database for CDC.

## Failover to other nodes

If you set the `ApplicationIntent` extra connection attribute for your endpoint to `ReadOnly`, your AWS DMS task connects to the read-only node with the highest read-only routing priority. It then fails over to other read-only nodes in your availability group when the highest priority read-only node is unavailable. If you don't set `ApplicationIntent`, your AWS DMS task only connects to the primary (read/write) node in your availability group.

## Endpoint settings when using SQL Server as a source for AWS DMS

You can use endpoint settings to configure your SQL Server source database similar to using extra connection attributes. You specify the settings when you create the source endpoint using the AWS DMS console, or by using the `create-endpoint` command in the [AWS CLI](#), with the `--sqlserver-settings '{"EndpointSetting": "value", ...}'` JSON syntax.

The following table shows the endpoint settings that you can use with SQL Server as a source.

Name	Description
<code>AlwaysOnSharedSyncedBackupIsEnabled</code>	This attribute adjusts the behavior of AWS DMS when migrating from an SQL Server source database that is hosted as part of an Always On availability group cluster.  AWS DMS has enhanced support for SQL Server source databases that are configured to run in an Always On cluster. In this case, AWS DMS attempts to track if transaction backups are happening from nodes in the Always On cluster other than the node where the source database instance is hosted. At migration task start-up, AWS DMS tries to connect to each node in the cluster, but fails if it can't connect to any one of the nodes.  If you need AWS DMS to poll all the nodes in the Always On cluster for transaction backups, set this attribute to <code>false</code> .

Name	Description
	Default value: <code>true</code>  Valid values: <code>true</code> or <code>false</code>
ActivateSafeguard	This attribute turns Safeguard on or off. For information about Safeguard, see <code>SafeguardPolicy</code> following.  Default value: <code>true</code>  Valid values: <code>{false, true}</code>

Name	Description
SafeguardPolicy	<p>For optimal performance, AWS DMS tries to capture all unread changes from the active transaction log (TLOG). However, sometimes due to truncation, the active TLOG might not contain all of the unread changes. When this occurs, AWS DMS accesses the backup log to capture the missing changes. To minimize the need to access the backup log, AWS DMS prevents truncation using one of the following methods:</p> <p><b>1. Start transactions in the database:</b> This is the default method. When this method is used, AWS DMS prevents TLOG truncation by mimicking a transaction in the database. As long as such a transaction is open, changes that appear after the transaction started aren't truncated. If you need Microsoft Replication to be enabled in your database, then you must choose this method.</p> <p>The purpose of the EXCLUSIVE_AUTOMATIC_TRUNCATION setting is to prevent truncation of unread changes from the transaction log. When you choose this option, AWS DMS creates a table in the source database called <code>aws_truncation_safeguard</code>. This table prevents the truncation of the transaction log by mimicking a transaction in the database. It prevents the events being truncated and moved to the backup log for five minutes (by default). AWS DMS can then read all of the events from the transaction log without needing to read events from the backup log. You can safely delete the <code>aws_truncation_safeguard</code> table if there are no tasks with SafeguardPolicy set to "Start transactions in the database".</p> <p><b>2. Exclusively use <code>sp_repldone</code> within a single task:</b> When this method is used, AWS DMS reads the changes and then uses <code>sp_repldone</code> to mark the TLOG transactions as ready for truncation. Although this method doesn't involve any transactional activities, it can only be used when Microsoft Replication isn't running. Also, when using this method, only one AWS DMS task can access the database at any given time. Therefore, if you need to run parallel AWS DMS tasks against the same database, use the default method.</p> <p><b>Default value:</b>  <code>RELY_ON_SQL_SERVER_REPLICATION_AGENT</code></p> <p><b>Valid values:</b> {EXCLUSIVE_AUTOMATIC_TRUNCATION, RELY_ON_SQL_SERVER_REPLICATION_AGENT}</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"> <li>• EXCLUSIVE_AUTOMATIC_TRUNCATION doesn't work on RDS for SQL Server source instances because RDS users don't have access to run the SQL Server stored procedure, <code>sp_repldone</code>.</li> <li>• If you set SafeguardPolicy to EXCLUSIVE_AUTOMATIC_TRUNCATION without using</li> </ul>

Name	Description
	the sysadmin role, you must grant permissions on the dbo.syscategories and dbo.sysjobs objects to the dmsuser user.
SetUpMsCdcForTables	<p>This attribute turns on MS-CDC for the source database and for tables in the task mapping that don't have MS-Replication enabled. Setting this value to true runs the sp_cdc_enable_db stored procedure on the source database, and runs the sp_cdc_enable_table stored procedure on each table in the task that doesn't have MS-Replication enabled in the source database. For more information about turning on distribution, see <a href="#">Setting up ongoing replication using the sysadmin role with self-managed SQL Server (p. 180)</a>.</p> <p>Valid values: {true, false}</p>
ReadBackupOnly	<p>Use of this attribute requires <b>sysadmin</b> privileges. When this attribute is set to Y, during ongoing replication AWS DMS reads changes only from transaction log backups and doesn't read from the active transaction log file. Setting this parameter to Y enables you to control active transaction log file growth during full load and ongoing replication tasks. However, it can add some source latency to ongoing replication.</p> <p>Valid values: N or Y. The default is N.</p> <p><b>Note:</b> This parameter doesn't work on Amazon RDS SQL Server source instances because of the way RDS performs backups.</p>
Use3rdPartyBackupDevice	When this attribute is set to Y, AWS DMS processes third-party transaction log backups if they are created in native format.
MultiSubnetFailover=Yes	This ODBC driver attribute helps DMS to connect to the new primary in case of an Availability Group failover. This attribute is designed for situations when the connection is broken or the listener IP address is incorrect. In these situations, AWS DMS attempts to connect to all IP addresses associated with the Availability Group listener.
ApplicationIntent=readonly	This ODBC driver attribute setting causes SQL Server to route your replication task to the highest priority read-only node. Without this setting, SQL Server routes your replication task to the primary read-write node.
FatalOnSimpleModel	<p>When set to true, this setting generates a fatal error when SQL Server database recovery model is set to simple.</p> <p>Default value: false</p> <p>Valid values: true or false</p>

Name	Description
TlogAccessMode	Indicates the mode used to fetch CDC data.  Default value: PreferTlog  Valid values: BackupOnly, PreferBackup, PreferTlog, TlogOnly
ForceLobLookup	Forces LOB lookup on inline LOB.  Default value: false  Valid values: true, false

## Source data types for SQL Server

Data migration that uses SQL Server as a source for AWS DMS supports most SQL Server data types. The following table shows the SQL Server source data types that are supported when using AWS DMS and the default mapping from AWS DMS data types.

For information on how to view the data type that is mapped in the target, see the section for the target endpoint you are using.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

SQL Server data types	AWS DMS data types
BIGINT	INT8
BIT	BOOLEAN
DECIMAL	NUMERIC
INT	INT4
MONEY	NUMERIC
NUMERIC (p,s)	NUMERIC
SMALLINT	INT2
SMALLMONEY	NUMERIC
TINYINT	UINT1
REAL	REAL4
FLOAT	REAL8
DATETIME	DATETIME
DATETIME2 (SQL Server 2008 and later)	DATETIME
SMALLDATETIME	DATETIME
DATE	DATE
TIME	TIME

SQL Server data types	AWS DMS data types
DATETIMEOFFSET	WSTRING
CHAR	STRING
VARCHAR	STRING
VARCHAR (max)	<p>CLOB</p> <p>TEXT</p> <p>To use this data type with AWS DMS, you must enable the use of CLOB data types for a specific task.</p> <p>For SQL Server tables, AWS DMS updates LOB columns in the target even for UPDATE statements that don't change the value of the LOB column in SQL Server.</p> <p>During CDC, AWS DMS supports CLOB data types only in tables that include a primary key.</p>
NCHAR	WSTRING
NVARCHAR (length)	WSTRING
NVARCHAR (max)	<p>NCLOB</p> <p>NTEXT</p> <p>To use this data type with AWS DMS, you must enable the use of SupportLobs for a specific task. For more information about enabling Lob support, see <a href="#">Setting LOB support for source databases in an AWS DMS task (p. 479)</a>.</p> <p>For SQL Server tables, AWS DMS updates LOB columns in the target even for UPDATE statements that don't change the value of the LOB column in SQL Server.</p> <p>During CDC, AWS DMS supports CLOB data types only in tables that include a primary key.</p>
BINARY	BYTES
VARBINARY	BYTES

SQL Server data types	AWS DMS data types
VARBINARY (max)	BLOB  IMAGE  For SQL Server tables, AWS DMS updates LOB columns in the target even for UPDATE statements that don't change the value of the LOB column in SQL Server.  To use this data type with AWS DMS, you must enable the use of BLOB data types for a specific task.  AWS DMS supports BLOB data types only in tables that include a primary key.
TIMESTAMP	BYTES
UNIQUEIDENTIFIER	STRING
HIERARCHYID	Use HIERARCHYID when replicating to a SQL Server target endpoint.  Use WSTRING (250) when replicating to all other target endpoints.
XML	NCLOB  For SQL Server tables, AWS DMS updates LOB columns in the target even for UPDATE statements that don't change the value of the LOB column in SQL Server.  To use this data type with AWS DMS, you must enable the use of NCLOB data types for a specific task.  During CDC, AWS DMS supports NCLOB data types only in tables that include a primary key.
GEOMETRY	Use GEOMETRY when replicating to target endpoints that support this data type.  Use CLOB when replicating to target endpoints that don't support this data type.
GEOGRAPHY	Use GEOGRAPHY when replicating to target endpoints that support this data type.  Use CLOB when replicating to target endpoints that don't support this data type.

AWS DMS doesn't support tables that include fields with the following data types.

- CURSOR
- SQL\_VARIANT

- TABLE

**Note**

User-defined data types are supported according to their base type. For example, a user-defined data type based on DATETIME is handled as a DATETIME data type.

## Using Microsoft Azure SQL database as a source for AWS DMS

With AWS DMS, you can use Microsoft Azure SQL Database as a source in much the same way as you do SQL Server. AWS DMS supports, as a source, the same list of database versions that are supported for SQL Server running on-premises or on an Amazon EC2 instance.

For more information, see [Using a Microsoft SQL Server database as a source for AWS DMS \(p. 175\)](#).

**Note**

AWS DMS doesn't support change data capture operations (CDC) with Azure SQL Database.

## Using Microsoft Azure SQL Managed Instance as a source for AWS DMS

With AWS DMS, you can use Microsoft Azure SQL Managed Instance as a source in much the same way as you do SQL Server. AWS DMS supports, as a source, the same list of database versions that are supported for SQL Server running on-premises or on an Amazon EC2 instance.

For more information, see [Using a Microsoft SQL Server database as a source for AWS DMS \(p. 175\)](#).

## Using Google Cloud for MySQL as a source for AWS DMS

With AWS DMS, you can use Google Cloud for MySQL as a source in much the same way as you do MySQL. AWS DMS supports, as a source, GCP for MySQL versions 5.6, 5.7, and 8.0.

For more information, see [Using a MySQL-compatible database as a source for AWS DMS \(p. 216\)](#).

**Note**

Support for GCP MySQL 8.0 as a source is available in AWS DMS version 3.4.6.

AWS DMS doesn't support the SSL mode verify-full for GCP for MySQL instances.

The GCP MySQL security setting Allow only SSL connections isn't supported, because it requires both server and client certificate verification. AWS DMS only supports server certificate verification.

AWS DMS supports the default GCP CloudSQL for MySQL value of CRC32 for the binlog\_checksum database flag.

## Using a PostgreSQL database as an AWS DMS source

You can migrate data from one or many PostgreSQL databases using AWS DMS. With a PostgreSQL database as a source, you can migrate data to either another PostgreSQL database or one of the other supported databases. AWS DMS supports a PostgreSQL version 9.4 and later (for versions 9.x), 10.x, 11.x, 12.x, 13.x and 14.x database as a source for these types of databases:

- On-premises databases
- Databases on an Amazon EC2 instance

- Databases on an Amazon RDS DB instance
- Databases on an DB instance based on Amazon Aurora PostgreSQL-Compatible Edition
- Databases on an DB instance based on Amazon Aurora PostgreSQL-Compatible Serverless Edition

**Note**

DMS supports Amazon Aurora PostgreSQL—Serverless V1 as a source for Full load only. But you can use Amazon Aurora PostgreSQL—Serverless V2 as a source for Full load, Full load + CDC, and CDC only tasks.

AWS DMS version to use
Use any available AWS DMS version. 10.x, 11.x, 12.x
Use AWS DMS version 3.4.3 and above.
Use AWS DMS version 3.4.7 and above.

You can use Secure Socket Layers (SSL) to encrypt connections between your PostgreSQL endpoint and the replication instance. For more information on using SSL with a PostgreSQL endpoint, see [Using SSL with AWS Database Migration Service \(p. 647\)](#).

As an additional security requirement when using PostgreSQL as a source, the user account specified must be a registered user in the PostgreSQL database.

To configure a PostgreSQL database as an AWS DMS source endpoint, do the following:

- Create a PostgreSQL user with appropriate permissions to provide AWS DMS access to your PostgreSQL source database.

**Note**

- If your PostgreSQL source database is self-managed, see [Working with self-managed PostgreSQL databases as a source in AWS DMS \(p. 197\)](#) for more information.
- If your PostgreSQL source database is managed by Amazon RDS, see [Working with AWS-managed PostgreSQL databases as DMS sources \(p. 198\)](#) for more information.
- Create a PostgreSQL source endpoint that conforms with your chosen PostgreSQL database configuration.
- Create a task or set of tasks to migrate your tables.

To create a full-load-only task, no further endpoint configuration is needed.

Before you create a task for change data capture (a CDC-only or full-load and CDC task), see [Enabling CDC using a self-managed PostgreSQL database as a AWS DMS source \(p. 197\)](#) or [Enabling CDC with an AWS-managed PostgreSQL DB instance with AWS DMS \(p. 199\)](#).

**Topics**

- [Working with self-managed PostgreSQL databases as a source in AWS DMS \(p. 197\)](#)
- [Working with AWS-managed PostgreSQL databases as DMS sources \(p. 198\)](#)
- [Enabling change data capture \(CDC\) using logical replication \(p. 201\)](#)

- [Using native CDC start points to set up a CDC load of a PostgreSQL source \(p. 202\)](#)
- [Migrating from PostgreSQL to PostgreSQL using AWS DMS \(p. 204\)](#)
- [Removing AWS DMS artifacts from a PostgreSQL source database \(p. 208\)](#)
- [Additional configuration settings when using a PostgreSQL database as a DMS source \(p. 208\)](#)
- [Extra connection attributes when using PostgreSQL as a DMS source \(p. 209\)](#)
- [Limitations on using a PostgreSQL database as a DMS source \(p. 212\)](#)
- [Source data types for PostgreSQL \(p. 214\)](#)

## Working with self-managed PostgreSQL databases as a source in AWS DMS

With a self-managed PostgreSQL database as a source, you can migrate data to either another PostgreSQL database, or one of the other target databases supported by AWS DMS. The database source can be an on-premises database or a self-managed engine running on an Amazon EC2 instance. You can use a DB instance for both full-load tasks and change data capture (CDC) tasks.

### Prerequisites to using a self-managed PostgreSQL database as a AWS DMS source

Before migrating data from a self-managed PostgreSQL source database, do the following:

- Make sure that you use a PostgreSQL database that is version 9.4.x or later.
- For full-load plus CDC tasks or CDC-only tasks, grant superuser permissions for the user account specified for the PostgreSQL source database. The user account needs superuser permissions to access replication-specific functions in the source. For full-load only tasks, the user account needs SELECT permissions on tables to migrate them.
- Add the IP address of the AWS DMS replication server to the pg\_hba.conf configuration file and enable replication and socket connections. An example follows.

```
# Replication Instance
host all all 12.3.4.56/00 md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
host replication dms 12.3.4.56/00 md5
```

PostgreSQL's pg\_hba.conf configuration file controls client authentication. (HBA stands for host-based authentication.) The file is traditionally stored in the database cluster's data directory.

- If you're configuring a database as a source for logical replication using AWS DMS see [Enabling CDC using a self-managed PostgreSQL database as a AWS DMS source \(p. 197\)](#)

#### Note

Some AWS DMS transactions are idle for some time before the DMS engine uses them again. By using the parameter `idle_in_transaction_session_timeout` in PostgreSQL versions 9.6 and later, you can cause idle transactions to time out and fail. Don't end idle transactions when you use AWS DMS.

### Enabling CDC using a self-managed PostgreSQL database as a AWS DMS source

AWS DMS supports change data capture (CDC) using logical replication. To enable logical replication of a self-managed PostgreSQL source database, set the following parameters and values in the postgresql.conf configuration file:

- Set `wal_level = logical`.
- Set `max_replication_slots` to a value greater than 1.

Set the `max_replication_slots` value according to the number of tasks that you want to run. For example, to run five tasks you set a minimum of five slots. Slots open automatically as soon as a task starts and remain open even when the task is no longer running. Make sure to manually delete open slots.

- Set `max_wal_senders` to a value greater than 1.

The `max_wal_senders` parameter sets the number of concurrent tasks that can run.

- The `wal_sender_timeout` parameter ends replication connections that are inactive longer than the specified number of milliseconds. The default is 60000 milliseconds (60 seconds). Setting the value to 0 (zero) disables the timeout mechanism, and is a valid setting for DMS.

When setting `wal_sender_timeout` to a non-zero value, DMS requires a minimum of 10000 milliseconds (10 seconds), and fails if the value is less than 10000. Keep the value less than 5 minutes to avoid causing a delay during a Multi-AZ failover of a DMS replication instance.

Some parameters are static, and you can only set them at server start. Any changes to their entries in the configuration file (for a self-managed database) or DB parameter group (for an RDS for PostgreSQL database) are ignored until the server is restarted. For more information, see the [PostgreSQL documentation](#).

For more information about enabling CDC, see [Enabling change data capture \(CDC\) using logical replication \(p. 201\)](#).

## Working with AWS-managed PostgreSQL databases as DMS sources

You can use an AWS-managed PostgreSQL DB instance as a source for AWS DMS. You can perform both full-load tasks and change data capture (CDC) tasks using an AWS-managed PostgreSQL source.

### Prerequisites for using an AWS-managed PostgreSQL database as a DMS source

Before migrating data from an AWS-managed PostgreSQL source database, do the following:

- Use the AWS master user account for the PostgreSQL DB instance as the user account for the PostgreSQL source endpoint for AWS DMS. The master user account has the required roles that allow it to set up CDC. If you use an account other than the master user account, the account must have the `rds_superuser` role and the `rds_replication` role. The `rds_replication` role grants permissions to manage logical slots and to stream data using logical slots.

If you don't use the master user account for the DB instance, make sure to create several objects from the master user account for the account that you use. For information about creating these, see [Migrating an Amazon RDS for PostgreSQL database without using the master user account \(p. 199\)](#).

- If your source database is in a virtual private cloud (VPC), choose the VPC security group that provides access to the DB instance where the database resides. This is needed for the DMS replication instance to connect successfully to the source DB instance. When the database and DMS replication instance are in same VPC, add the appropriate security group to its own inbound rules.

#### Note

Some AWS DMS transactions are idle for some time before the DMS engine uses them again. By using the parameter `idle_in_transaction_session_timeout` in PostgreSQL versions 9.6 and later, you can cause idle transactions to time out and fail. Don't end idle transactions when you use AWS DMS.

## Enabling CDC with an AWS-managed PostgreSQL DB instance with AWS DMS

AWS DMS supports CDC on Amazon RDS PostgreSQL databases when the DB instance is configured to use logical replication. The following table summarizes the logical replication compatibility of each AWS-managed PostgreSQL version.

You can't use RDS PostgreSQL read replicas for CDC (ongoing replication).

PostgreSQL version	AWS DMS full load support	AWS DMS CDC support
Aurora PostgreSQL version 2.1 with PostgreSQL 10.5 compatibility (or lower)	Yes	No
Aurora PostgreSQL version 2.2 with PostgreSQL 10.6 compatibility (or higher)	Yes	Yes
RDS for PostgreSQL with PostgreSQL 10.21 compatibility (or higher)	Yes	Yes

### To enable logical replication for an RDS PostgreSQL DB instance

1. Use the AWS master user account for the PostgreSQL DB instance as the user account for the PostgreSQL source endpoint. The master user account has the required roles that allow it to set up CDC.

If you use an account other than the master user account, make sure to create several objects from the master account for the account that you use. For more information, see [Migrating an Amazon RDS for PostgreSQL database without using the master user account \(p. 199\)](#).

2. Set the `rds.logical_replication` parameter in your DB CLUSTER parameter group to 1. This static parameter requires a reboot of the DB instance to take effect. As part of applying this parameter, AWS DMS sets the `wal_level`, `max_wal_senders`, `max_replication_slots`, and `max_connections` parameters. These parameter changes can increase write ahead log (WAL) generation, so only set `rds.logical_replication` when you use logical replication slots.
3. The `wal_sender_timeout` parameter ends replication connections that are inactive longer than the specified number of milliseconds. The default is 60000 milliseconds (60 seconds). Setting the value to 0 (zero) disables the timeout mechanism, and is a valid setting for DMS.

When setting `wal_sender_timeout` to a non-zero value, DMS requires a minimum of 10000 milliseconds (10 seconds), and fails if the value is between 0 and 10000. Keep the value less than 5 minutes to avoid causing a delay during a Multi-AZ failover of a DMS replication instance.

4. Ensure the value of the `max_worker_processes` parameter in your DB Cluster Parameter Group is equal to, or higher than the total combined values of `max_logical_replication_workers`, `autovacuum_max_workers`, and `max_parallel_workers`. A high number of background worker processes might impact application workloads on small instances. So, monitor performance of your database if you set `max_worker_processes` higher than the default value.

### Migrating an Amazon RDS for PostgreSQL database without using the master user account

In some cases, you might not use the master user account for the Amazon RDS PostgreSQL DB instance that you are using as a source. In these cases, you create several objects to capture data definition

language (DDL) events. You create these objects in the account other than the master account and then create a trigger in the master user account.

**Note**

If you set the captureDDLS extra connection attribute to *N* on the source endpoint, you don't have to create the following table and trigger on the source database.

Use the following procedure to create these objects.

**To create objects**

1. Choose the schema where the objects are to be created. The default schema is `public`. Ensure that the schema exists and is accessible by the `NoPriv` account.
2. Log in to the PostgreSQL DB instance using the user account other than the master account, here the `NoPriv` account.
3. Create the table `awsdms_ddl_audit` by running the following command, replacing `objects_schema` in the following code with the name of the schema to use.

```
create table objects_schema.awsdms_ddl_audit
(
    c_key      bigserial primary key,
    c_time     timestamp,      -- Informational
    c_user     varchar(64),   -- Informational: current_user
    c_txn      varchar(16),   -- Informational: current transaction
    c_tag      varchar(24),   -- Either 'CREATE TABLE' or 'ALTER TABLE' or 'DROP TABLE'
    c_oid      integer,       -- For future use - TG_OBJECTID
    c_name     varchar(64),   -- For future use - TG_OBJECTNAME
    c_schema   varchar(64),   -- For future use - TG_SCHEMANAME. For now - holds
    current_schema
    c_ddlqry   text          -- The DDL query associated with the current DDL event
)
```

4. Create the function `awsdms_intercept_ddl` by running the following command, replacing `objects_schema` in the code following with the name of the schema to use.

```
CREATE OR REPLACE FUNCTION objects_schema.awsdms_intercept_ddl()
RETURNS event_trigger
LANGUAGE plpgsql
SECURITY DEFINER
AS $$$
declare _qry text;
BEGIN
    if (tg_tag='CREATE TABLE' or tg_tag='ALTER TABLE' or tg_tag='DROP TABLE') then
        SELECT current_query() into _qry;
        insert into objects_schema.awsdms_ddl_audit
        values
        (
            default,current_timestamp,current_user,cast(TXID_CURRENT()as
varchar(16)),tg_tag,0,'',current_schema,_qry
        );
        delete from objects_schema.awsdms_ddl_audit;
    end if;
END;
$$;
```

5. Log out of the `NoPriv` account and log in with an account that has the `rds_superuser` role assigned to it.

6. Create the event trigger `awsdms_intercept_ddl` by running the following command.

```
CREATE EVENT TRIGGER awsdms_intercept_ddl ON ddl_command_end
EXECUTE PROCEDURE objects_schema.awsdms_intercept_ddl();
```

When you have completed the procedure preceding, you can create the AWS DMS source endpoint using the NoPriv account.

**Note**

These events are triggered by `CREATE TABLE`, `ALTER TABLE`, and `DROP TABLE` statements.

Make sure that all users and roles that access these events have the necessary DDL permissions. For example:

```
grant all on public.awsdms_ddl_audit to public;
grant all on public.awsdms_ddl_audit_c_key_seq to public;
```

## Enabling change data capture (CDC) using logical replication

You can use PostgreSQL's native logical replication feature to enable change data capture (CDC) during database migration for PostgreSQL sources. You can use this feature with a self-managed PostgreSQL and also an Amazon RDS for PostgreSQL SQL DB instance. This approach reduces downtime and help ensure that the target database is in sync with the source PostgreSQL database.

AWS DMS supports CDC for PostgreSQL tables with primary keys. If a table doesn't have a primary key, the write-ahead logs (WAL) don't include a before image of the database row. In this case, DMS can't update the table. Here, you can use additional configuration settings and use table replica identity as a workaround. However, this approach can generate extra logs. We recommend that you use table replica identity as a workaround only after careful testing. For more information, see [Additional configuration settings when using a PostgreSQL database as a DMS source \(p. 208\)](#).

**Note**

`REPLICA IDENTITY FULL` is supported with a logical decoding plugin, but isn't supported with a pglogical plugin. For more information, see [pglogical documentation](#).

For full load and CDC and CDC only tasks, AWS DMS uses logical replication slots to retain WAL logs for replication until the logs are decoded. On restart (not resume) for a full load and CDC task or a CDC task, the replication slot gets recreated.

**Note**

For logical decoding, DMS uses either `test_decoding` or `pglogical` plugin. If the `pglogical` plugin is available on a source PostgreSQL database, DMS creates a replication slot using `pglogical`, otherwise a `test_decoding` plugin is used. For more information about the `test_decoding` plugin, see [PostgreSQL Documentation](#).

## Configuring the pglogical plugin

Implemented as a PostgreSQL extension, the `pglogical` plugin is a logical replication system and model for selective data replication. The following table identifies source PostgreSQL database versions that support the `pglogical` plugin.

PostgreSQL source	Supports pglogical
Self-managed PostgreSQL 9.4 or higher	Yes
Amazon RDS PostgreSQL 9.5 or lower	No

PostgreSQL source	Supports pglogical
Amazon RDS PostgreSQL 9.6 or higher	Yes
Aurora PostgreSQL 1.x till 2.5.x	No
Aurora PostgreSQL 2.6.x or higher	Yes
Aurora PostgreSQL 3.3.x or higher	Yes

Before configuring pglogical for use with AWS DMS, first enable logical replication for change data capture (CDC) on your PostgreSQL source database.

- For information about enabling logical replication for CDC on *self-managed* PostgreSQL source databases, see [Enabling CDC using a self-managed PostgreSQL database as a AWS DMS source \(p. 197\)](#)
- For information about enabling logical replication for CDC on *AWS-managed* PostgreSQL source databases, see [Enabling CDC with an AWS-managed PostgreSQL DB instance with AWS DMS \(p. 199\)](#).

After logical replication is enabled on your PostgreSQL source database, use the following steps to configure pglogical for use with DMS.

### To use the pglogical plugin for logical replication on a PostgreSQL source database with AWS DMS

1. Create a pglogical extension on your source PostgreSQL database:
  - a. Set the correct parameter:
    - For self-managed PostgreSQL databases, set the database parameter `shared_preload_libraries= 'pglogical'`.
    - For PostgreSQL on Amazon RDS and Amazon Aurora PostgreSQL-Compatible Edition databases, set the parameter `shared_preload_libraries` to `pglogical` in the same RDS parameter group.
  - b. Restart your PostgreSQL source database.
  - c. On the PostgreSQL database, run the command, `create extension pglogical;`
2. Run the following command to verify that pglogical installed successfully:

```
select * FROM pg_catalog.pg_extension
```

You can now create a AWS DMS task that performs change data capture for your PostgreSQL source database endpoint.

#### Note

If you don't enable pglogical on your PostgreSQL source database, AWS DMS uses the `test_decoding` plugin by default. When pglogical is enabled for logical decoding, AWS DMS uses pglogical by default. But you can set the extra connection attribute, `PluginName` to use the `test_decoding` plugin instead.

## Using native CDC start points to set up a CDC load of a PostgreSQL source

To enable native CDC start points with PostgreSQL as a source, set the `slotName` extra connection attribute to the name of an existing logical replication slot when you create the endpoint. This logical

replication slot holds ongoing changes from the time of endpoint creation, so it supports replication from a previous point in time.

PostgreSQL writes the database changes to WAL files that are discarded only after AWS DMS successfully reads changes from the logical replication slot. Using logical replication slots can protect logged changes from being deleted before they are consumed by the replication engine.

However, depending on rate of change and consumption, changes being held in a logical replication slot can cause elevated disk usage. We recommend that you set space usage alarms in the source PostgreSQL instance when you use logical replication slots. For more information on setting the slotName extra connection attribute, see [Extra connection attributes when using PostgreSQL as a DMS source \(p. 209\)](#).

The following procedure walks through this approach in more detail.

### To use a native CDC start point to set up a CDC load of a PostgreSQL source endpoint

1. Identify the logical replication slot used by an earlier replication task (a parent task) that you want to use as a start point. Then query the pg\_replication\_slots view on your source database to make sure that this slot doesn't have any active connections. If it does, resolve and close them before proceeding.

For the following steps, assume that your logical replication slot is abc1d2efghijk\_34567890\_z0yx98w7\_6v54\_32ut\_1srq\_1a2b34c5d67ef.

2. Create a new source endpoint that includes the following extra connection attribute setting.

```
slotName=abc1d2efghijk_34567890_z0yx98w7_6v54_32ut_1srq_1a2b34c5d67ef;
```

3. Create a new CDC-only task using the AWS CLI or AWS DMS API. For example, using the CLI you might run the following create-replication-task command.

AWS DMS doesn't currently support creating a CDC task with a native start point using the console.

```
aws dms create-replication-task --replication-task-identifier postgresql-slot-name-test
--source-endpoint-arn arn:aws:dms:us-west-2:012345678901:endpoint:ABCD1EFGHIJK2LMNOPQRST3UV4
--target-endpoint-arn arn:aws:dms:us-west-2:012345678901:endpoint:ZYX9WVUTSRQONM8LKJIHGF7ED6
--replication-instance-arn arn:aws:dms:us-west-2:012345678901:rep:AAAAAAAAA5BB4CCC3DDDD2EE
--migration-type cdc --table-mappings "file://mappings.json" --cdc-start-position "4AF/B00000D0"
--replication-task-settings "file://task-pg.json"
```

In the preceding command, the following options are set:

- The source-endpoint-arn option is set to the new value that you created in step 2.
- The replication-instance-arn option is set to the same value as for the parent task from step 1.
- The table-mappings and replication-task-settings options are set to the same values as for the parent task from step 1.
- The cdc-start-position option is set to a start position value. To find this start position, either query the pg\_replication\_slots view on your source database or view the console details for the parent task in step 1. For more information, see [Determining a CDC native start point \(p. 482\)](#).

When this CDC task runs, AWS DMS raises an error if the specified logical replication slot doesn't exist. It also raises an error if the task isn't created with a valid setting for cdc-start-position.

When using native CDC start points with the pglogical plugin and you want to use a new replication slot, complete the setup steps following before creating a CDC task.

### To use a new replication slot not previously created as part of another DMS task

1. Create a replication slot, as shown following:

```
SELECT * FROM pg_create_logical_replication_slot('replication_slot_name', 'pglogical');
```

2. After the database creates the replication slot, get and note the **restart\_lsn** and **confirmed\_flush\_lsn** values for the slot:

```
select * from pg_replication_slots where slot_name like 'replication_slot_name';
```

Note that the Native CDC Start position for a CDC task created after the replication slot can't be older than the **confirmed\_flush\_lsn** value.

For information about the **restart\_lsn** and **confirmed\_flush\_lsn** values, see [pg\\_replication\\_slots](#)

3. Create a pglogical node.

```
SELECT pglogical.create_node(node_name := 'node_name', dsn := 'your_dsn_name');
```

4. Create two replication sets using the pglogical.create\_replication\_set function. The first replication set tracks updates and deletes for tables that have primary keys. The second replication set tracks only inserts, and has the same name as the first replication set, with the added prefix 'i'.

```
SELECT pglogical.create_replication_set('replication_slot_name', false, true, true, false);
SELECT pglogical.create_replication_set('ireplication_slot_name', true, false, false, true);
```

5. Add a table to the replication set.

```
SELECT pglogical.replication_set_add_table('replication_slot_name',
'schemaname.tablename', true);
SELECT pglogical.replication_set_add_table('ireplication_slot_name',
'schemaname.tablename', true);
```

6. Set the extra connection attribute (ECA) following when you create your source endpoint.

```
PluginName=PGLOGICAL;slotName=slot_name;
```

You can now create a CDC only task with a PostgreSQL native start point using the new replication slot. For more information about the pglogical plugin, see the [pglogical 3.7 documentation](#)

## Migrating from PostgreSQL to PostgreSQL using AWS DMS

When you migrate from a database engine other than PostgreSQL to a PostgreSQL database, AWS DMS is almost always the best migration tool to use. But when you are migrating from a PostgreSQL database to a PostgreSQL database, PostgreSQL tools can be more effective.

## Using PostgreSQL native tools to migrate data

We recommend that you use PostgreSQL database migration tools such as pg\_dump under the following conditions:

- You have a homogeneous migration, where you are migrating from a source PostgreSQL database to a target PostgreSQL database.
- You are migrating an entire database.
- The native tools allow you to migrate your data with minimal downtime.

The pg\_dump utility uses the COPY command to create a schema and data dump of a PostgreSQL database. The dump script generated by pg\_dump loads data into a database with the same name and recreates the tables, indexes, and foreign keys. To restore the data to a database with a different name, use the pg\_restore command and the -d parameter.

If you are migrating data from a PostgreSQL source database running on EC2 to an Amazon RDS for PostgreSQL target, you can use the pglogical plugin.

For more information about importing a PostgreSQL database into Amazon RDS for PostgreSQL or Amazon Aurora PostgreSQL-Compatible Edition, see <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/PostgreSQL.Procedural.Importing.html>.

## Using DMS to migrate data from PostgreSQL to PostgreSQL

AWS DMS can migrate data, for example, from a source PostgreSQL database that is on premises to a target Amazon RDS for PostgreSQL or Aurora PostgreSQL instance. Core or basic PostgreSQL data types most often migrate successfully.

### Note

When replicating partitioned tables from a PostgreSQL source to PostgreSQL target, you don't need to mention the parent table as part of the selection criteria in the DMS task. Mentioning the parent table causes data to be duplicated in child tables on the target, possibly causing a PK violation. By selecting child tables alone in the table mapping selection criteria, the parent table is automatically populated.

Data types that are supported on the source database but aren't supported on the target might not migrate successfully. AWS DMS streams some data types as strings if the data type is unknown. Some data types, such as XML and JSON, can successfully migrate as small files but can fail if they are large documents.

When performing data type migration, be aware of the following:

- In some cases, the PostgreSQL NUMERIC(p,s) data type doesn't specify any precision and scale. For DMS versions 3.4.2 and earlier, DMS uses a precision of 28 and a scale of 6 by default, NUMERIC(28,6). For example, the value 0.61111104488373 from the source is converted to 0.611111 on the PostgreSQL target.
- A table with an ARRAY data type must have a primary key. A table with an ARRAY data type missing a primary key gets suspended during full load.

The following table shows source PostgreSQL data types and whether they can be migrated successfully.

Data type	Migrates successfully	Partially migrates	Doesn't migrate	Comments
INTEGER	X			
SMALLINT	X			

Data type	Migrates successfully	Partially migrates	Doesn't migrate	Comments
BIGINT	X			
NUMERIC/DECIMAL(p,s)		X		Where $0 < p < 39$ and $0 < s$
NUMERIC/DECIMAL		X		Where $p > 38$ or $p=s=0$
REAL	X			
DOUBLE	X			
SMALLSERIAL	X			
SERIAL	X			
BIGSERIAL	X			
MONEY	X			
CHAR		X		Without specified precision
CHAR(n)	X			
VARCHAR		X		Without specified precision
VARCHAR(n)	X			
TEXT	X			
BYTEA	X			
TIMESTAMP	X			Positive and negative infinity values are truncated to '9999-12-31 23:59:59' and '4713-01-01 00:00:00 BC' respectively.
TIMESTAMP(Z)		X		
DATE	X			
TIME	X			
TIME (z)		X		
INTERVAL		X		
BOOLEAN	X			

Data type	Migrates successfully	Partially migrates	Doesn't migrate	Comments
ENUM			X	
CIDR	X			
INET			X	
MACADDR			X	
TSVECTOR			X	
TSQUERY			X	
XML		X		
POINT	X			PostGIS spatial data type
LINE			X	
LSEG			X	
BOX			X	
PATH			X	
POLYGON	X			PostGIS spatial data type
CIRCLE			X	
JSON		X		
ARRAY	X			Requires Primary Key
COMPOSITE			X	
RANGE			X	
LINESTRING	X			PostGIS spatial data type
MULTIPOINT	X			PostGIS spatial data type
MULTILINESTRING	X			PostGIS spatial data type
MULTIPOLYGON	X			PostGIS spatial data type
GEOMETRYCOLLECTION	X			PostGIS spatial data type

## Migrating PostGIS spatial data types

*Spatial data* identifies the geometry information of an object or location in space. PostgreSQL object-relational databases support PostGIS spatial data types.

Before migrating PostgreSQL spatial data objects, ensure that the PostGIS plugin is enabled at the global level. Doing this ensures that AWS DMS creates the exact source spatial data columns for the PostgreSQL target DB instance.

For PostgreSQL to PostgreSQL homogeneous migrations, AWS DMS supports the migration of PostGIS geometric and geographic (geodetic coordinates) data object types and subtypes such as the following:

- POINT
- LINESTRING
- POLYGON
- MULTIPOINT
- MULTILINESTRING
- MULTIPOLYGON
- GEOMETRYCOLLECTION

## Removing AWS DMS artifacts from a PostgreSQL source database

To capture DDL events, AWS DMS creates various artifacts in the PostgreSQL database when a migration task starts. When the task completes, you might want to remove these artifacts.

To remove the artifacts, issue the following statements (in the order they appear), where `{AmazonRDSMigration}` is the schema in which the artifacts were created. Dropping a schema should be done with extreme caution. Never drop an operational schema, especially not a public one.

```
drop event trigger awsdms_intercept_ddl;
```

The event trigger doesn't belong to a specific schema.

```
drop function {AmazonRDSMigration}.awsdms_intercept_ddl()
drop table {AmazonRDSMigration}.awsdms_ddl_audit
drop schema {AmazonRDSMigration}
```

## Additional configuration settings when using a PostgreSQL database as a DMS source

You can add additional configuration settings when migrating data from a PostgreSQL database in two ways:

- You can add values to the extra connection attribute to capture DDL events and to specify the schema in which the operational DDL database artifacts are created. For more information, see [Extra connection attributes when using PostgreSQL as a DMS source \(p. 209\)](#).
- You can override connection string parameters. Choose this option to do either of the following:
  - Specify internal AWS DMS parameters. Such parameters are rarely required so aren't exposed in the user interface.
  - Specify pass-through (passthru) values for the specific database client. AWS DMS includes pass-through parameters in the connection string passed to the database client.
- By using the table-level parameter `REPLICATE IDENTITY` in PostgreSQL versions 9.4 and higher, you can control information written to write-ahead logs (WALs). In particular, it does so for WALs that identify rows that are updated or deleted. `REPLICATE IDENTITY FULL` records the old values of all columns in the row. Use `REPLICATE IDENTITY FULL` carefully for each table as `FULL` generates an

extra number of WALs that might not be necessary. For more information, see [ALTER TABLE-REPLICA IDENTITY](#)

## Extra connection attributes when using PostgreSQL as a DMS source

You can use extra connection attributes to configure your PostgreSQL source. You specify these settings when you create the source endpoint. If you have multiple connection attribute settings, separate them from each other by semicolons with no additional white space (for example, oneSetting;thenAnother).

The following table shows the extra connection attributes that you can use when using PostgreSQL as a source for AWS DMS.

Attribute name	Description
captureDDLs	<p>To capture DDL events, AWS DMS creates various artifacts in the PostgreSQL database when the task starts. You can later remove these artifacts as described in <a href="#">Removing AWS DMS artifacts from a PostgreSQL source database (p. 208)</a>.</p> <p>If this value is set to N, you don't have to create tables or triggers on the source database.</p> <p>Streamed DDL events are captured.</p> <p>Default value: Y</p> <p>Valid values: Y/N</p> <p>Example: captureDDLs=Y;</p>
ddlArtifactsSchema	<p>Sets the schema in which the operational DDL database artifacts are created.</p> <p>Default value: public</p> <p>Valid values: String</p> <p>Example: ddlArtifactsSchema=xyzddlschema;</p>
failTasksOnLobTruncation	<p>When set to true, this value causes a task to fail if the actual size of a LOB column is greater than the specified LobMaxSize.</p> <p>If task is set to Limited LOB mode and this option is set to true, the task fails instead of truncating the LOB data.</p> <p>Default value: false</p> <p>Valid values: Boolean</p> <p>Example: failTasksOnLobTruncation=true;</p>
executeTimeout	<p>Sets the client statement timeout for the PostgreSQL instance, in seconds. The default value is 60 seconds.</p> <p>Example: executeTimeout=100;</p>

Attribute name	Description
pluginName	<p>Specifies the plugin to use to create a replication slot.</p> <p>Valid values: pglogical, test_decoding</p> <p>Example: <code>pluginName=test_decoding;</code></p>
slotName	<p>Sets the name of a previously created logical replication slot for a CDC load of the PostgreSQL source instance.</p> <p>When used with the AWS DMS API <code>CdcStartPosition</code> request parameter, this attribute also enables using native CDC start points. DMS verifies that the specified logical replication slot exists before starting the CDC load task. It also verifies that the task was created with a valid setting of <code>CdcStartPosition</code>. If the specified slot doesn't exist or the task doesn't have a valid <code>CdcStartPosition</code> setting, DMS raises an error.</p> <p>For more information about setting the <code>CdcStartPosition</code> request parameter, see <a href="#">Determining a CDC native start point (p. 482)</a>. For more information about using <code>CdcStartPosition</code>, see the documentation for the <code>CreateReplicationTask</code>, <code>StartReplicationTask</code>, and <code>ModifyReplicationTask</code> API operations in the <a href="#">AWS Database Migration Service API Reference</a>.</p> <p>Valid values: String</p> <p>Example: <code>slotName=abc1d2efghijk_34567890_z0yx98w7_6v54_32ut_1srq_1</code></p>
heartbeatEnable	<p>The WAL heartbeat feature mimics a dummy transaction, so that idle logical replication slots don't hold onto old WAL logs that result in storage full situations on the source. This heartbeat keeps <code>restart_lsn</code> moving and prevents storage full scenarios.</p> <p>Default value: <code>false</code></p> <p>Valid values: <code>true/false</code></p> <p>Example: <code>heartbeatEnable=true;</code></p>
heartbeatFrequency	<p>Sets the WAL heartbeat frequency (in minutes).</p> <p>Default value: 5</p> <p>Valid values: Number</p> <p>Example: <code>heartbeatFrequency=1;</code></p>

Attribute name	Description
heartbeatSchema	<p>Sets the schema in which the heartbeat artifacts are created.</p> <p>Default value: public</p> <p>Valid values: String</p> <p>Example: heartbeatSchema=xyzheartbeatschema</p>
consumeMonotonicEvents	<p>Used to control how monolithic transactions with duplicate Log Sequence Numbers (LSNs) are replicated. When this parameter is <code>false</code>, events with duplicate LSNs are consumed and replicated on the target. When this parameter is <code>true</code>, only the first event is replicated while events with duplicate LSNs aren't consumed nor replicated on the target.</p> <p>Default value: <code>false</code></p> <p>Valid values: <code>false/true</code></p> <p>Example: consumeMonotonicEvents=<code>true</code>;</p>
mapUnboundedNumericAsString	<p>This parameter treats columns with unbounded NUMERIC data types as STRING in order to successfully migrate without losing precision of the numeric value. Use this parameter only for replication from PostgreSQL source to PostgreSQL target, or databases with PostgreSQL compatibility.</p> <p>Default value: <code>false</code></p> <p>Valid values: <code>false/true</code></p> <p>Example: mapUnboundedNumericAsString=<code>true</code>;</p> <p>Using this parameter might result in some replication performance degradation because of transformation from numeric to string and back to numeric. This parameter is supported for use by DMS version 3.4.4 and higher</p> <p><b>Note</b>  Only use <code>mapUnboundedNumericAsString</code> in PostgreSQL source and target endpoints together. Use of <code>mapUnboundedNumericAsString</code> on source PostgreSQL endpoints restricts precision to 28 during CDC. Use of <code>mapUnboundedNumericAsString</code> on target endpoints, migrates data with Precision 28 Scale 6. Do not use <code>mapUnboundedNumericAsString</code> with non-PostgreSQL targets.</p>

## Using the MapBooleanAsBoolean PostgreSQL endpoint setting

You can use endpoint settings to configure your PostgreSQL source similar to using extra connection attributes. You can specify these settings when you create the source endpoint using the `create-endpoint` command in the AWS CLI, with the `--postgre-sql-settings json-settings` option. Here,

json-settings is a JSON object containing parameters to specify the settings. You can also specify a .json file containing the same json-settings object. For example, as in the following:

```
--postgre-sql-settings file:///your-file-path/my_postgresql_settings.json
```

Here, my\_postgre-sql\_settings.json is the name of a .json file that contains the same json-settings object.

You can use PostgreSQL endpoint settings to map a boolean as a boolean from your PostgreSQL source to a Amazon Redshift target. By default, a BOOLEAN type is migrated as varchar(5). You can specify MapBooleanAsBoolean to let PostgreSQL to migrate the boolean type as boolean as shown in the example following.

```
--postgre-sql-settings='{"MapBooleanAsBoolean": "true"}'
```

## Limitations on using a PostgreSQL database as a DMS source

The following limitations apply when using PostgreSQL as a source for AWS DMS:

- AWS DMS doesn't work with Amazon RDS for PostgreSQL 10.4 or Amazon Aurora PostgreSQL 10.4 either as source or target.
- A captured table must have a primary key. If a table doesn't have a primary key, AWS DMS ignores DELETE and UPDATE record operations for that table. As a workaround, see [Enabling change data capture \(CDC\) using logical replication \(p. 201\)](#).

**Note:** We don't recommend migrating without a Primary Key/Unique Index, otherwise additional limitations apply such as "NO" Batch apply capability, Full LOB capability, Data Validation and inability to replicate to Redshift target efficiently.

- AWS DMS ignores an attempt to update a primary key segment. In these cases, the target identifies the update as one that didn't update any rows. However, because the results of updating a primary key in PostgreSQL are unpredictable, no records are written to the exceptions table.
- AWS DMS doesn't support the **Start Process Changes from Timestamp** run option.
- AWS DMS doesn't replicate changes that result from partition or subpartition operations (ADD, DROP, or TRUNCATE).
- Replication of multiple tables with the same name where each name has a different case (for example, table1, TABLE1, and Table1) can cause unpredictable behavior. Because of this issue, AWS DMS doesn't support this type of replication.
- In most cases, AWS DMS supports change processing of CREATE, ALTER, and DROP DDL statements for tables. AWS DMS doesn't support this change processing if the tables are held in an inner function or procedure body block or in other nested constructs.

For example, the following change isn't captured.

```
CREATE OR REPLACE FUNCTION attu.create_distributors1() RETURNS void
LANGUAGE plpgsql
AS $$$
BEGIN
create table attu.distributors1(did serial PRIMARY KEY,name
varchar(40) NOT NULL);
END;
$$;
```

- Currently, boolean data types in a PostgreSQL source are migrated to a SQL Server target as bit data type with inconsistent values. As a workaround, pre-create the table with a VARCHAR(1) data type for the column (or have AWS DMS create the table). Then have downstream processing treat an "F" as False and a "T" as True.

- AWS DMS doesn't support change processing of TRUNCATE operations.
- The OID LOB data type isn't migrated to the target.
- AWS DMS supports the PostGIS data type for only homogeneous migrations.
- If your source is a PostgreSQL database that is on-premises or on an Amazon EC2 instance, ensure that the test\_decoding output plugin is installed on your source endpoint. You can find this plugin in the PostgreSQL contrib package. For more information about the test-decoding plugin, see the [PostgreSQL documentation](#).
- AWS DMS doesn't support change processing to set and unset column default values (using the ALTER COLUMN SET DEFAULT clause on ALTER TABLE statements).
- AWS DMS doesn't support change processing to set column nullability (using the ALTER COLUMN [SET|DROP] NOT NULL clause on ALTER TABLE statements).
- When logical replication is enabled, the maximum number of changes kept in memory per transaction is 4 MB. After that, changes are spilled to disk. As a result ReplicationSlotDiskUsage increases, and restart\_lsn doesn't advance until the transaction is completed or stopped and the rollback finishes. Because it is a long transaction, it can take a long time to rollback. So, avoid long running transactions when logical replication is enabled. Instead, break the transaction into several smaller transactions.
- A table with an ARRAY data type must have a primary key. A table with an ARRAY data type missing a primary key gets suspended during full load.
- AWS DMS doesn't support replication of partitioned tables. When a partitioned table is detected, the following occurs:
  - The endpoint reports a list of parent and child tables.
  - AWS DMS creates the table on the target as a regular table with the same properties as the selected tables.
  - If the parent table in the source database has the same primary key value as its child tables, a "duplicate key" error is generated.
- To replicate partitioned tables from a PostgreSQL source to a PostgreSQL target, first manually create the parent and child tables on the target. Then define a separate task to replicate to those tables. In such a case, set the task configuration to **Truncate before loading**.
- The PostgreSQL NUMERIC data type isn't fixed in size. When transferring data that is a NUMERIC data type but without precision and scale, DMS uses NUMERIC(28, 6) (a precision of 28 and scale of 6) by default. As an example, the value 0.61111104488373 from the source is converted to 0.611111 on the PostgreSQL target.
- AWS DMS supports Aurora PostgreSQL Serverless V1 as a source for full load tasks only. AWS DMS supports Aurora PostgreSQL Serverless V2 as a source for full load, full load and CDC, and CDC-only tasks.
- AWS DMS doesn't support replication of a table with a unique index created with a coalesce function.
- When using LOB mode, both the source table and the corresponding target table must have an identical Primary Key. If one of the tables does not have a Primary Key, the result of DELETE and UPDATE record operations will be unpredictable.
- When using the Parallel Load feature, table segmentation according to partitions or sub-partitions isn't supported. For more information about Parallel Load, see [Using parallel load for selected tables, views, and collections \(p. 530\)](#)
- AWS DMS doesn't support Deferred Constraints.
- AWS DMS version 3.4.7 supports PostgreSQL 14.x as a source with these limitations:
  - AWS DMS doesn't support change processing of two phase commits.
  - AWS DMS doesn't support logical replication to stream long in-progress transactions.
- AWS DMS doesn't support CDC for Amazon RDS Proxy for PostgreSQL as a source.
- When using [source filters \(p. 543\)](#) that don't contain a Primary Key column, DELETE operations won't be captured.

- If your source database is also a target for another third-party replication system, DDL changes might not migrate during CDC. Because that situation can prevent the awsdms\_intercept\_ddl event trigger from firing. To work around the situation, modify that trigger on your source database as follows:

```
alter event trigger awsdms_intercept_ddl enable always;
```

## Source data types for PostgreSQL

The following table shows the PostgreSQL source data types that are supported when using AWS DMS and the default mapping to AWS DMS data types.

For information on how to view the data type that is mapped in the target, see the section for the target endpoint you are using.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

PostgreSQL data types	DMS data types
INTEGER	INT4
SMALLINT	INT2
BIGINT	INT8
NUMERIC (p,s)	If precision is from 0 through 38, then use NUMERIC. If precision is 39 or greater, then use STRING.
DECIMAL(P,S)	If precision is from 0 through 38, then use NUMERIC. If precision is 39 or greater, then use STRING.
REAL	REAL4
DOUBLE	REAL8
SMALLSERIAL	INT2
SERIAL	INT4
BIGSERIAL	INT8
MONEY	NUMERIC(38,4) The MONEY data type is mapped to FLOAT in SQL Server.
CHAR	WSTRING (1)
CHAR(N)	WSTRING (n)
VARCHAR(N)	WSTRING (n)
TEXT	NCLOB

PostgreSQL data types	DMS data types
BYTEA	BLOB
TIMESTAMP	DATETIME
TIMESTAMP (z)	DATETIME
TIMESTAMP WITH TIME ZONE	DATETIME
DATE	DATE
TIME	TIME
TIME (z)	TIME
INTERVAL	STRING (128)—1 YEAR, 2 MONTHS, 3 DAYS, 4 HOURS, 5 MINUTES, 6 SECONDS
BOOLEAN	CHAR (5) false or true
ENUM	STRING (64)
CIDR	STRING (50)
INET	STRING (50)
MACADDR	STRING (18)
BIT (n)	STRING (n)
BIT VARYING (n)	STRING (n)
UUID	STRING
TSVECTOR	CLOB
TSQUERY	CLOB
XML	CLOB
POINT	STRING (255) "(x,y)"
LINE	STRING (255) "(x,y,z)"
LSEG	STRING (255) "((x1,y1),(x2,y2))"
BOX	STRING (255) "((x1,y1),(x2,y2))"
PATH	CLOB "((x1,y1),(xn,yn))"
POLYGON	CLOB "((x1,y1),(xn,yn))"
CIRCLE	STRING (255) "(x,y),r"
JSON	NCLOB
JSONB	NCLOB
ARRAY	NCLOB
COMPOSITE	NCLOB

PostgreSQL data types	DMS data types
HSTORE	NCLOB
INT4RANGE	STRING (255)
INT8RANGE	STRING (255)
NUMRANGE	STRING (255)
STRRANGE	STRING (255)

## Working with LOB source data types for PostgreSQL

PostgreSQL column sizes affect the conversion of PostgreSQL LOB data types to AWS DMS data types. To work with this, take the following steps for the following AWS DMS data types:

- BLOB – Set **Limit LOB size to** the **Maximum LOB size (KB)** value at task creation.
- CLOB – Replication handles each character as a UTF8 character. Therefore, find the length of the longest character text in the column, shown here as `max_num_chars_text`. Use this length to specify the value for **Limit LOB size to**. If the data includes 4-byte characters, multiply by 2 to specify the **Limit LOB size to** value, which is in bytes. In this case, **Limit LOB size to** is equal to `max_num_chars_text` multiplied by 2.
- NCLOB – Replication handles each character as a double-byte character. Therefore, find the length of the longest character text in the column (`max_num_chars_text`) and multiply by 2. You do this to specify the value for **Limit LOB size to**. In this case, **Limit LOB size to** is equal to `max_num_chars_text` multiplied by 2. If the data includes 4-byte characters, multiply by 2 again. In this case, **Limit LOB size to** is equal to `max_num_chars_text` multiplied by 4.

## Using a MySQL-compatible database as a source for AWS DMS

You can migrate data from any MySQL-compatible database (MySQL, MariaDB, or Amazon Aurora MySQL) using AWS Database Migration Service. MySQL versions 5.5, 5.6, 5.7, and 8.0. MariaDB versions 10.0.24 to 10.0.28, 10.1, 10.2, 10.3, 10.4, and 10.5, and also Amazon Aurora MySQL, are supported for on-premises.

### Note

Support for MySQL 8.0 as a source is available in AWS DMS versions 3.4.0 and later, except when the transaction payload is compressed. AWS DMS doesn't currently support CDC replication using MySQL 8.0 as a source when binary log encryption is turned on.

You can use SSL to encrypt connections between your MySQL-compatible endpoint and the replication instance. For more information on using SSL with a MySQL-compatible endpoint, see [Using SSL with AWS Database Migration Service \(p. 647\)](#).

In the following sections, the term "self-managed" applies to any database that is installed either on-premises or on Amazon EC2. The term "AWS-managed" applies to any database on Amazon RDS, Amazon Aurora, or Amazon S3.

For additional details on working with MySQL-compatible databases and AWS DMS, see the following sections.

### Topics

- [Migrating from MySQL to MySQL using AWS DMS \(p. 217\)](#)

- [Using any MySQL-compatible database as a source for AWS DMS \(p. 219\)](#)
- [Using a self-managed MySQL-compatible database as a source for AWS DMS \(p. 219\)](#)
- [Using an AWS-managed MySQL-compatible database as a source for AWS DMS \(p. 220\)](#)
- [Limitations on using a MySQL database as a source for AWS DMS \(p. 221\)](#)
- [Extra connection attributes when using MySQL as a source for AWS DMS \(p. 222\)](#)
- [Source data types for MySQL \(p. 223\)](#)

## Migrating from MySQL to MySQL using AWS DMS

For a heterogeneous migration, where you are migrating from a database engine other than MySQL to a MySQL database, AWS DMS is almost always the best migration tool to use. But for a homogeneous migration, where you are migrating from a MySQL database to a MySQL database, native tools can be more effective.

We recommend that you use native MySQL database migration tools such as `mysqldump` under the following conditions:

- You have a homogeneous migration, where you are migrating from a source MySQL database to a target MySQL database.
- You are migrating an entire database.
- The native tools allow you to migrate your data with minimal downtime.

You can import data from an existing MySQL or MariaDB database to an Amazon RDS MySQL or MariaDB DB instance. You do so by copying the database with `mysqldump` and piping it directly into the Amazon RDS MySQL or MariaDB DB instance. The `mysqldump` command-line utility is commonly used to make backups and transfer data from one MySQL or MariaDB server to another. It is included with MySQL and MariaDB client software.

For more information about importing a MySQL database into Amazon RDS for MySQL or Amazon Aurora MySQL-Compatible Edition, see [Importing data into a MySQL DB instance](#) and [Importing data from a MySQL or MariaDB DB to an Amazon RDS MySQL or MariaDB DB instance](#).

### Using AWS DMS to migrate data from MySQL to MySQL

AWS DMS can migrate data from, for example, a source MySQL database that is on premises to a target Amazon RDS for MySQL or Aurora MySQL instance. Core or basic MySQL data types most often migrate successfully.

Data types that are supported on the source database but aren't supported on the target might not migrate successfully. AWS DMS streams some data types as strings if the data type is unknown. Some data types, such as XML, can successfully migrate as small files but can fail if they are large documents.

The following table shows source MySQL data types and whether they can be migrated successfully.

Data type	Migrates successfully	Will partially migrate	Will not migrate
INT	X		
BIGINT	X		
MEDIUMINT	X		
TINYINT	X		
DECIMAL(p,s)	X		

Data type	Migrates successfully	Will partially migrate	Will not migrate
BINARY	X		
BIT(M)	X		
BLOB	X		
LONGBLOB	X		
MEDIUMBLOB	X		
TINYBLOB	X		
DATE	X		
DATETIME	X		
TIME		X	
TIMESTAMP	X		
YEAR	X		
DOUBLE	X		
FLOAT		X	
VARCHAR(N)	X		
VARBINARY(N)	X		
CHAR(N)	X		
TEXT	X		
LONGTEXT	X		
MEDIUMTEXT	X		
TINYTEXT	X		
JSON	X		
GEOMETRY			X
POINT			X
LINESTRING			X
POLYGON			X
MULTILINESTRING			X
MULTIPOLYGON			X
GEOMETRYCOLLECTION			X
ENUM	X		
SET	X		

## Using any MySQL-compatible database as a source for AWS DMS

Before you begin to work with a MySQL database as a source for AWS DMS, make sure that you have the following prerequisites. These prerequisites apply to either self-managed or AWS-managed sources.

You must have an account for AWS DMS that has the Replication Admin role. The role needs the following privileges:

- **REPLICATION CLIENT** – This privilege is required for CDC tasks only. In other words, full-load-only tasks don't require this privilege.
- **REPLICATION SLAVE** – This privilege is required for CDC tasks only. In other words, full-load-only tasks don't require this privilege.
- **SUPER** – This privilege is required only in MySQL versions before 5.6.6.

The AWS DMS user must also have SELECT privileges for the source tables designated for replication.

## Using a self-managed MySQL-compatible database as a source for AWS DMS

You can use the following self-managed MySQL-compatible databases as sources for AWS DMS:

- MySQL Community Edition
- MySQL Standard Edition
- MySQL Enterprise Edition
- MySQL Cluster Carrier Grade Edition
- MariaDB Community Edition
- MariaDB Enterprise Edition
- MariaDB Column Store

To use CDC, make sure to enable binary logging. To enable binary logging, the following parameters must be configured in MySQL's my.ini (Windows) or my.cnf (UNIX) file.

Parameter	Value
server-id	Set this parameter to a value of 1 or greater.
log-bin	Set the path to the binary log file, such as log-bin=E:\MySql_Logs\BinLog. Don't include the file extension.
binlog_format	Set this parameter to ROW. We recommend this setting during replication because in certain cases when binlog_format is set to STATEMENT, it can cause inconsistency when replicating data to the target. The database engine also writes similar inconsistent data to the target when binlog_format is set to MIXED, because the database engine automatically switches to STATEMENT-based logging which can result in writing inconsistent data on the target database.
expire_logs_days	Set this parameter to a value of 1 or greater. To prevent overuse of disk space, we recommend that you don't use the default value of 0.
binlog_checksum	Set this parameter to NONE.

Parameter	Value
binlog_row_image	Set this parameter to FULL.
log_slave_updates	Set this parameter to TRUE if you are using a MySQL or MariaDB read-replica as a source.

If your source uses the NDB (clustered) database engine, the following parameters must be configured to enable CDC on tables that use that storage engine. Add these changes in MySQL's `my.ini` (Windows) or `my.cnf` (UNIX) file.

Parameter	Value
ndb_log_bin	Set this parameter to ON. This value ensures that changes in clustered tables are logged to the binary log.
ndb_log_update_as_write	Set this parameter to OFF. This value prevents writing UPDATE statements as INSERT statements in the binary log.
ndb_log_updated_only	Set this parameter to OFF. This value ensures that the binary log contains the entire row and not just the changed columns.

## Using an AWS-managed MySQL-compatible database as a source for AWS DMS

You can use the following AWS-managed MySQL-compatible databases as sources for AWS DMS:

- MySQL Community Edition
- MariaDB Community Edition
- Amazon Aurora MySQL-Compatible Edition

When using an AWS-managed MySQL-compatible database as a source for AWS DMS, make sure that you have the following prerequisites for CDC:

- To enable binary logs for RDS for MySQL and for RDS for MariaDB, enable automatic backups at the instance level. To enable binary logs for an Aurora MySQL cluster, change the variable `binlog_format` in the parameter group.

For more information about setting up automatic backups, see [Working with automated backups](#) in the *Amazon RDS User Guide*.

For more information about setting up binary logging for an Amazon RDS for MySQL database, see [Setting the binary logging format](#) in the *Amazon RDS User Guide*.

For more information about setting up binary logging for an Aurora MySQL cluster, see [How do I turn on binary logging for my Amazon Aurora MySQL cluster?](#).

- If you plan to use CDC, turn on binary logging. For more information on setting up binary logging for an Amazon RDS for MySQL database, see [Setting the binary logging format](#) in the *Amazon RDS User Guide*.
- Ensure that the binary logs are available to AWS DMS. Because AWS-managed MySQL-compatible databases purge the binary logs as soon as possible, you should increase the length of time that the logs remain available. For example, to increase log retention to 24 hours, run the following command.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

- Set the `binlog_format` parameter to "ROW".

**Note**

For MariaDB, if the `binlog_format` parameter is switched to ROW for replication purposes, subsequent binary logs are still created in MIXED format. This can prevent DMS from performing change data capture. So, when switching the `binlog_format` parameter for MariaDB, perform a reboot or start then stop your replication task.

- Set the `binlog_row_image` parameter to "Full".
- Set the `binlog_checksum` parameter to "NONE". For more information about setting parameters in Amazon RDS MySQL, see [Working with automated backups](#) in the *Amazon RDS User Guide*.
- If you are using an Amazon RDS MySQL or Amazon RDS MariaDB read replica as a source, enable backups on the read replica, and ensure the `log_slave_updates` parameter is set to TRUE.

## Limitations on using a MySQL database as a source for AWS DMS

When using a MySQL database as a source, consider the following:

- Change data capture (CDC) isn't supported for Amazon RDS MySQL 5.5 or lower. For Amazon RDS MySQL, you must use version 5.6, 5.7, or 8.0 to enable CDC. CDC is supported for self-managed MySQL 5.5 sources.
- For CDC, `CREATE TABLE`, `ADD COLUMN`, and `DROP COLUMN` changing the column data type, and renaming a column are supported. However, `DROP TABLE`, `RENAME TABLE`, and updates made to other attributes, such as column default value, column nullability, character set and so on, are not supported.
- For partitioned tables on the source, when you set **Target table preparation mode** to **Drop tables on target**, AWS DMS creates a simple table without any partitions on the MySQL target. To migrate partitioned tables to a partitioned table on the target, precreate the partitioned tables on the target MySQL database.
- Using an `ALTER TABLE table_name ADD COLUMN column_name` statement to add columns to the beginning (FIRST) or the middle of a table (AFTER) isn't supported. Columns are always added to the end of the table.
- CDC isn't supported when a table name contains uppercase and lowercase characters, and the source engine is hosted on an operating system with case-insensitive file names. An example is Microsoft Windows or OS X using HFS+.
- You can use Aurora MySQL-Compatible Edition Serverless for full load, but you can't use it for CDC. This is because you can't enable the prerequisites for MySQL. For more information, see [Parameter groups and Aurora Serverless v1](#).
- The `AUTO_INCREMENT` attribute on a column isn't migrated to a target database column.
- Capturing changes when the binary logs aren't stored on standard block storage isn't supported. For example, CDC doesn't work when the binary logs are stored on Amazon S3.
- AWS DMS creates target tables with the InnoDB storage engine by default. If you need to use a storage engine other than InnoDB, you must manually create the table and migrate to it using [do nothing](#) mode.
- You can't use Aurora MySQL replicas as a source for AWS DMS unless your DMS migration task mode is [Migrate existing data](#)—full load only.

- If the MySQL-compatible source is stopped during full load, the AWS DMS task doesn't stop with an error. The task ends successfully, but the target might be out of sync with the source. If this happens, either restart the task or reload the affected tables.
- Indexes created on a portion of a column value aren't migrated. For example, the index CREATE INDEX first\_ten\_chars ON customer (name(10)) isn't created on the target.
- In some cases, the task is configured to not replicate LOBs ("SupportLobs" is false in task settings or **Don't include LOB columns** is chosen in the task console). In these cases, AWS DMS doesn't migrate any MEDIUMBLOB, LONGBLOB, MEDIUMTEXT, and LONGTEXT columns to the target.
- BLOB, TINYBLOB, TEXT, and TINYTEXT columns aren't affected and are migrated to the target.
- Temporal data tables or system—versioned tables are not supported on MariaDB source and target databases.
- If migrating between two Amazon RDS Aurora MySQL clusters, the RDS Aurora MySQL source endpoint must be a read/write instance, not a replica instance.
- AWS DMS currently doesn't support views migration for MariaDB.
- AWS DMS doesn't support DDL changes for partitioned tables for MySQL.
- AWS DMS doesn't currently support XA transactions.
- AWS DMS doesn't support GTID for replication.
- AWS DMS doesn't support binary log transaction compression.

## Extra connection attributes when using MySQL as a source for AWS DMS

You can use extra connection attributes to configure a MySQL source. You specify these settings when you create the source endpoint. If you have multiple connection attribute settings, separate them from each other by semicolons with no additional white space (for example, oneSetting;thenAnother).

The following table shows the extra connection attributes available when using Amazon RDS MySQL as a source for AWS DMS.

Name	Description
eventsPollInterval	<p>Specifies how often to check the binary log for new changes/events when the database is idle.</p> <p>Default value: 5</p> <p>Valid values: 1–60</p> <p>Example: eventsPollInterval=5;</p> <p>In the example, AWS DMS checks for changes in the binary logs every five seconds.</p>
executeTimeout	<p>For AWS DMS versions 3.4.7 and later, sets the client statement timeout for a MySQL source endpoint, in seconds.</p> <p>Default value: 60</p> <p>Example: executeTimeout=1500;</p> <p>For this example, any statement that AWS DMS executes against a MySQL source database times out if it takes longer than 1500 seconds.</p>

Name	Description
serverTimezone	<p>Specifies the time zone for the source MySQL database.</p> <p>Example: <code>serverTimezone=US/Pacific;</code></p> <p>Don't enclose time zone data in single quotation marks.</p>
afterConnectScript	<p>Specifies a script to run immediately after AWS DMS connects to the endpoint. The migration task continues running regardless if the SQL statement succeeds or fails.</p> <p>Valid values: One or more valid SQL statements, set off by a semicolon.</p> <p>Example: <code>afterConnectScript=ALTER SESSION SET CURRENT_SCHEMA = system;</code></p>
CleanSrcMetadataOnMismatch	<p>Cleans and recreates table metadata information on the replication instance when a mismatch occurs. For example, in a situation where running an alter DDL on the table could result in different information about the table cached in the replication instance. Boolean.</p> <p>Default value: <code>false</code></p> <p>Example: <code>CleanSrcMetadataOnMismatch=false;</code></p>

## Source data types for MySQL

The following table shows the MySQL database source data types that are supported when using AWS DMS and the default mapping from AWS DMS data types.

For information on how to view the data type that is mapped in the target, see the section for the target endpoint you are using.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

MySQL data types	AWS DMS data types
INT	INT4
BIGINT	INT8
MEDIUMINT	INT4
TINYINT	INT1
SMALLINT	INT2
UNSIGNED TINYINT	UINT1
UNSIGNED SMALLINT	UINT2
UNSIGNED MEDIUMINT	UINT4
UNSIGNED INT	UINT4

MySQL data types	AWS DMS data types
UNSIGNED BIGINT	UINT8
DECIMAL(10)	NUMERIC (10,0)
BINARY	BYTES(1)
BIT	BOOLEAN
BIT(64)	BYTES(8)
BLOB	BYTES(65535)
LONGBLOB	BLOB
MEDIUMBLOB	BLOB
TINYBLOB	BYTES(255)
DATE	DATE
DATETIME	<p>DATETIME</p> <p>DATETIME without a parenthetical value is replicated without milliseconds. DATETIME with a parenthetical value of 1 to 5 (such as DATETIME(5)) is replicated with milliseconds.</p> <p>When replicating a DATETIME column, the time remains the same on the target. It is not converted to UTC.</p>
TIME	STRING
TIMESTAMP	<p>DATETIME</p> <p>When replicating a TIMESTAMP column, the time is converted to UTC on the target.</p>
YEAR	INT2
DOUBLE	REAL8
FLOAT	<p>REAL(DOUBLE)</p> <p>If the FLOAT values are not in the range following, use a transformation to map FLOAT to STRING. For more information about transformations, see <a href="#">Transformation rules and actions (p. 500)</a>.</p> <p>The supported FLOAT range is -1.79E+308 to -2.23E-308, 0, and 2.23E-308 to 1.79E+308</p>
VARCHAR (45)	WSTRING (45)
VARCHAR (2000)	WSTRING (2000)
VARCHAR (4000)	WSTRING (4000)
VARBINARY (4000)	BYTES (4000)

MySQL data types	AWS DMS data types
VARBINARY (2000)	BYTES (2000)
CHAR	WSTRING
TEXT	WSTRING
LONGTEXT	NCLOB
MEDIUMTEXT	NCLOB
TINYTEXT	WSTRING(255)
GEOMETRY	BLOB
POINT	BLOB
LINESTRING	BLOB
POLYGON	BLOB
MULTIPOINT	BLOB
MULTILINESTRING	BLOB
MULTIPOLYGON	BLOB
GEOMETRYCOLLECTION	BLOB
ENUM	WSTRING ( <i>length</i> )  Here, <i>length</i> is the length of the longest value in the ENUM.
SET	WSTRING ( <i>length</i> )  Here, <i>length</i> is the total length of all values in the SET, including commas.
JSON	CLOB

#### Note

In some cases, you might specify the DATETIME and TIMESTAMP data types with a "zero" value (that is, 0000-00-00). If so, make sure that the target database in the replication task supports "zero" values for the DATETIME and TIMESTAMP data types. Otherwise, these values are recorded as null on the target.

## Using an SAP ASE database as a source for AWS DMS

You can migrate data from an SAP Adaptive Server Enterprise (ASE) database—formerly known as Sybase—using AWS DMS. With an SAP ASE database as a source, you can migrate data to any of the other supported AWS DMS target databases. AWS DMS supports SAP ASE versions 12.5.3 or higher, 15, 15.5, 15.7, 16 and later as sources.

For additional details on working with SAP ASE databases and AWS DMS, see the following sections.

#### Topics

- [Prerequisites for using an SAP ASE database as a source for AWS DMS \(p. 226\)](#)

- Limitations on using SAP ASE as a source for AWS DMS (p. 226)
- Permissions required for using SAP ASE as a source for AWS DMS (p. 227)
- Removing the truncation point (p. 227)
- Extra connection attributes when using SAP ASE as a source for AWS DMS (p. 227)
- Source data types for SAP ASE (p. 229)

## Prerequisites for using an SAP ASE database as a source for AWS DMS

For an SAP ASE database to be a source for AWS DMS, do the following:

- Enable SAP ASE replication for tables by using the `sp_setreptable` command. For more information, see [Sybase Infocenter Archive](#).
- Disable RepAgent on the SAP ASE database. For more information, see [Stop and disable the RepAgent thread in the primary database](#).
- To replicate to SAP ASE version 15.7 on an Windows EC2 instance configured for non-Latin characters (for example, Chinese), install SAP ASE 15.7 SP121 on the target computer.

### Note

For ongoing change data capture (CDC) replication, DMS runs `dbcc logtransfer` and `dbcc log` to read data from the transaction log.

## Limitations on using SAP ASE as a source for AWS DMS

The following limitations apply when using an SAP ASE database as a source for AWS DMS:

- You can run only one AWS DMS task with ongoing replication or CDC for each SAP ASE database. You can run multiple full-load-only tasks in parallel.
- You can't rename a table. For example, the following command fails.

```
sp_rename 'Sales.SalesRegion', 'SalesReg';
```

- You can't rename a column. For example, the following command fails.

```
sp_rename 'Sales.Sales.Region', 'RegID', 'COLUMN';
```

- Zero values located at the end of binary data type strings are truncated when replicated to the target database. For example, `0x00000000000000000000000000000000100000100000000` in the source table becomes `0x000000000000000000000000000000001000001` in the target table.
- If the database default is set not to allow NULL values, AWS DMS creates the target table with columns that don't allow NULL values. Consequently, if a full load or CDC replication task contains empty values, AWS DMS throws an error. You can prevent these errors by allowing NULL values in the source database by using the following commands.

```
sp_dboption database_name, 'allow nulls by default', 'true'  
go  
use database_name  
CHECKPOINT  
go
```

- The `reorg rebuild` index command isn't supported.
- AWS DMS does not support clusters or using MSA (Multi-Site Availability)/Warm Standby as a source.

- When AR\_H\_TIMESTAMP transformation header expression is used in mapping rules, the milliseconds won't be captured for an added column.
- Running Merge operations during CDC will result in a non-recoverable error. To bring the target back in sync, run a full load.
- Rollback trigger events are not supported for tables that use a data row locking scheme.

## Permissions required for using SAP ASE as a source for AWS DMS

To use an SAP ASE database as a source in an AWS DMS task, you need to grant permissions. Grant the user account specified in the AWS DMS database definitions the following permissions in the SAP ASE database:

- sa\_role
- replication\_role
- sybase\_ts\_role
- By default, where you need to have permission to run the sp\_setreptable stored procedure, AWS DMS enables the SAP ASE replication option. If you want to run sp\_setreptable on a table directly from the database endpoint and not through AWS DMS itself, you can use the enableReplication extra connection attribute. For more information, see [Extra connection attributes when using SAP ASE as a source for AWS DMS \(p. 227\)](#).

## Removing the truncation point

When a task starts, AWS DMS establishes a \$replication\_truncation\_point entry in the syslogshold system view, indicating that a replication process is in progress. While AWS DMS is working, it advances the replication truncation point at regular intervals, according to the amount of data that has already been copied to the target.

After the \$replication\_truncation\_point entry is established, keep the AWS DMS task running to prevent the database log from becoming excessively large. If you want to stop the AWS DMS task permanently, remove the replication truncation point by issuing the following command:

```
dbcc settrunc('ltm','ignore')
```

After the truncation point is removed, you can't resume the AWS DMS task. The log continues to be truncated automatically at the checkpoints (if automatic truncation is set).

## Extra connection attributes when using SAP ASE as a source for AWS DMS

You can use extra connection attributes to configure an SAP ASE source. You specify these settings when you create the source endpoint. You must separate multiple extra connection attribute settings from each other by semicolons and no additional white space.

The following table shows the extra connection attributes available when using SAP ASE as a source for AWS DMS.

Name	Description
charset	Set this attribute to the SAP ASE name that corresponds to the international character set.

Name	Description
	<p>Default value: <code>iso_1</code></p> <p>Example: <code>charset=utf8;</code></p> <p>Valid values:</p> <ul style="list-style-type: none"><li>• <code>acsii_8</code></li><li>• <code>big5hk</code></li><li>• <code>cp437</code></li><li>• <code>cp850</code></li><li>• <code>cp852</code></li><li>• <code>cp852</code></li><li>• <code>cp855</code></li><li>• <code>cp857</code></li><li>• <code>cp858</code></li><li>• <code>cp860</code></li><li>• <code>cp864</code></li><li>• <code>cp866</code></li><li>• <code>cp869</code></li><li>• <code>cp874</code></li><li>• <code>cp932</code></li><li>• <code>cp936</code></li><li>• <code>cp950</code></li><li>• <code>cp1250</code></li><li>• <code>cp1251</code></li><li>• <code>cp1252</code></li><li>• <code>cp1253</code></li><li>• <code>cp1254</code></li><li>• <code>cp1255</code></li><li>• <code>cp1256</code></li><li>• <code>cp1257</code></li><li>• <code>cp1258</code></li><li>• <code>deckanji</code></li><li>• <code>euccns</code></li><li>• <code>eucgb</code></li><li>• <code>eucjis</code></li><li>• <code>eucksc</code></li><li>• <code>gb18030</code></li><li>• <code>greek8</code></li><li>• <code>iso_1</code></li><li>• <code>iso88592</code></li><li>• <code>iso88595</code></li><li>• <code>iso88596</code></li><li>• <code>iso88597</code></li><li>• <code>iso88598</code></li></ul>

Name	Description
	<ul style="list-style-type: none"> <li>• iso88599</li> <li>• iso15</li> <li>• kz1048</li> <li>• koi8</li> <li>• roman8</li> <li>• iso88599</li> <li>• sjis</li> <li>• tis620</li> <li>• turkish8</li> <li>• utf8</li> </ul> <p>For any further questions about supported character sets in a SAP ASE database, see <a href="#">Adaptive Server Enterprise: Supported character sets</a>.</p>
enableReplication	<p>Set this attribute if you want to enable <code>sp_setreptable</code> on tables from the database end and not through AWS DMS.</p> <p>Default value: <code>true</code></p> <p>Example: <code>enableReplication=false;</code></p> <p>Valid values: <code>true</code> or <code>false</code></p>
encryptPassword	<p>Set this attribute if you have enabled "net password encryption reqd" at the source database.</p> <p>Default value: <code>0</code></p> <p>Example: <code>encryptPassword=1;</code></p> <p>Valid values: <code>0</code>, <code>1</code>, or <code>2</code></p> <p>For more information on these parameter values, see <a href="#">Adaptive Server Enterprise: Using the EncryptPassword Connection string property</a>.</p>
provider	<p>Set this attribute if you want to use Transport Layer Security (TLS) 1.1 or 1.2 for versions of ASE 15.7 and later.</p> <p>Default value: Adaptive Server Enterprise</p> <p>Example: <code>provider=Adaptive Server Enterprise 16.03.06;</code></p> <p>Valid values: Adaptive Server Enterprise 16.03.06</p>

## Source data types for SAP ASE

For a list of the SAP ASE source data types that are supported when using AWS DMS and the default mapping from AWS DMS data types, see the following table. AWS DMS doesn't support SAP ASE source tables with columns of the user-defined type (UDT) data type. Replicated columns with this data type are created as NULL.

For information on how to view the data type that is mapped in the target, see the [Targets for data migration \(p. 293\)](#) section for your target endpoint.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

SAP ASE data types	AWS DMS data types
BIGINT	INT8
UNSIGNED BIGINT	UINT8
INT	INT4
UNSIGNED INT	UINT4
SMALLINT	INT2
UNSIGNED SMALLINT	UINT2
TINYINT	UINT1
DECIMAL	NUMERIC
NUMERIC	NUMERIC
FLOAT	REAL8
DOUBLE	REAL8
REAL	REAL4
MONEY	NUMERIC
SMALLMONEY	NUMERIC
DATETIME	DATETIME
BIGDATETIME	DATETIME(6)
SMALLDATETIME	DATETIME
DATE	DATE
TIME	TIME
BIGTIME	TIME
CHAR	STRING
UNICHAR	WSTRING
NCHAR	WSTRING
VARCHAR	STRING
UNIVARCHAR	WSTRING
NVARCHAR	WSTRING
BINARY	BYTES
VARBINARY	BYTES

SAP ASE data types	AWS DMS data types
BIT	BOOLEAN
TEXT	CLOB
UNITEXT	NCLOB
IMAGE	BLOB

## Using MongoDB as a source for AWS DMS

AWS DMS supports MongoDB versions 3.x and 4.0 as a database source. Starting with AWS DMS 3.4.5, AWS DMS supports MongoDB versions 4.2 and 4.4. Starting with MongoDB version 4.2, AWS DMS 3.4.5 and later supports distributed transactions. For more information on MongoDB distributed transactions, see [Transactions](#) in MongoDB.

If you are new to MongoDB, be aware of the following important MongoDB database concepts:

- A record in MongoDB is a *document*, which is a data structure composed of field and value pairs. The value of a field can include other documents, arrays, and arrays of documents. A document is roughly equivalent to a row in a relational database table.
- A *collection* in MongoDB is a group of documents, and is roughly equivalent to a relational database table.
- A *database* in MongoDB is a set of collections, and is roughly equivalent to a schema in a relational database.
- Internally, a MongoDB document is stored as a binary JSON (BSON) file in a compressed format that includes a type for each field in the document. Each document has a unique ID.

AWS DMS supports two migration modes when using MongoDB as a source, *Document mode* or *Table mode*. You specify which migration mode to use when you create the MongoDB endpoint or by setting the **Metadata mode** parameter from the AWS DMS console. Optionally, you can create a second column named `_id` that acts as the primary key by selecting the check mark button for **\_id as a separate column** in the endpoint configuration panel.

Your choice of migration mode affects the resulting format of the target data, as explained following.

### Document mode

In document mode, the MongoDB document is migrated as is, meaning that the document data is consolidated into a single column named `_doc` in a target table. Document mode is the default setting when you use MongoDB as a source endpoint.

For example, consider the following documents in a MongoDB collection called `myCollection`.

```
> db.myCollection.find()
{ "_id" : ObjectId("5a94815f40bd44d1b02bdfe0"), "a" : 1, "b" : 2, "c" : 3 }
{ "_id" : ObjectId("5a94815f40bd44d1b02bdfe1"), "a" : 4, "b" : 5, "c" : 6 }
```

After migrating the data to a relational database table using document mode, the data is structured as follows. The data fields in the MongoDB document are consolidated into the `_doc` column.

oid_id	_doc
5a94815f40bd44d1b02bdfe0	{ "a" : 1, "b" : 2, "c" : 3 }

5a94815f40bd44d1b02bdfe1

{ "a" : 4, "b" : 5, "c" : 6 }

You can optionally set the extra connection attribute `extractDocID` to `true` to create a second column named `"_id"` that acts as the primary key. If you are going to use CDC, set this parameter to `true`.

In document mode, AWS DMS manages the creation and renaming of collections like this:

- If you add a new collection to the source database, AWS DMS creates a new target table for the collection and replicates any documents.
- If you rename an existing collection on the source database, AWS DMS doesn't rename the target table.

If the target endpoint is Amazon DocumentDB, run the migration in **Document mode**.

#### Table mode

In table mode, AWS DMS transforms each top-level field in a MongoDB document into a column in the target table. If a field is nested, AWS DMS flattens the nested values into a single column. AWS DMS then adds a key field and data types to the target table's column set.

For each MongoDB document, AWS DMS adds each key and type to the target table's column set. For example, using table mode, AWS DMS migrates the previous example into the following table.

oid_id	a	b	c
5a94815f40bd44d1b02bdfe0		2	3
5a94815f40bd44d1b02bdfe1		5	6

Nested values are flattened into a column containing dot-separated key names. The column is named the concatenation of the flattened field names separated by periods. For example, AWS DMS migrates a JSON document with a field of nested values such as `{"a" : {"b" : {"c": 1}}}` into a column named `a.b.c`.

To create the target columns, AWS DMS scans a specified number of MongoDB documents and creates a set of all the fields and their types. AWS DMS then uses this set to create the columns of the target table. If you create or modify your MongoDB source endpoint using the console, you can specify the number of documents to scan. The default value is 1000 documents. If you use the AWS CLI, you can use the extra connection attribute `docsToInvestigate`.

In table mode, AWS DMS manages documents and collections like this:

- When you add a document to an existing collection, the document is replicated. If there are fields that don't exist in the target, those fields aren't replicated.
- When you update a document, the updated document is replicated. If there are fields that don't exist in the target, those fields aren't replicated.
- Deleting a document is fully supported.
- Adding a new collection doesn't result in a new table on the target when done during a CDC task.
- Renaming a collection isn't supported.

## Permissions needed when using MongoDB as a source for AWS DMS

For an AWS DMS migration with a MongoDB source, you can create either a user account with root privileges, or a user with permissions only on the database to migrate.

The following code creates a user to be the root account.

```
use admin
db.createUser(
{
  user: "root",
  pwd: "password",
  roles: [ { role: "root", db: "admin" } ]
}
)
```

For a MongoDB 3.x source, the following code creates a user with minimal privileges on the database to be migrated.

```
use database_to_migrate
db.createUser(
{
  user: "dms-user",
  pwd: "password",
  roles: [ { role: "read", db: "local" }, "read" ]
})
```

For a MongoDB 4.x source, the following code creates a user with minimal privileges.

```
{ resource: { db: "", collection: "" }, actions: [ "find", "changeStream" ] }
```

For example, create the following role in the "admin" database.

```
use admin
db.createRole(
{
  role: "changestreamrole",
  privileges: [
    { resource: { db: "", collection: "" }, actions: [ "find", "changeStream" ] }
  ],
  roles: []
}
)
```

And once the role is created, create a user in the database to be migrated.

```
> use test
> db.createUser(
{
  user: "dms-user12345",
  pwd: "password",
  roles: [ { role: "changestreamrole", db: "admin" }, "read" ]
})
```

## Configuring a MongoDB replica set for CDC

To use ongoing replication or CDC with MongoDB, AWS DMS requires access to the MongoDB operations log (oplog). To create the oplog, you need to deploy a replica set if one doesn't exist. For more information, see [the MongoDB documentation](#).

You can use CDC with either the primary or secondary node of a MongoDB replica set as the source endpoint.

### To convert a standalone instance to a replica set

1. Using the command line, connect to mongo.

```
mongo localhost
```

2. Stop the mongod service.

```
service mongod stop
```

3. Restart mongod using the following command:

```
mongod --replSet "rs0" --auth -port port_number
```

4. Test the connection to the replica set using the following commands:

```
mongo -u root -p password --host rs0/localhost:port_number  
--authenticationDatabase "admin"
```

If you plan to perform a document mode migration, select option `_id` as a separate column when you create the MongoDB endpoint. Selecting this option creates a second column named `_id` that acts as the primary key. This second column is required by AWS DMS to support data manipulation language (DML) operations.

## Security requirements when using MongoDB as a source for AWS DMS

AWS DMS supports two authentication methods for MongoDB. The two authentication methods are used to encrypt the password, so they are only used when the `authType` parameter is set to `PASSWORD`.

The MongoDB authentication methods are as follows:

- **MONGODB-CR** – For backward compatibility
- **SCRAM-SHA-1** – The default when using MongoDB version 3.x and 4.0

If an authentication method isn't specified, AWS DMS uses the default method for the version of the MongoDB source.

## Segmenting MongoDB collections and migrating in parallel

To improve performance of a migration task, MongoDB source endpoints support two options for parallel full load in table mapping.

In other words, you can migrate a collection in parallel by using either autosegmentation or range segmentation with table mapping for a parallel full load in JSON settings. With autosegmentation, you can specify the criteria for AWS DMS to automatically segment your source for migration in each thread. With range segmentation, you can tell AWS DMS the specific range of each segment for DMS to migrate in each thread. For more information on these settings, see [Table and collection settings rules and operations \(p. 523\)](#).

### Migrating a MongoDB database in parallel using autosegmentation ranges

You can migrate your documents in parallel by specifying the criteria for AWS DMS to automatically partition (segment) your data for each thread. In particular, you specify the number of documents

to migrate per thread. Using this approach, AWS DMS attempts to optimize segment boundaries for maximum performance per thread.

You can specify the segmentation criteria using the table-settings options following in table mapping.

Table-settings option	Description
"type"	(Required) Set to "partitions-auto" for MongoDB as a source.
"number-of-partitions"	(Optional) Total number of partitions (segments) used for migration. The default is 16.
"collection-count-from-metadata"	(Optional) If this option is set to true, AWS DMS uses an estimated collection count for determining the number of partitions. If this option is set to false, AWS DMS uses the actual collection count. The default is true.
"max-records-skip-per-page"	(Optional) The number of records to skip at once when determining the boundaries for each partition. AWS DMS uses a paginated skip approach to determine the minimum boundary for a partition. The default is 10,000.  Setting a relatively large value can result in cursor timeouts and task failures. Setting a relatively low value results in more operations per page and a slower full load.
"batch-size"	(Optional) Limits the number of documents returned in one batch. Each batch requires a round trip to the server. If the batch size is zero (0), the cursor uses the server-defined maximum batch size. The default is 0.

The example following shows a table mapping for autosegmentation.

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "admin",
        "table-name": "departments"
      },
      "rule-action": "include",
      "filters": []
    },
    {
      "rule-type": "table-settings",
      "rule-id": "2",
      "rule-name": "2",
      "object-locator": {
        "schema-name": "admin",
        "table-name": "departments"
      },
    }
  ]
}
```

```
    "parallel-load": {  
        "type": "partitions-auto",  
        "number-of-partitions": 5,  
        "collection-count-from-metadata": "true",  
        "max-records-skip-per-page": 1000000,  
        "batch-size": 50000  
    }  
}  
]  
}
```

Autosegmentation has the limitation following. The migration for each segment fetches the collection count and the minimum `_id` for the collection separately. It then uses a paginated skip to calculate the minimum boundary for that segment.

Therefore, ensure that the minimum `_id` value for each collection remains constant until all the segment boundaries in the collection are calculated. If you change the minimum `_id` value for a collection during its segment boundary calculation, it can cause data loss or duplicate row errors.

## Migrating a MongoDB database in parallel using range segmentation

You can migrate your documents in parallel by specifying the ranges for each segment in a thread. Using this approach, you tell AWS DMS the specific documents to migrate in each thread according to your choice of document ranges per thread.

The image following shows a MongoDB collection that has seven items, and `_id` as the primary key.

Key	Value
▼ (1) ObjectId("5f805c74873173399a278d78") { 3 fields }	
└ _id	ObjectId("5f805c74873173399a278d78")
└ num	1
└ name	a
▼ (2) ObjectId("5f805c97873173399a278d79") { 3 fields }	
└ _id	ObjectId("5f805c97873173399a278d79")
└ num	2
└ name	b
▼ (3) ObjectId("5f805cb0873173399a278d7a") { 3 fields }	
└ _id	ObjectId("5f805cb0873173399a278d7a")
└ num	3
└ name	c
▼ (4) ObjectId("5f805ccb873173399a278d7b") { 3 fields }	
└ _id	ObjectId("5f805ccb873173399a278d7b")
└ num	4
└ name	d
▼ (5) ObjectId("5f805cc5873173399a278d7c") { 3 fields }	
└ _id	ObjectId("5f805cc5873173399a278d7c")
└ num	5
└ name	e
▼ (6) ObjectId("5f805cd0873173399a278d7d") { 3 fields }	
└ _id	ObjectId("5f805cd0873173399a278d7d")
└ num	6
└ name	f
▼ (7) ObjectId("5f805cdd873173399a278d7e") { 3 fields }	
└ _id	ObjectId("5f805cdd873173399a278d7e")
└ num	7
└ name	g

To split the collection into three specific segments for AWS DMS to migrate in parallel, you can add table mapping rules to your migration task. This approach is shown in the following JSON example.

```
{
  // Task table mappings:
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "testdatabase",
        "table-name": "testtable"
      },
      "rule-action": "include"
    }, // "selection" :"rule-type"
    {
      "rule-type": "table-settings",
      "rule-id": "2",
      "rule-name": "2",
      "object-locator": {
        "schema-name": "testdatabase",
        "table-name": "testtable"
      },
      "parallel-load": {
        "type": "ranges",
        "columns": [
          "_id",
          "num"
        ],
        "boundaries": [
          // First segment selects documents with _id less-than-or-equal-to
          5f805c97873173399a278d79
          // and num less-than-or-equal-to 2.
          [
            "5f805c97873173399a278d79",
            "2"
          ],
          // Second segment selects documents with _id > 5f805c97873173399a278d79 and
          // _id less-than-or-equal-to 5f805cc5873173399a278d7c and
          // num > 2 and num less-than-or-equal-to 5.
          [
            "5f805cc5873173399a278d7c",
            "5"
          ]
          // Third segment is implied and selects documents with _id >
          5f805cc5873173399a278d7c.
        ] // :"boundaries"
      } // :"parallel-load"
    } // "table-settings" :"rule-type"
  ] // :"rules"
} // :Task table mappings
}
```

That table mapping definition splits the source collection into three segments and migrates in parallel. The following are the segmentation boundaries.

```
Data with _id less-than-or-equal-to "5f805c97873173399a278d79" and num less-than-or-equal-to 2 (2 records)
Data with _id > "5f805c97873173399a278d79" and num > 2 and _id less-than-or-equal-to
"5f805cc5873173399a278d7c" and num less-than-or-equal-to 5 (3 records)
Data with _id > "5f805cc5873173399a278d7c" and num > 5 (2 records)
```

After the migration task is complete, you can verify from the task logs that the tables loaded in parallel, as shown in the following example. You can also verify the MongoDB find clause used to unload each segment from the source table.

```
[TASK_MANAGER    ] I: Start loading segment #1 of 3 of table
'testdatabase'.'testtable' (Id = 1) by subtask 1. Start load timestamp 0005B191D638FE86
(replicationtask_util.c:752)

[SOURCE_UNLOAD   ] I: Range Segmentation filter for Segment #0 is initialized.
(mongodb_unload.c:157)

[SOURCE_UNLOAD   ] I: Range Segmentation filter for Segment #0 is: { "_id" : { "$lte" :
{ "$oid" : "5f805c97873173399a278d79" } }, "num" : { "$lte" : { "$numberInt" : "2" } } }
(mongodb_unload.c:328)

[SOURCE_UNLOAD   ] I: Unload finished for segment #1 of segmented table
'testdatabase'.'testtable' (Id = 1). 2 rows sent.

[TASK_MANAGER    ] I: Start loading segment #1 of 3 of table 'testdatabase'.'testtable' (Id
= 1) by subtask 1. Start load timestamp 0005B191D638FE86 (replicationtask_util.c:752)

[SOURCE_UNLOAD   ] I: Range Segmentation filter for Segment #0 is initialized.
(mongodb_unload.c:157)

[SOURCE_UNLOAD   ] I: Range Segmentation filter for Segment #0 is: { "_id" : { "$lte" :
{ "$oid" : "5f805c97873173399a278d79" } }, "num" : { "$lte" : { "$numberInt" : "2" } } }
(mongodb_unload.c:328)

[SOURCE_UNLOAD   ] I: Unload finished for segment #1 of segmented table
'testdatabase'.'testtable' (Id = 1). 2 rows sent.

[TARGET_LOAD     ] I: Load finished for segment #1 of segmented table
'testdatabase'.'testtable' (Id = 1). 1 rows received. 0 rows skipped. Volume transferred
480.

[TASK_MANAGER    ] I: Load finished for segment #1 of table 'testdatabase'.'testtable' (Id
= 1) by subtask 1. 2 records transferred.
```

Currently, AWS DMS supports the following MongoDB data types as a segment key column:

- Double
- String
- ObjectId
- 32 bit integer
- 64 bit integer

## Migrating multiple databases when using MongoDB as a source for AWS DMS

AWS DMS versions 3.4.5 and later support migrating multiple databases in a single task for all supported MongoDB versions. If you want to migrate multiple databases, take these steps:

1. When you create the MongoDB source endpoint, do one of the following:
  - On the DMS console's **Create endpoint** page, make sure that **Database name** is empty under **Endpoint configuration**.
  - Using the AWS CLI `CreateEndpoint` command, assign an empty string value to the `DatabaseName` parameter in `MongoDBSettings`.
2. For each database that you want to migrate from a MongoDB source, specify the database name as a schema name in the table mapping for the task. You can do this using either the guided input

in the console or directly in JSON. For more information on the guided input, see [Specifying table selection and transformations rules from the console \(p. 491\)](#). For more information on the JSON, see [Selection rules and actions \(p. 495\)](#).

For example, you might specify the JSON following to migrate three MongoDB databases.

### Example Migrate all tables in a schema

The JSON following migrates all tables from the Customers, Orders, and Suppliers databases in your source endpoint to your target endpoint.

```
{  
    "rules": [  
        {  
            "rule-type": "selection",  
            "rule-id": "1",  
            "rule-name": "1",  
            "object-locator": {  
                "schema-name": "Customers",  
                "table-name": "%"  
            },  
            "rule-action": "include",  
            "filters": []  
        },  
        {  
            "rule-type": "selection",  
            "rule-id": "2",  
            "rule-name": "2",  
            "object-locator": {  
                "schema-name": "Orders",  
                "table-name": "%"  
            },  
            "rule-action": "include",  
            "filters": []  
        },  
        {  
            "rule-type": "selection",  
            "rule-id": "3",  
            "rule-name": "3",  
            "object-locator": {  
                "schema-name": "Inventory",  
                "table-name": "%"  
            },  
            "rule-action": "include",  
            "filters": []  
        }  
    ]  
}
```

## Limitations when using MongoDB as a source for AWS DMS

The following are limitations when using MongoDB as a source for AWS DMS:

- When the `_id` option is set as a separate column, the ID string can't exceed 200 characters.
- Object ID and array type keys are converted to columns that are prefixed with `oid` and `array` in table mode.

Internally, these columns are referenced with the prefixed names. If you use transformation rules in AWS DMS that reference these columns, make sure to specify the prefixed column. For example, you specify  `${oid__id}`  and not  `${_id}` , or  `${array__addresses}`  and not  `${_addresses}` .

- Collection names and key names can't include the dollar symbol (\$).
- Table mode and document mode have the limitations described preceding.
- Migrating in parallel using autosegmentation has the limitations described preceding.
- Source filters aren't supported for MongoDB.
- AWS DMS doesn't support documents where the nesting level is greater than 97.

## Endpoint configuration settings when using MongoDB as a source for AWS DMS

When you set up your MongoDB source endpoint, you can specify multiple endpoint configuration settings using the AWS DMS console.

The following table describes the configuration settings available when using MongoDB databases as an AWS DMS source.

Setting (attribute)	Valid values	Default value and description
<b>Authentication mode</b>	"none" "password"	The value "password" prompts for a user name and password. When "none" is specified, user name and password parameters aren't used.
<b>Authentication source</b>	A valid MongoDB database name.	The name of the MongoDB database that you want to use to validate your credentials for authentication. The default value is "admin".
<b>Authentication mechanism</b>	"default" "mongodb_cr" "scram_sha_1"	The authentication mechanism. The value "default" is "scram_sha_1". This setting isn't used when authType is set to "no".
<b>Metadata mode</b>	Document and table	Chooses document mode or table mode.
<b>Number of documents to scan (docsToInvestigate)</b>	A positive integer greater than 0.	Use this option in table mode only to define the target table definition.
<b>_id as a separate column</b>	Check mark in box	Optional check mark box that creates a second column named _id that acts as the primary key.

If you choose **Document** as **Metadata mode**, different options are available.

If the target endpoint is DocumentDB, make sure to run the migration in **Document mode**. Also, modify your source endpoint and select the option **\_id as separate column**. This is a mandatory prerequisite if your source MongoDB workload involves transactions.

## Source data types for MongoDB

Data migration that uses MongoDB as a source for AWS DMS supports most MongoDB data types. In the following table, you can find the MongoDB source data types that are supported when using AWS DMS.

and the default mapping from AWS DMS data types. For more information about MongoDB data types, see [BSON types](#) in the MongoDB documentation.

For information on how to view the data type that is mapped in the target, see the section for the target endpoint that you are using.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

MongoDB data types	AWS DMS data types
Boolean	Bool
Binary	BLOB
Date	Date
Timestamp	Date
Int	INT4
Long	INT8
Double	REAL8
String (UTF-8)	CLOB
Array	CLOB
OID	String
REGEX	CLOB
CODE	CLOB

## Using Amazon DocumentDB (with MongoDB compatibility) as a source for AWS DMS

AWS DMS supports Amazon DocumentDB (with MongoDB compatibility) versions 3.6 and 4.0 as a database source. Using Amazon DocumentDB as a source, you can migrate data from one Amazon DocumentDB cluster to another Amazon DocumentDB cluster. You can also migrate data from an Amazon DocumentDB cluster to one of the other target endpoints supported by AWS DMS.

If you are new to Amazon DocumentDB, be aware of the following important concepts for Amazon DocumentDB databases:

- A record in Amazon DocumentDB is a *document*, a data structure composed of field and value pairs. The value of a field can include other documents, arrays, and arrays of documents. A document is roughly equivalent to a row in a relational database table.
- A *collection* in Amazon DocumentDB is a group of documents, and is roughly equivalent to a relational database table.
- A *database* in Amazon DocumentDB is a set of collections, and is roughly equivalent to a schema in a relational database.

AWS DMS supports two migration modes when using Amazon DocumentDB as a source, document mode and table mode. You specify the migration mode when you create the Amazon DocumentDB source

endpoint in the AWS DMS console, using either the **Metadata mode** option or the extra connection attribute `nestingLevel`. Following, you can find an explanation how the choice of migration mode affects the resulting format of the target data.

### Document mode

In *document mode*, the JSON document is migrated as is. That means the document data is consolidated into one of two items. When you use a relational database as a target, the data is a single column named `_doc` in a target table. When you use a nonrelational database as a target, the data is a single JSON document. Document mode is the default mode, which we recommend when migrating to an Amazon DocumentDB target.

For example, consider the following documents in a Amazon DocumentDB collection called `myCollection`.

```
> db.myCollection.find()
{ "_id" : ObjectId("5a94815f40bd44d1b02bdfe0"), "a" : 1, "b" : 2, "c" : 3 }
{ "_id" : ObjectId("5a94815f40bd44d1b02bdfe1"), "a" : 4, "b" : 5, "c" : 6 }
```

After migrating the data to a relational database table using document mode, the data is structured as follows. The data fields in the document are consolidated into the `_doc` column.

oid_id	_doc
5a94815f40bd44d1b02bdfe0	{ "a" : 1, "b" : 2, "c" : 3 }
5a94815f40bd44d1b02bdfe1	{ "a" : 4, "b" : 5, "c" : 6 }

You can optionally set the extra connection attribute `extractDocID` to `true` to create a second column named `_id` that acts as the primary key. If you are going to use change data capture (CDC), set this parameter to `true` except when using Amazon DocumentDB as the target.

#### Note

If you add a new collection to the source database, AWS DMS creates a new target table for the collection and replicates any documents.

### Table mode

In *table mode*, AWS DMS transforms each top-level field in a Amazon DocumentDB document into a column in the target table. If a field is nested, AWS DMS flattens the nested values into a single column. AWS DMS then adds a key field and data types to the target table's column set.

For each Amazon DocumentDB document, AWS DMS adds each key and type to the target table's column set. For example, using table mode, AWS DMS migrates the previous example into the following table.

oid_id	a	b	c
5a94815f40bd44d1b02bdfe0	1	2	3
5a94815f40bd44d1b02bdfe1	4	5	6

Nested values are flattened into a column containing dot-separated key names. The column is named using the concatenation of the flattened field names separated by periods. For example, AWS DMS migrates a JSON document with a field of nested values such as `{"a" : {"b" : {"c" : 1}}}` into a column named `a.b.c`.

To create the target columns, AWS DMS scans a specified number of Amazon DocumentDB documents and creates a set of all the fields and their types. AWS DMS then uses this set to create the columns of the target table. If you create or modify your Amazon DocumentDB source endpoint using the console, you can specify the number of documents to scan. The default value is 1,000 documents. If you use the AWS CLI, you can use the extra connection attribute `docsToInvestigate`.

In table mode, AWS DMS manages documents and collections like this:

- When you add a document to an existing collection, the document is replicated. If there are fields that don't exist in the target, those fields aren't replicated.
- When you update a document, the updated document is replicated. If there are fields that don't exist in the target, those fields aren't replicated.
- Deleting a document is fully supported.
- Adding a new collection doesn't result in a new table on the target when done during a CDC task.
- Renaming a collection isn't supported.

### Topics

- [Setting permissions to use Amazon DocumentDB as a source \(p. 244\)](#)
- [Configuring CDC for an Amazon DocumentDB cluster \(p. 245\)](#)
- [Connecting to Amazon DocumentDB using TLS \(p. 245\)](#)
- [Creating an Amazon DocumentDB source endpoint \(p. 246\)](#)
- [Segmenting Amazon DocumentDB collections and migrating in parallel \(p. 247\)](#)
- [Migrating multiple databases when using Amazon DocumentDB as a source for AWS DMS \(p. 251\)](#)
- [Limitations when using Amazon DocumentDB as a source for AWS DMS \(p. 252\)](#)
- [Using extra connections attributes with Amazon DocumentDB as a source \(p. 253\)](#)
- [Source data types for Amazon DocumentDB \(p. 253\)](#)

## Setting permissions to use Amazon DocumentDB as a source

When using Amazon DocumentDB source for an AWS DMS migration, you can create a user account with root privileges. Or you can create a user with permissions only for the database to be migrated.

The following code creates a user as the root account.

```
use admin
db.createUser(
{
  user: "root",
  pwd: "password",
  roles: [ { role: "root", db: "admin" } ]
})
```

For Amazon DocumentDB 3.6, the code following creates a user with minimal privileges on the database to be migrated.

```
use database_to_migrate
db.createUser(
{
  user: "dms-user",
  pwd: "password",
```

```
    roles: [ { role: "read", db: "db_name" }, "read"]
})
```

For Amazon DocumentDB 4.0 and above, AWS DMS uses a deployment-wide change stream. Here, the code following creates a user with minimal privileges.

```
db.createUser(
{
  user: "dms-user",
  pwd: "password",
  roles: [ { role: "readAnyDatabase", db: "admin" }]
})
```

## Configuring CDC for an Amazon DocumentDB cluster

To use ongoing replication or CDC with Amazon DocumentDB, AWS DMS requires access to the Amazon DocumentDB cluster's change streams. For a description of the time-ordered sequence of update events in your cluster's collections and databases, see [Using change streams](#) in the *Amazon DocumentDB Developer Guide*.

Authenticate to your Amazon DocumentDB cluster using the MongoDB shell. Then run the following command to enable change streams.

```
db.adminCommand({modifyChangeStreams: 1,
  database: "DB_NAME",
  collection: "",
  enable: true});
```

This approach enables the change stream for all collections in your database. After change streams are enabled, you can create a migration task that migrates existing data and at the same time replicates ongoing changes. AWS DMS continues to capture and apply changes even after the bulk data is loaded. Eventually, the source and target databases synchronize, minimizing downtime for a migration.

## Connecting to Amazon DocumentDB using TLS

By default, a newly created Amazon DocumentDB cluster accepts secure connections only using Transport Layer Security (TLS). When TLS is enabled, every connection to Amazon DocumentDB requires a public key.

You can retrieve the public key for Amazon DocumentDB by downloading the file `rds-combined-ca-bundle.pem` from an AWS-hosted Amazon S3 bucket. For more information on downloading this file, see [Encrypting connections using TLS](#) in the *Amazon DocumentDB Developer Guide*.

After you download the `rds-combined-ca-bundle.pem` file, you can import the public key that it contains into AWS DMS. The following steps describe how to do so.

### To import your public key using the AWS DMS console

1. Sign in to the AWS Management Console and choose AWS DMS.
2. In the navigation pane, choose **Certificates**.
3. Choose **Import certificate**. The **Import new CA certificate** page appears.
4. In the **Certificate configuration** section, do one of the following:

- For **Certificate identifier**, enter a unique name for the certificate, such as docdb-cert.
  - Choose **Choose file**, navigate to the location where you saved the rds-combined-ca-bundle.pem file, and select it.
5. Choose **Add new CA certificate**.

The AWS CLI following example uses the AWS DMS import-certificate command to import the public key rds-combined-ca-bundle.pem file.

```
aws dms import-certificate \
    --certificate-identifier docdb-cert \
    --certificate-pem file://./rds-combined-ca-bundle.pem
```

## Creating an Amazon DocumentDB source endpoint

You can create an Amazon DocumentDB source endpoint using either the console or AWS CLI. Use the procedure following with the console.

### To configure an Amazon DocumentDB source endpoint using the AWS DMS console

1. Sign in to the AWS Management Console and choose AWS DMS.
2. Choose **Endpoints** from the navigation pane, then choose **Create Endpoint**.
3. For **Endpoint identifier**, provide a name that helps you easily identify it, such as docdb-source.
4. For **Source engine**, choose **Amazon DocumentDB (with MongoDB compatibility)**.
5. For **Server name**, enter the name of the server where your Amazon DocumentDB database endpoint resides. For example, you might enter the public DNS name of your Amazon EC2 instance, such as democluster.cluster-cjf6q8nxfefi.us-east-2.docdb.amazonaws.com.
6. For **Port**, enter 27017.
7. For **SSL mode**, choose **verify-full**. If you have disabled SSL on your Amazon DocumentDB cluster, you can skip this step.
8. For **CA certificate**, choose the Amazon DocumentDB certificate, rds-combined-ca-bundle.pem. For instructions on adding this certificate, see [Connecting to Amazon DocumentDB using TLS \(p. 245\)](#).
9. For **Database name**, enter the name of the database to be migrated.

Use the following procedure with the CLI.

### To configure an Amazon DocumentDB source endpoint using the AWS CLI

- Run the following AWS DMS create-endpoint command to configure an Amazon DocumentDB source endpoint, replacing placeholders with your own values.

```
aws dms create-endpoint \
    --endpoint-identifier a_memorable_name \
    --endpoint-type source \
    --engine-name docdb \
    --username value \
    --password value \
    --server-name servername_where_database_endpoint_resides \
    --port 27017 \
    --database-name name_of_endpoint_database
```

## Segmenting Amazon DocumentDB collections and migrating in parallel

To improve performance of a migration task, Amazon DocumentDB source endpoints support two options of the parallel full load feature in table mapping. In other words, you can migrate a collection in parallel by using either the autosegmentation or the range segmentation options of table mapping for a parallel full load in JSON settings. The auto-segmenting options allow you to specify the criteria for AWS DMS to automatically segment your source for migration in each thread. The range segmentation options allow you to tell AWS DMS the specific range of each segment for DMS to migrate in each thread. For more information on these settings, see [Table and collection settings rules and operations \(p. 523\)](#).

### Migrating an Amazon DocumentDB database in parallel using autosegmentation ranges

You can migrate your documents in parallel by specifying the criteria for AWS DMS to automatically partition (segment) your data for each thread, especially the number of documents to migrate per thread. Using this approach, AWS DMS attempts to optimize segment boundaries for maximum performance per thread.

You can specify the segmentation criteria using the table-settings options following in table-mapping:

Table-settings option	Description
"type"	(Required) Set to "partitions-auto" for Amazon DocumentDB as a source.
"number-of-partitions"	(Optional) Total number of partitions (segments) used for migration. The default is 16.
"collection-count-from-metadata"	(Optional) If set to true, AWS DMS uses an estimated collection count for determining the number of partitions. If set to false, AWS DMS uses the actual collection count. The default is true.
"max-records-skip-per-page"	(Optional) The number of records to skip at once when determining the boundaries for each partition. AWS DMS uses a paginated skip approach to determine the minimum boundary for a partition. The default is 10000. Setting a relatively large value might result in cursor timeouts and task failures. Setting a relatively low value results in more operations per page and a slower full load.
"batch-size"	(Optional) Limits the number of documents returned in one batch. Each batch requires a round trip to the server. If the batch size is zero (0), the cursor uses the server-defined maximum batch size. The default is 0.

The example following shows a table mapping for autosegmentation.

```
{  
    "rules": [  
        {  
            "rule-type": "selection",  
            "rule-id": "1",  
            "rule-name": "1",  
            "object-locator": {  
                "schema-name": "admin",  
                "table-name": "departments"  
            },  
            "rule-action": "include",  
            "filters": []  
        },  
        {  
            "rule-type": "table-settings",  
            "rule-id": "2",  
            "rule-name": "2",  
            "object-locator": {  
                "schema-name": "admin",  
                "table-name": "departments"  
            },  
            "parallel-load": {  
                "type": "partitions-auto",  
                "number-of-partitions": 5,  
                "collection-count-from-metadata": "true",  
                "max-records-skip-per-page": 1000000,  
                "batch-size": 50000  
            }  
        }  
    ]  
}
```

Auto-segmentation has the limitation following. The migration for each segment fetches the collection count and the minimum `_id` for the collection separately. It then uses a paginated skip to calculate the minimum boundary for that segment. Therefore, ensure that the minimum `_id` value for each collection remains constant until all the segment boundaries in the collection are calculated. If you change the minimum `_id` value for a collection during its segment boundary calculation, this might cause data loss or duplicate row errors.

## Migrating an Amazon DocumentDB database in parallel using specific segment ranges

The following example shows an Amazon DocumentDB collection that has seven items, and `_id` as the primary key.

Key	Value
▼ (1) ObjectId("5f805c74873173399a278d78")	{ 3 fields }
└ _id	ObjectId("5f805c74873173399a278d78")
└ num	1
└ name	a
▼ (2) ObjectId("5f805c97873173399a278d79")	{ 3 fields }
└ _id	ObjectId("5f805c97873173399a278d79")
└ num	2
└ name	b
▼ (3) ObjectId("5f805cb0873173399a278d7a")	{ 3 fields }
└ _id	ObjectId("5f805cb0873173399a278d7a")
└ num	3
└ name	c
▼ (4) ObjectId("5f805cbb873173399a278d7b")	{ 3 fields }
└ _id	ObjectId("5f805cbb873173399a278d7b")
└ num	4
└ name	d
▼ (5) ObjectId("5f805cc5873173399a278d7c")	{ 3 fields }
└ _id	ObjectId("5f805cc5873173399a278d7c")
└ num	5
└ name	e
▼ (6) ObjectId("5f805cd0873173399a278d7d")	{ 3 fields }
└ _id	ObjectId("5f805cd0873173399a278d7d")
└ num	6
└ name	f
▼ (7) ObjectId("5f805cdd873173399a278d7e")	{ 3 fields }
└ _id	ObjectId("5f805cdd873173399a278d7e")
└ num	7
└ name	g

To split the collection into three segments and migrate in parallel, you can add table mapping rules to your migration task as shown in the following JSON example.

```
{
  // Task table mappings:
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "testdatabase",
        "table-name": "testtable"
      },
      "rule-action": "include"
    }
  ]
}
```

```

    }, // "selection" :"rule-type"
{
    "rule-type": "table-settings",
    "rule-id": "2",
    "rule-name": "2",
    "object-locator": {
        "schema-name": "testdatabase",
        "table-name": "testtable"
    },
    "parallel-load": {
        "type": "ranges",
        "columns": [
            "_id",
            "num"
        ],
        "boundaries": [
            // First segment selects documents with _id less-than-or-equal-to
            5f805c97873173399a278d79
            // and num less-than-or-equal-to 2.
            [
                "5f805c97873173399a278d79",
                "2"
            ],
            // Second segment selects documents with _id > 5f805c97873173399a278d79 and
            // _id less-than-or-equal-to 5f805cc5873173399a278d7c and
            // num > 2 and num less-than-or-equal-to 5.
            [
                "5f805cc5873173399a278d7c",
                "5"
            ]
            // Third segment is implied and selects documents with _id >
            5f805cc5873173399a278d7c.
        ] // :"boundaries"
    } // :"parallel-load"
} // "table-settings" :"rule-type"
] // :"rules"
} // :Task table mappings

```

That table mapping definition splits the source collection into three segments and migrates in parallel. The following are the segmentation boundaries.

```

Data with _id less-than-or-equal-to "5f805c97873173399a278d79" and num less-than-or-equal-
to 2 (2 records)
Data with _id less-than-or-equal-to "5f805cc5873173399a278d7c" and num less-than-or-equal-
to 5 and not in (_id less-than-or-equal-to "5f805c97873173399a278d79" and num less-than-
or-equal-to 2) (3 records)
Data not in (_id less-than-or-equal-to "5f805cc5873173399a278d7c" and num less-than-or-
equal-to 5) (2 records)

```

After the migration task is complete, you can verify from the task logs that the tables loaded in parallel, as shown in the following example. You can also verify the Amazon DocumentDB find clause used to unload each segment from the source table.

```

[TASK_MANAGER      ] I:  Start loading segment #1 of 3 of table
'testdatabase'.'testtable' (Id = 1) by subtask 1. Start load timestamp 0005B191D638FE86
(replicationtask_util.c:752)

[SOURCE_UNLOAD    ] I:  Range Segmentation filter for Segment #0 is initialized.
(mongodb_unload.c:157)

```

```
[SOURCE_UNLOAD    ] I: Range Segmentation filter for Segment #0 is: { "_id" : { "$lte" : { "$oid" : "5f805c97873173399a278d79" } }, "num" : { "$lte" : { "$numberInt" : "2" } } }  
(mongodb_unload.c:328)  
  
[SOURCE_UNLOAD    ] I: Unload finished for segment #1 of segmented table  
'testdatabase'.'testtable' (Id = 1). 2 rows sent.  
  
[TASK_MANAGER     ] I: Start loading segment #1 of 3 of table 'testdatabase'.'testtable' (Id  
= 1) by subtask 1. Start load timestamp 0005B191D638FE86 (replicationtask_util.c:752)  
  
[SOURCE_UNLOAD    ] I: Range Segmentation filter for Segment #0 is initialized.  
(mongodb_unload.c:157)  
  
[SOURCE_UNLOAD    ] I: Range Segmentation filter for Segment #0 is: { "_id" : { "$lte" : { "$oid" : "5f805c97873173399a278d79" } }, "num" : { "$lte" : { "$numberInt" : "2" } } }  
(mongodb_unload.c:328)  
  
[SOURCE_UNLOAD    ] I: Unload finished for segment #1 of segmented table  
'testdatabase'.'testtable' (Id = 1). 2 rows sent.  
  
[TARGET_LOAD      ] I: Load finished for segment #1 of segmented table  
'testdatabase'.'testtable' (Id = 1). 1 rows received. 0 rows skipped. Volume transferred  
480.  
  
[TASK_MANAGER     ] I: Load finished for segment #1 of table 'testdatabase'.'testtable' (Id  
= 1) by subtask 1. 2 records transferred.
```

Currently, AWS DMS supports the following Amazon DocumentDB data types as a segment key column:

- Double
- String
- ObjectId
- 32 bit integer
- 64 bit integer

## Migrating multiple databases when using Amazon DocumentDB as a source for AWS DMS

AWS DMS versions 3.4.5 and later support migrating multiple databases in a single task only for Amazon DocumentDB versions 4.0 and above. If you want to migrate multiple databases, do the following:

1. When you create the Amazon DocumentDB source endpoint:
  - In the AWS Management Console for AWS DMS, leave **Database name** empty under **Endpoint configuration** on the **Create endpoint** page.
  - In the AWS Command Line Interface (AWS CLI), assign an empty string value to the **DatabaseName** parameter in **DocumentDBSettings** that you specify for the **CreateEndpoint** action.
2. For each database that you want to migrate from this Amazon DocumentDB source endpoint, specify the name of each database as the name of a schema in the table-mapping for the task using either the guided input in the console or directly in JSON. For more information on the guided input, see the description of the [Specifying table selection and transformations rules from the console \(p. 491\)](#). For more information on the JSON, see [Selection rules and actions \(p. 495\)](#).

For example, you might specify the JSON following to migrate three Amazon DocumentDB databases.

### Example Migrate all tables in a schema

The JSON following migrates all tables from the Customers, Orders, and Suppliers databases in your source endpoint to your target endpoint.

```
{  
    "rules": [  
        {  
            "rule-type": "selection",  
            "rule-id": "1",  
            "rule-name": "1",  
            "object-locator": {  
                "schema-name": "Customers",  
                "table-name": "%"  
            },  
            "object-locator": {  
                "schema-name": "Orders",  
                "table-name": "%"  
            },  
            "object-locator": {  
                "schema-name": "Inventory",  
                "table-name": "%"  
            },  
            "rule-action": "include"  
        }  
    ]  
}
```

## Limitations when using Amazon DocumentDB as a source for AWS DMS

The following are limitations when using Amazon DocumentDB as a source for AWS DMS:

- When the `_id` option is set as a separate column, the ID string can't exceed 200 characters.
- Object ID and array type keys are converted to columns that are prefixed with `oid` and `array` in table mode.

Internally, these columns are referenced with the prefixed names. If you use transformation rules in AWS DMS that reference these columns, make sure to specify the prefixed column. For example, specify  `${oid__id}`  and not  `${_id}` , or  `${array__addresses}`  and not  `${_addresses}` .

- Collection names and key names can't include the dollar symbol (\$).
- Table mode and document mode have the limitations discussed preceding.
- Migrating in parallel using autosegmentation has the limitations described preceding.
- An Amazon DocumentDB (MongoDB compatible) source doesn't support using a specific timestamp as a start position for change data capture (CDC). An ongoing replication task starts capturing changes regardless of the timestamp.
- When using DocumentDB (MongoDB compatible) as a source, DMS can handle a maximum of 250 records per second.
- AWS DMS doesn't support documents where the nesting level is greater than 97.
- Source filters aren't supported for DocumentDB.

## Using extra connections attributes with Amazon DocumentDB as a source

When you set up your Amazon DocumentDB source endpoint, you can specify extra connection attributes. Extra connection attributes are specified by key-value pairs. If you have multiple connection attribute settings, separate them from each other by semicolons with no additional white space (for example, oneSetting;thenAnother).

The following table describes the extra connection attributes available when using Amazon DocumentDB databases as an AWS DMS source.

Attribute name	Valid values	Default value and description
nestingLevel	"none" "one"	"none" – Specify "none" to use document mode. Specify "one" to use table mode.
extractDocID	"true" "false"	"false" – Use this attribute when nestingLevel is set to "none".  If your target database is Amazon DocumentDB, set extractDocID=true.
docsToInvestigate	A positive integer greater than 0.	1000 – Use this attribute when nestingLevel is set to "one".

### Note

If the target endpoint is Amazon DocumentDB, run the migration in **Document mode**, and set the extra connection attribute extractDocID=true. To set the extra connection attribute extractDocID=true, modify your source endpoint and check the box **\_id as separate column**.

## Source data types for Amazon DocumentDB

In the following table, you can find the Amazon DocumentDB source data types that are supported when using AWS DMS. You can also find the default mapping from AWS DMS data types in this table. For more information about data types, see [BSON types](#) in the MongoDB documentation.

For information on how to view the data type that is mapped in the target, see the section for the target endpoint that you are using.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

Amazon DocumentDB data types	AWS DMS data types
Boolean	Bool
Binary	BLOB
Date	Date
Timestamp	Date
Int	INT4

Amazon DocumentDB data types	AWS DMS data types
Long	INT8
Double	REAL8
String (UTF-8)	CLOB
Array	CLOB
OID	String

## Using Amazon S3 as a source for AWS DMS

You can migrate data from an Amazon S3 bucket using AWS DMS. To do this, provide access to an Amazon S3 bucket containing one or more data files. In that S3 bucket, include a JSON file that describes the mapping between the data and the database tables of the data in those files.

The source data files must be present in the Amazon S3 bucket before the full load starts. You specify the bucket name using the `bucketName` parameter.

The source data files must be in comma-separated value (.csv) format. Name them using the following naming convention. In this convention, `schemaName` is the source schema and `tableName` is the name of a table within that schema.

```
/schemaName(tableName)/LOAD001.csv
/schemaName(tableName)/LOAD002.csv
/schemaName(tableName)/LOAD003.csv
...
```

For example, suppose that your data files are in `mybucket`, at the following Amazon S3 path.

```
s3://mybucket/hr/employee
```

At load time, AWS DMS assumes that the source schema name is `hr`, and that the source table name is `employee`.

In addition to `bucketName` (which is required), you can optionally provide a `bucketFolder` parameter to specify where AWS DMS should look for data files in the Amazon S3 bucket. Continuing the previous example, if you set `bucketFolder` to `sourcedata`, then AWS DMS reads the data files at the following path.

```
s3://mybucket/sourcedata/hr/employee
```

You can specify the column delimiter, row delimiter, null value indicator, and other parameters using extra connection attributes. For more information, see [Extra connection attributes for Amazon S3 as a source for AWS DMS \(p. 259\)](#).

You can specify a bucket owner and prevent sniping by using the `ExpectedBucketOwner` Amazon S3 endpoint setting, as shown following. Then, when you make a request to test a connection or perform a migration, S3 checks the account ID of the bucket owner against the specified parameter.

```
--s3-settings='{"ExpectedBucketOwner": "AWS_Account_ID"}'
```

### Topics

- [Defining external tables for Amazon S3 as a source for AWS DMS \(p. 255\)](#)
- [Using CDC with Amazon S3 as a source for AWS DMS \(p. 257\)](#)
- [Prerequisites when using Amazon S3 as a source for AWS DMS \(p. 258\)](#)
- [Limitations when using Amazon S3 as a source for AWS DMS \(p. 259\)](#)
- [Extra connection attributes for Amazon S3 as a source for AWS DMS \(p. 259\)](#)
- [Source data types for Amazon S3 \(p. 260\)](#)

## Defining external tables for Amazon S3 as a source for AWS DMS

In addition to the data files, you must also provide an external table definition. An *external table definition* is a JSON document that describes how AWS DMS should interpret the data from Amazon S3. The maximum size of this document is 2 MB. If you create a source endpoint using the AWS DMS Management Console, you can enter the JSON directly into the table-mapping box. If you use the AWS Command Line Interface (AWS CLI) or AWS DMS API to perform migrations, you can create a JSON file to specify the external table definition.

Suppose that you have a data file that includes the following.

```
101,Smith,Bob,2014-06-04,New York
102,Smith,Bob,2015-10-08,Los Angeles
103,Smith,Bob,2017-03-13,Dallas
104,Smith,Bob,2017-03-13,Dallas
```

Following is an example external table definition for this data.

```
{
    "TableCount": "1",
    "Tables": [
        {
            "TableName": "employee",
            "TablePath": "hr/employee/",
            "TableOwner": "hr",
            "TableColumns": [
                {
                    "ColumnName": "Id",
                    "ColumnType": "INT8",
                    "ColumnNullable": "false",
                    "ColumnIsPk": "true"
                },
                {
                    "ColumnName": "LastName",
                    "ColumnType": "STRING",
                    "ColumnLength": "20"
                },
                {
                    "ColumnName": "FirstName",
                    "ColumnType": "STRING",
                    "ColumnLength": "30"
                },
                {
                    "ColumnName": "HireDate",
                    "ColumnType": "DATETIME"
                },
                {
                    "ColumnName": "OfficeLocation",
                    "ColumnType": "STRING",
                    "ColumnLength": "20"
                }
            ]
        }
    ]
}
```

```
        }
    ],
    "TableColumnsTotal": "5"
}
}
```

The elements in this JSON document are as follows:

**TableCount** – the number of source tables. In this example, there is only one table.

**Tables** – an array consisting of one JSON map per source table. In this example, there is only one map. Each map consists of the following elements:

- **TableName** – the name of the source table.
- **TablePath** – the path in your Amazon S3 bucket where AWS DMS can find the full data load file. If a **bucketFolder** value is specified, its value is prepended to the path.
- **TableOwner** – the schema name for this table.
- **TableColumns** – an array of one or more maps, each of which describes a column in the source table:
  - **ColumnName** – the name of a column in the source table.
  - **ColumnType** – the data type for the column. For valid data types, see [Source data types for Amazon S3 \(p. 260\)](#).
  - **ColumnLength** – the number of bytes in this column. Maximum column length is limited to 2147483647 Bytes (2,047 MegaBytes) since an S3 source doesn't support FULL LOB mode. **ColumnLength** is valid for the following data types:
    - **BYTE**
    - **STRING**
  - **ColumnNullable** – a Boolean value that is **true** if this column can contain NULL values (**default=false**).
  - **ColumnIsPk** – a Boolean value that is **true** if this column is part of the primary key (**default=false**).
  - **ColumnDateFormat** – the input date format for a column with DATE, TIME, and DATETIME types, and used to parse a data string into a date object. Possible values include:

```
- YYYY-MM-dd HH:mm:ss
- YYYY-MM-dd HH:mm:ss.F
- YYYY/MM/dd HH:mm:ss
- YYYY/MM/dd HH:mm:ss.F
- MM/dd/YYYY HH:mm:ss
- MM/dd/YYYY HH:mm:ss.F
- YYYYMMdd HH:mm:ss
- YYYYMMdd HH:mm:ss.F
```

- **TableColumnsTotal** – the total number of columns. This number must match the number of elements in the **TableColumns** array.

If you don't specify otherwise, AWS DMS assumes that **ColumnLength** is zero.

**Note**

In supported versions of AWS DMS, the S3 source data can also contain an optional operation column as the first column before the **TableName** column value. This operation column identifies the operation (INSERT) used to migrate the data to an S3 target endpoint during a full load.

If present, the value of this column is the initial character of the INSERT operation keyword (I). If specified, this column generally indicates that the S3 source was created by DMS as an S3 target during a previous migration.

In DMS versions prior to 3.4.2, this column wasn't present in S3 source data created from a previous DMS full load. Adding this column to S3 target data allows the format of all rows written to the S3 target to be consistent whether they are written during a full load or during a CDC load. For more information on the options for formatting S3 target data, see [Indicating source DB operations in migrated S3 data \(p. 359\)](#).

For a column of the NUMERIC type, specify the precision and scale. *Precision* is the total number of digits in a number, and *scale* is the number of digits to the right of the decimal point. You use the ColumnPrecision and ColumnScale elements for this, as shown following.

```
...
{
    "ColumnName": "HourlyRate",
    "ColumnType": "NUMERIC",
    "ColumnPrecision": "5"
    "ColumnScale": "2"
}
...
```

For a column of the DATETIME type with data that contains fractional seconds, specify the scale. *Scale* is the number of digits for the fractional seconds, and can range from 0 to 9. You use the ColumnScale element for this, as shown following.

```
...
{
    "ColumnName": "HireDate",
    "ColumnType": "DATETIME",
    "ColumnScale": "3"
}
...
```

If you don't specify otherwise, AWS DMS assumes ColumnScale is zero and truncates the fractional seconds.

## Using CDC with Amazon S3 as a source for AWS DMS

After AWS DMS performs a full data load, it can optionally replicate data changes to the target endpoint. To do this, you upload change data capture files (CDC files) to your Amazon S3 bucket. AWS DMS reads these CDC files when you upload them, and then applies the changes at the target endpoint.

The CDC files are named as follows:

```
CDC00001.csv
CDC00002.csv
CDC00003.csv
...

```

### Note

To replicate CDC files in the change data folder successfully upload them in a lexical (sequential) order. For example, upload the file CDC00002.csv before the file CDC00003.csv. Otherwise, CDC00002.csv is skipped and isn't replicated if you load it after CDC00003.csv. But the file CDC00004.csv replicates successfully if loaded after CDC00003.csv.

To indicate where AWS DMS can find the files, specify the cdcPath parameter. Continuing the previous example, if you set cdcPath to *changedata*, then AWS DMS reads the CDC files at the following path.

```
s3://mybucket/changedata
```

If you set cdcPath to *changedata* and bucketFolder to *myFolder*, then AWS DMS reads the CDC files at the following path.

```
s3://mybucket/myFolder/changedata
```

The records in a CDC file are formatted as follows:

- Operation – the change operation to be performed: INSERT or I, UPDATE or U, or DELETE or D. These keyword and character values are case-insensitive.

**Note**

In supported AWS DMS versions, AWS DMS can identify the operation to perform for each load record in two ways. AWS DMS can do this from the record's keyword value (for example, INSERT) or from its keyword initial character (for example, I). In prior versions, AWS DMS recognized the load operation only from the full keyword value.

In prior versions of AWS DMS, the full keyword value was written to log the CDC data. Also, prior versions wrote the operation value to any S3 target using only the keyword initial.

Recognizing both formats allows AWS DMS to handle the operation regardless of how the operation column is written to create the S3 source data. This approach supports using S3 target data as the source for a later migration. With this approach, you don't need to change the format of any keyword initial value that appears in the operation column of the later S3 source.

- Table name – the name of the source table.
- Schema name – the name of the source schema.
- Data – one or more columns that represent the data to be changed.

Following is an example CDC file for a table named employee.

```
INSERT,employee,hr,101,Smith,Bob,2014-06-04,New York
UPDATE,employee,hr,101,Smith,Bob,2015-10-08,Los Angeles
UPDATE,employee,hr,101,Smith,Bob,2017-03-13,Dallas
DELETE,employee,hr,101,Smith,Bob,2017-03-13,Dallas
```

## Prerequisites when using Amazon S3 as a source for AWS DMS

To use Amazon S3 as a source for AWS DMS, your source S3 bucket must be in the same AWS Region as the DMS replication instance that migrates your data. In addition, the AWS account you use for the migration must have read access to the source bucket.

The AWS Identity and Access Management (IAM) role assigned to the user account used to create the migration task must have the following set of permissions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::mybucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3>ListBucket"
            ]
        }
    ]
}
```

```

        ],
        "Resource": [
            "arn:aws:s3:::mybucket*"
        ]
    }
}

```

The AWS Identity and Access Management (IAM) role assigned to the user account used to create the migration task must have the following set of permissions if versioning is enabled on the Amazon S3 bucket.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "S3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::mybucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3>ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::mybucket*"
            ]
        }
    ]
}

```

## Limitations when using Amazon S3 as a source for AWS DMS

The following limitations apply when using Amazon S3 as a source:

- Don't enable versioning for S3. If you need S3 versioning, use lifecycle policies to actively delete old versions. Otherwise, you might encounter endpoint test connection failures because of an S3 list-object call timeout. To create a lifecycle policy for an S3 bucket, see [Managing your storage lifecycle](#). To delete a version of an S3 object, see [Deleting object versions from a versioning-enabled bucket](#).
- A VPC-enabled (gateway VPC) S3 bucket is supported in versions 3.4.7 and higher.

## Extra connection attributes for Amazon S3 as a source for AWS DMS

You can specify the following options as extra connection attributes.

Option	Description
bucketFolder	(Optional) A folder name in the S3 bucket. If this attribute is provided, source data files and CDC files are read from the path <code>s3://myBucket/bucketFolder/schemaName/tableName/</code> and

Option	Description
	s3:// <i>myBucket/bucketFolder</i> / respectively. If this attribute isn't specified, then the path used is <i>schemaName/tableName</i> . An example follows.  bucketFolder=sourceData;
bucketName	The name of the S3 bucket. An example follows.  bucketName=myBucket;
cdcPath	The location of CDC files. This attribute is required if a task captures change data; otherwise, it's optional. If cdcPath is present, then AWS DMS reads CDC files from this path and replicates the data changes to the target endpoint. For more information, see <a href="#">Using CDC with Amazon S3 as a source for AWS DMS (p. 257)</a> . An example follows.  cdcPath=changeData;
csvDelimiter	The delimiter used to separate columns in the source files. The default is a comma. An example follows.  csvDelimiter=,;
csvRowDelimiter	The delimiter used to separate rows in the source files. The default is a newline (\n). An example follows.  csvRowDelimiter=\n;
ignoreHeaderRows	When this value is set to 1, AWS DMS ignores the first row header in a .csv file. A value of 1 enables the feature, a value of 0 disables the feature.  The default is 0.  Example:  ignoreHeaderRows=1;
rfc4180	When this value is set to true or y, each leading double quotation mark has to be followed by an ending double quotation mark. This formatting complies with RFC 4180. When this value is set to false or n, string literals are copied to the target as is. In this case, a delimiter (row or column) signals the end of the field. Thus, you can't use a delimiter as part of the string, because it signals the end of the value.  The default is true.  Valid values: true, false, y, n  Example:  rfc4180=false;

## Source data types for Amazon S3

Data migration that uses Amazon S3 as a source for AWS DMS needs to map data from Amazon S3 to AWS DMS data types. For more information, see [Defining external tables for Amazon S3 as a source for AWS DMS \(p. 255\)](#).

For information on how to view the data type that is mapped in the target, see the section for the target endpoint you are using.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

The following AWS DMS data types are used with Amazon S3 as a source:

- BYTE – Requires ColumnLength. For more information, see [Defining external tables for Amazon S3 as a source for AWS DMS \(p. 255\)](#).
- DATE
- TIME
- DATETIME – For more information and an example, see the DATETIME type example in [Defining external tables for Amazon S3 as a source for AWS DMS \(p. 255\)](#).
- INT1
- INT2
- INT4
- INT8
- NUMERIC – Requires ColumnPrecision and ColumnScale. For more information and an example, see the NUMERIC type example in [Defining external tables for Amazon S3 as a source for AWS DMS \(p. 255\)](#).
- REAL4
- REAL8
- STRING – Requires ColumnLength. For more information, see [Defining external tables for Amazon S3 as a source for AWS DMS \(p. 255\)](#).
- UINT1
- UINT2
- UINT4
- UINT8
- BLOB
- CLOB
- BOOLEAN

## Using IBM Db2 for Linux, Unix, and Windows database (Db2 LUW) as a source for AWS DMS

You can migrate data from an IBM Db2 for Linux, Unix, and Windows (Db2 LUW) database to any supported target database using AWS Database Migration Service (AWS DMS). AWS DMS supports the following versions of Db2 LUW as migration sources:

- Version 9.7, with all Fix Packs supported
- Version 10.1, with all Fix Packs supported
- Version 10.5, with all Fix Packs supported except for Fix Pack 5
- Version 11.1, with all Fix Packs supported
- Version 11.5, Mods (0-7) with only Fix Pack 0 supported

You can use Secure Sockets Layer (SSL) to encrypt connections between your Db2 LUW endpoint and the replication instance. For more information on using SSL with a Db2 LUW endpoint, see [Using SSL with AWS Database Migration Service \(p. 647\)](#).

## Prerequisites when using Db2 LUW as a source for AWS DMS

The following prerequisites are required before you can use an Db2 LUW database as a source.

To enable ongoing replication, also called change data capture (CDC), do the following:

- Set the database to be recoverable, which AWS DMS requires to capture changes. A database is recoverable if either or both of the database configuration parameters LOGARCHMETH1 and LOGARCHMETH2 are set to ON.

If your database is recoverable, then AWS DMS can access the Db2 ARCHIVE LOG if needed.
- Ensure that the DB2 transaction logs are available, with a sufficient retention period to be processed by AWS DMS.
- DB2 requires SYSADM or DBADM authorization to extract transaction log records. Grant the user account the following permissions:
  - SYSADM or DBADM
  - DATAACCESS
- When using IBM DB2 for LUW version 9.7 as a source, set the extra connection attribute (ECA), CurrentLSN as follows:

CurrentLSN=*LSN* where *LSN* specifies a log sequence number (LSN) where you want the replication to start. Or, CurrentLSN=*scan*.

## Limitations when using Db2 LUW as a source for AWS DMS

AWS DMS doesn't support clustered databases. However, you can define a separate Db2 LUW for each of the endpoints of a cluster.

When using ongoing replication (CDC), the following limitations apply:

- When a table with multiple partitions is truncated, the number of DDL events shown in the AWS DMS console is equal to the number of partitions. This is because Db2 LUW records a separate DDL for each partition.
- The following DDL actions aren't supported on partitioned tables:
  - ALTER TABLE ADD PARTITION
  - ALTER TABLE DETACH PARTITION
  - ALTER TABLE ATTACH PARTITION
- AWS DMS doesn't support an ongoing replication migration from a DB2 high availability disaster recovery (HADR) standby instance. The standby is inaccessible.
- The DECFLOAT data type isn't supported. Consequently, changes to DECFLOAT columns are ignored during ongoing replication.
- The RENAME COLUMN statement isn't supported.
- When performing updates to Multi-Dimensional Clustering (MDC) tables, each update is shown in the AWS DMS console as INSERT + DELETE.
- When the task setting **Include LOB columns in replication** isn't enabled, any table that has LOB columns is suspended during ongoing replication.
- For Db2 LUW versions 10.5 and higher, variable-length string columns with data that is stored out-of-row are ignored. This limitation only applies to tables created with extended row size for columns with data types like VARCHAR and VARGRAPHIC. To work around this limitation, move the table to a table space with a higher page size. For more information, see [What can I do if I want to change the pagesize of DB2 tablespaces](#).

- For ongoing replication, DMS doesn't support migrating data loaded at the page level by the DB2 LOAD utility. Instead, use the IMPORT utility which uses SQL inserts. For more information, see [differences between the import and load utilities](#).
- While a replication task is running, DMS captures CREATE TABLE DDLs only if the tables were created with the DATA CAPTURE CHANGE attribute.
- DMS doesn't support DB2 Database Partition Feature (DPF).
- For ongoing replication (CDC), if you plan to start replication from a specific timestamp, you must set the StartFromContext connection attribute to the required timestamp.

## Extra connection attributes when using Db2 LUW as a source for AWS DMS

You can use extra connection attributes to configure your Db2 LUW source. You specify these settings when you create the source endpoint. If you have multiple connection attribute settings, separate them from each other by semicolons with no additional white space (for example, oneSetting;thenAnother).

The following table shows the extra connection attributes that you can use with Db2 LUW as a source.

Name	Description
CurrentLSN	For ongoing replication (CDC), use CurrentLSN to specify a log sequence number (LSN) where you want the replication to start.
MaxKBytesPerRead	Maximum number of bytes per read, as a NUMBER value. The default is 64 KB.
SetDataCaptureChanges	Enables ongoing replication (CDC) as a BOOLEAN value. The default is true.
StartFromContext	<p>For ongoing replication (CDC), use StartFromContext to specify a log's lower limit from where to start the replication. StartFromContext accepts different forms of values. Valid values include:</p> <ul style="list-style-type: none"> <li>timestamp (UTC). For example:</li> </ul> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <pre>StartFromContext=timestamp:2021-09-21T13:00:00</pre> </div> <ul style="list-style-type: none"> <li>NOW</li> </ul> <p>For IBM DB2 LUW version 10.5 and later, NOW combined with CurrentLSN=scan, starts the task from the latest LSO. For example:</p> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <pre>CurrentLSN=scan;StartFromContext=NOW</pre> </div> <ul style="list-style-type: none"> <li>A specific LRI. For example:</li> </ul> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <pre>StartFromContext=010000000000022CC000000000004FB13</pre> </div>

Name	Description
	<p>To determine the LRI/LSN range of a log file, run the db2f1sn command as shown in the example following.</p> <pre>db2f1sn -db <i>SAMPLE</i> -lrirange 2</pre> <p>The output from that example is similar to the following.</p> <pre>S0000002.LOG: has LRI range 000000000000000010000000000002254000000000004F9A6 to 0000000000000000100000000000022CC000000000004FB13</pre> <p>In that output, the log file is S0000002.LOG and the <b>StartFromContext</b> LRI value is the 34 bytes at the end of the range.</p> <pre>0100000000000022CC000000000004FB13</pre>

## Source data types for IBM Db2 LUW

Data migration that uses Db2 LUW as a source for AWS DMS supports most Db2 LUW data types. The following table shows the Db2 LUW source data types that are supported when using AWS DMS and the default mapping from AWS DMS data types. For more information about Db2 LUW data types, see the [Db2 LUW documentation](#).

For information on how to view the data type that is mapped in the target, see the section for the target endpoint that you're using.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

Db2 LUW data types	AWS DMS data types
INTEGER	INT4
SMALLINT	INT2
BIGINT	INT8
DECIMAL (p,s)	NUMERIC (p,s)
FLOAT	REAL8
DOUBLE	REAL8
REAL	REAL4
DECFLOAT (p)	If precision is 16, then REAL8; if precision is 34, then STRING
GRAPHIC (n)	WSTRING, for fixed-length graphic strings of double byte chars with a length greater than 0 and less than or equal to 127

Db2 LUW data types	AWS DMS data types
VARGRAPHIC (n)	WSTRING, for varying-length graphic strings with a length greater than 0 and less than or equal to 16,352 double byte chars
LONG VARGRAPHIC (n)	CLOB, for varying-length graphic strings with a length greater than 0 and less than or equal to 16,352 double byte chars
CHARACTER (n)	STRING, for fixed-length strings of double byte chars with a length greater than 0 and less than or equal to 255
VARCHAR (n)	STRING, for varying-length strings of double byte chars with a length greater than 0 and less than or equal to 32,704
LONG VARCHAR (n)	CLOB, for varying-length strings of double byte chars with a length greater than 0 and less than or equal to 32,704
CHAR (n) FOR BIT DATA	BYTES
VARCHAR (n) FOR BIT DATA	BYTES
LONG VARCHAR FOR BIT DATA	BYTES
DATE	DATE
TIME	TIME
TIMESTAMP	DATETIME
BLOB (n)	BLOB  Maximum length is 2,147,483,647 bytes
CLOB (n)	CLOB  Maximum length is 2,147,483,647 bytes
DBCLOB (n)	CLOB  Maximum length is 1,073,741,824 double byte chars
XML	CLOB

## Using IBM Db2 for z/OS databases as a source for AWS DMS

You can migrate data from an IBM for z/OS database to any supported target database using AWS Database Migration Service (AWS DMS). AWS DMS supports Db2 for z/OS version 12 as a source for data migration.

## Prerequisites when using Db2 for z/OS as a source for AWS DMS

To use an IBM Db2 for z/OS database as a source in AWS DMS, grant the following privileges to the Db2 for z/OS user specified in the source endpoint connection settings.

```
GRANT SELECT ON SYSIBM.SYSTABLES TO Db2USER;
GRANT SELECT ON SYSIBM.SYSTABLESPACE TO Db2USER;
GRANT SELECT ON SYSIBM.SYSTABLEPART TO Db2USER;
GRANT SELECT ON SYSIBM.SYSCOLUMNS TO Db2USER;
GRANT SELECT ON SYSIBM.SYSDATABASE TO Db2USER;
GRANT SELECT ON SYSIBM.SYSDUMMY1 TO Db2USER
```

Also grant SELECT ON *user defined* source tables.

An AWS DMS IBM Db2 for z/OS source endpoint relies on the IBM Data Server Driver for ODBC to access data. The database server must have a valid IBM ODBC Connect license for DMS to connect to this endpoint.

## Limitations when using Db2 for z/OS as a source for AWS DMS

The following limitations apply when using an IBM Db2 for z/OS database as a source for AWS DMS:

- Only Full Load replication tasks are supported. Change data capture (CDC) isn't supported.
- Parallel load isn't supported.
- Data validation of views are not supported.
- Schema, table, and columns names must be specified in UPPER case in table mappings for Column/table level transformations and row level selection filters.

## Source data types for IBM Db2 for z/OS

Data migrations that use Db2 for z/OS as a source for AWS DMS support most Db2 for z/OS data types. The following table shows the Db2 for z/OS source data types that are supported when using AWS DMS, and the default mapping from AWS DMS data types.

For more information about Db2 for z/OS data types, see the [IBM Db2 for z/OS documentation](#).

For information on how to view the data type that is mapped in the target, see the section for the target endpoint that you're using.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

Db2 for z/OS data types	AWS DMS data types
INTEGER	INT4
SMALLINT	INT2
BIGINT	INT8
DECIMAL (p,s)	NUMERIC (p,s)  If a decimal point is set to a comma (,) in the DB2 configuration, configure Replicate to support the DB2 setting.

Db2 for z/OS data types	AWS DMS data types
FLOAT	REAL8
DOUBLE	REAL8
REAL	REAL4
DECFLOAT (p)	If precision is 16, then REAL8; if precision is 34, then STRING
GRAPHIC (n)	If n>=127 then WSTRING, for fixed-length graphic strings of double byte chars with a length greater than 0 and less than or equal to 127
VARGRAPHIC (n)	WSTRING, for varying-length graphic strings with a length greater than 0 and less than or equal to 16,352 double byte chars
LONG VARGRAPHIC (n)	CLOB, for varying-length graphic strings with a length greater than 0 and less than or equal to 16,352 double byte chars
CHARACTER (n)	STRING, for fixed-length strings of double byte chars with a length greater than 0 and less than or equal to 255
VARCHAR (n)	STRING, for varying-length strings of double byte chars with a length greater than 0 and less than or equal to 32,704
LONG VARCHAR (n)	CLOB, for varying-length strings of double byte chars with a length greater than 0 and less than or equal to 32,704
CHAR (n) FOR BIT DATA	BYTES
VARCHAR (n) FOR BIT DATA	BYTES
LONG VARCHAR FOR BIT DATA	BYTES
DATE	DATE
TIME	TIME
TIMESTAMP	DATETIME
BLOB (n)	BLOB Maximum length is 2,147,483,647 bytes
CLOB (n)	CLOB Maximum length is 2,147,483,647 bytes
DBCLOB (n)	CLOB Maximum length is 1,073,741,824 double byte chars
XML	CLOB

Db2 for z/OS data types	AWS DMS data types
BINARY	BYTES
VARBINARY	BYTES
ROWID	BYTES. For more information about working with ROWID, see following.
TIMESTAMP WITH TIME ZONE	Not supported.

ROWID columns are migrated by default when the target table prep mode for the task is set to DROP\_AND\_CREATE (the default). Data validation ignores these columns because the rows are meaningless outside the specific database and table. To turn off migration of these columns, you can do one of the following preparatory steps:

- Precreate the target table without these columns. Then, set the target table prep mode of the task to either DO\_NOTHING or TRUNCATE\_BEFORE\_LOAD. You can use AWS Schema Conversion Tool (AWS SCT) to precreate the target table without the columns.
- Add a table mapping rule to a task that filters out these columns so that they're ignored. For more information, see [Transformation rules and actions \(p. 500\)](#).

## EBCDIC collations in PostgreSQL for AWS Mainframe Modernization service

AWS Mainframe Modernization program helps you modernize your mainframe applications to AWS managed runtime environments. It provides tools and resources that help you plan and implement your migration and modernization projects. For more information about mainframe modernization and migration, see [Mainframe Modernization with AWS](#).

If you currently use either the Micro Focus or BluAge component of the Mainframe Modernization service, you must tell AWS DMS to *shift* (translate) certain code points. You can use AWS DMS task settings to perform the shifts. The following example shows how to use the AWS DMS CharacterSetSettings operation to map the shifts in a DMS task setting.

```
"CharacterSetSettings": {
    "CharacterSetSupport": null,
    "CharacterReplacements": [
        {"SourceCharacterCodePoint": "0000", "TargetCharacterCodePoint": "0180"},
        {"SourceCharacterCodePoint": "00B8", "TargetCharacterCodePoint": "0160"},
        {"SourceCharacterCodePoint": "00BC", "TargetCharacterCodePoint": "0161"},
        {"SourceCharacterCodePoint": "00BD", "TargetCharacterCodePoint": "017D"},
        {"SourceCharacterCodePoint": "00BE", "TargetCharacterCodePoint": "017E"},
        {"SourceCharacterCodePoint": "00A8", "TargetCharacterCodePoint": "0152"},
        {"SourceCharacterCodePoint": "00B4", "TargetCharacterCodePoint": "0153"},
        {"SourceCharacterCodePoint": "00A6", "TargetCharacterCodePoint": "0178"}
    ]
}
```

Some EBCDIC collations already exist for PostgreSQL that understand the shifting that's needed. Several different code pages are supported. The sections following provide JSON samples of what you must shift for all the supported code pages. You can simply copy-and-paste the necessary JSON that you need in your DMS task.

## Micro Focus specific EBCDIC collations

For Micro Focus, shift a subset of characters as needed for the following collations.

```
da-DK-cp1142m-x-icu
de-DE-cp1141m-x-icu
en-GB-cp1146m-x-icu
en-US-cp1140m-x-icu
es-ES-cp1145m-x-icu
fi-FI-cp1143m-x-icu
fr-FR-cp1147m-x-icu
it-IT-cp1144m-x-icu
nl-BE-cp1148m-x-icu
```

### Example Micro Focus data shifts per collation:

#### en\_us\_cp1140m

Code Shift:

```
0000    0180
00A6    0160
00B8    0161
00BC    017D
00BD    017E
00BE    0152
00A8    0153
00B4    0178
```

Corresponding input mapping for an AWS DMS task:

```
{"SourceCharacterCodePoint": "0000", "TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "00A6", "TargetCharacterCodePoint": "0160"}
,{"SourceCharacterCodePoint": "00B8", "TargetCharacterCodePoint": "0161"}
,{"SourceCharacterCodePoint": "00BC", "TargetCharacterCodePoint": "017D"}
,{"SourceCharacterCodePoint": "00BD", "TargetCharacterCodePoint": "017E"}
,{"SourceCharacterCodePoint": "00BE", "TargetCharacterCodePoint": "0152"}
,{"SourceCharacterCodePoint": "00A8", "TargetCharacterCodePoint": "0153"}
,{"SourceCharacterCodePoint": "00B4", "TargetCharacterCodePoint": "0178"}
```

#### en\_us\_cp1141m

Code Shift:

```
0000    0180
00B8    0160
00BC    0161
00BD    017D
00BE    017E
00A8    0152
00B4    0153
00A6    0178
```

Corresponding input mapping for an AWS DMS task:

```
{"SourceCharacterCodePoint": "0000", "TargetCharacterCodePoint": "0180"}
,{"SourceCharacterCodePoint": "00B8", "TargetCharacterCodePoint": "0160"}
,{"SourceCharacterCodePoint": "00BC", "TargetCharacterCodePoint": "0161"}
,{"SourceCharacterCodePoint": "00BD", "TargetCharacterCodePoint": "017D"}
```

```
,{"SourceCharacterCodePoint": "00BE","TargetCharacterCodePoint": "017E"}  
,{"SourceCharacterCodePoint": "00A8","TargetCharacterCodePoint": "0152"}  
, {"SourceCharacterCodePoint": "00B4","TargetCharacterCodePoint": "0153"}  
, {"SourceCharacterCodePoint": "00A6","TargetCharacterCodePoint": "0178"}
```

### en\_us\_cp1142m

Code Shift:

0000	0180
00A6	0160
00B8	0161
00BC	017D
00BD	017E
00BE	0152
00A8	0153
00B4	0178

Corresponding input mapping for an AWS DMS task:

```
,{"SourceCharacterCodePoint": "0000","TargetCharacterCodePoint": "0180"}  
, {"SourceCharacterCodePoint": "00A6","TargetCharacterCodePoint": "0160"}  
, {"SourceCharacterCodePoint": "00B8","TargetCharacterCodePoint": "0161"}  
, {"SourceCharacterCodePoint": "00BC","TargetCharacterCodePoint": "017D"}  
, {"SourceCharacterCodePoint": "00BD","TargetCharacterCodePoint": "017E"}  
, {"SourceCharacterCodePoint": "00BE","TargetCharacterCodePoint": "0152"}  
, {"SourceCharacterCodePoint": "00A8","TargetCharacterCodePoint": "0153"}  
, {"SourceCharacterCodePoint": "00B4","TargetCharacterCodePoint": "0178"}
```

### en\_us\_cp1143m

Code Shift:

0000	0180
00B8	0160
00BC	0161
00BD	017D
00BE	017E
00A8	0152
00B4	0153
00A6	0178

Corresponding input mapping for an AWS DMS task:

```
, {"SourceCharacterCodePoint": "0000","TargetCharacterCodePoint": "0180"}  
, {"SourceCharacterCodePoint": "00B8","TargetCharacterCodePoint": "0160"}  
, {"SourceCharacterCodePoint": "00BC","TargetCharacterCodePoint": "0161"}  
, {"SourceCharacterCodePoint": "00BD","TargetCharacterCodePoint": "017D"}  
, {"SourceCharacterCodePoint": "00BE","TargetCharacterCodePoint": "017E"}  
, {"SourceCharacterCodePoint": "00A8","TargetCharacterCodePoint": "0152"}  
, {"SourceCharacterCodePoint": "00B4","TargetCharacterCodePoint": "0153"}  
, {"SourceCharacterCodePoint": "00A6","TargetCharacterCodePoint": "0178"}
```

### en\_us\_cp1144m

Code Shift:

0000	0180
00B8	0160
00BC	0161
00BD	017D
00BE	017E
00A8	0152
00B4	0153
00A6	0178

Corresponding input mapping for an AWS DMS task:

```
[{"SourceCharacterCodePoint": "0000", "TargetCharacterCodePoint": "0180"}, {"SourceCharacterCodePoint": "00B8", "TargetCharacterCodePoint": "0160"}, {"SourceCharacterCodePoint": "00BC", "TargetCharacterCodePoint": "0161"}, {"SourceCharacterCodePoint": "00BD", "TargetCharacterCodePoint": "017D"}, {"SourceCharacterCodePoint": "00BE", "TargetCharacterCodePoint": "017E"}, {"SourceCharacterCodePoint": "00A8", "TargetCharacterCodePoint": "0152"}, {"SourceCharacterCodePoint": "00B4", "TargetCharacterCodePoint": "0153"}, {"SourceCharacterCodePoint": "00A6", "TargetCharacterCodePoint": "0178"}]
```

### en\_us\_cp1145m

Code Shift:

0000	0180
00A6	0160
00B8	0161
00A8	017D
00BC	017E
00BD	0152
00BE	0153
00B4	0178

Corresponding input mapping for an AWS DMS task:

```
[{"SourceCharacterCodePoint": "0000", "TargetCharacterCodePoint": "0180"}, {"SourceCharacterCodePoint": "00A6", "TargetCharacterCodePoint": "0160"}, {"SourceCharacterCodePoint": "00B8", "TargetCharacterCodePoint": "0161"}, {"SourceCharacterCodePoint": "00A8", "TargetCharacterCodePoint": "017D"}, {"SourceCharacterCodePoint": "00BC", "TargetCharacterCodePoint": "017E"}, {"SourceCharacterCodePoint": "00BD", "TargetCharacterCodePoint": "0152"}, {"SourceCharacterCodePoint": "00BE", "TargetCharacterCodePoint": "0153"}, {"SourceCharacterCodePoint": "00B4", "TargetCharacterCodePoint": "0178"}]
```

### en\_us\_cp1146m

Code Shift:

0000	0180
00A6	0160
00B8	0161
00BC	017D
00BD	017E
00BE	0152
00A8	0153
00B4	0178

Corresponding input mapping for an AWS DMS task:

```
{"SourceCharacterCodePoint": "0000", "TargetCharacterCodePoint": "0180"}  
,{"SourceCharacterCodePoint": "00A6", "TargetCharacterCodePoint": "0160"}  
,{"SourceCharacterCodePoint": "00B8", "TargetCharacterCodePoint": "0161"}  
,{"SourceCharacterCodePoint": "00BC", "TargetCharacterCodePoint": "017D"}  
,{"SourceCharacterCodePoint": "00BD", "TargetCharacterCodePoint": "017E"}  
,{"SourceCharacterCodePoint": "00BE", "TargetCharacterCodePoint": "0152"}  
,{"SourceCharacterCodePoint": "00A8", "TargetCharacterCodePoint": "0153"}  
,{"SourceCharacterCodePoint": "00B4", "TargetCharacterCodePoint": "0178"}
```

### en\_us\_cp1147m

Code Shift:

0000	0180
00B8	0160
00A8	0161
00BC	017D
00BD	017E
00BE	0152
00A8	0153
00B4	0178
00A6	0178

Corresponding input mapping for an AWS DMS task:

```
{"SourceCharacterCodePoint": "0000", "TargetCharacterCodePoint": "0180"}  
,{"SourceCharacterCodePoint": "00B8", "TargetCharacterCodePoint": "0160"}  
,{"SourceCharacterCodePoint": "00A8", "TargetCharacterCodePoint": "0161"}  
,{"SourceCharacterCodePoint": "00BC", "TargetCharacterCodePoint": "017D"}  
,{"SourceCharacterCodePoint": "00BD", "TargetCharacterCodePoint": "017E"}  
,{"SourceCharacterCodePoint": "00BE", "TargetCharacterCodePoint": "0152"}  
,{"SourceCharacterCodePoint": "00B4", "TargetCharacterCodePoint": "0153"}  
,{"SourceCharacterCodePoint": "00A6", "TargetCharacterCodePoint": "0178"}
```

### en\_us\_cp1148m

Code Shift:

0000	0180
00A6	0160
00B8	0161
00BC	017D
00BD	017E
00BE	0152
00A8	0153
00B4	0178

Corresponding input mapping for an AWS DMS task:

```
{"SourceCharacterCodePoint": "0000", "TargetCharacterCodePoint": "0180"}  
,{"SourceCharacterCodePoint": "00A6", "TargetCharacterCodePoint": "0160"}  
,{"SourceCharacterCodePoint": "00B8", "TargetCharacterCodePoint": "0161"}  
,{"SourceCharacterCodePoint": "00BC", "TargetCharacterCodePoint": "017D"}  
,{"SourceCharacterCodePoint": "00BD", "TargetCharacterCodePoint": "017E"}  
,{"SourceCharacterCodePoint": "00BE", "TargetCharacterCodePoint": "0152"}  
,{"SourceCharacterCodePoint": "00A8", "TargetCharacterCodePoint": "0153"}
```

```
,{"SourceCharacterCodePoint": "00B4","TargetCharacterCodePoint": "0178"}]
```

## BluAge specific EBCDIC collations

For BluAge, shift all of the following *low values* and *high values* as needed. These collations should only be used to support the Mainframe Migration BluAge service.

```
da-DK-cp1142b-x-icu
da-DK-cp277b-x-icu
de-DE-cp1141b-x-icu
de-DE-cp273b-x-icu
en-GB-cp1146b-x-icu
en-GB-cp285b-x-icu
en-US-cp037b-x-icu
en-US-cp1140b-x-icu
es-ES-cp1145b-x-icu
es-ES-cp284b-x-icu
fi-FI-cp1143b-x-icu
fi-FI-cp278b-x-icu
fr-FR-cp1147b-x-icu
fr-FR-cp297b-x-icu
it-IT-cp1144b-x-icu
it-IT-cp280b-x-icu
nl-BE-cp1148b-x-icu
nl-BE-cp500b-x-icu
```

### Example BluAge Data Shifts:

**da-DK-cp277b** and **da-DK-cp1142b**

Code Shift:

0180	0180
0001	0181
0002	0182
0003	0183
009C	0184
0009	0185
0086	0186
007F	0187
0097	0188
008D	0189
008E	018A
000B	018B
000C	018C
000D	018D
000E	018E
000F	018F
0010	0190
0011	0191
0012	0192
0013	0193
009D	0194
0085	0195
0008	0196
0087	0197
0018	0198
0019	0199
0092	019A
008F	019B
001C	019C

001D	019D
001E	019E
001F	019F
0080	01A0
0081	01A1
0082	01A2
0083	01A3
0084	01A4
000A	01A5
0017	01A6
001B	01A7
0088	01A8
0089	01A9
008A	01AA
008B	01AB
008C	01AC
0005	01AD
0006	01AE
0007	01AF
0090	01B0
0091	01B1
0016	01B2
0093	01B3
0094	01B4
0095	01B5
0096	01B6
0004	01B7
0098	01B8
0099	01B9
009A	01BA
009B	01BB
0014	01BC
0015	01BD
009E	01BE
001A	01BF
009F	027F

Corresponding input mapping for an AWS DMS task:

```
{
  "SourceCharacterCodePoint": "0180", "TargetCharacterCodePoint": "0180"},
  {"SourceCharacterCodePoint": "0001", "TargetCharacterCodePoint": "0181"},
  {"SourceCharacterCodePoint": "0002", "TargetCharacterCodePoint": "0182"},
  {"SourceCharacterCodePoint": "0003", "TargetCharacterCodePoint": "0183"},
  {"SourceCharacterCodePoint": "009C", "TargetCharacterCodePoint": "0184"},
  {"SourceCharacterCodePoint": "0009", "TargetCharacterCodePoint": "0185"},
  {"SourceCharacterCodePoint": "0086", "TargetCharacterCodePoint": "0186"},
  {"SourceCharacterCodePoint": "007F", "TargetCharacterCodePoint": "0187"},
  {"SourceCharacterCodePoint": "0097", "TargetCharacterCodePoint": "0188"},
  {"SourceCharacterCodePoint": "008D", "TargetCharacterCodePoint": "0189"},
  {"SourceCharacterCodePoint": "008E", "TargetCharacterCodePoint": "018A"},
  {"SourceCharacterCodePoint": "000B", "TargetCharacterCodePoint": "018B"},
  {"SourceCharacterCodePoint": "000C", "TargetCharacterCodePoint": "018C"},
  {"SourceCharacterCodePoint": "000D", "TargetCharacterCodePoint": "018D"},
  {"SourceCharacterCodePoint": "000E", "TargetCharacterCodePoint": "018E"},
  {"SourceCharacterCodePoint": "000F", "TargetCharacterCodePoint": "018F"},
  {"SourceCharacterCodePoint": "0010", "TargetCharacterCodePoint": "0190"},
  {"SourceCharacterCodePoint": "0011", "TargetCharacterCodePoint": "0191"},
  {"SourceCharacterCodePoint": "0012", "TargetCharacterCodePoint": "0192"},
  {"SourceCharacterCodePoint": "0013", "TargetCharacterCodePoint": "0193"},
  {"SourceCharacterCodePoint": "009D", "TargetCharacterCodePoint": "0194"},
  {"SourceCharacterCodePoint": "0085", "TargetCharacterCodePoint": "0195"},
  {"SourceCharacterCodePoint": "0008", "TargetCharacterCodePoint": "0196"},
  {"SourceCharacterCodePoint": "0087", "TargetCharacterCodePoint": "0197"},
  {"SourceCharacterCodePoint": "0018", "TargetCharacterCodePoint": "0198"}
}
```

```
[{"SourceCharacterCodePoint": "0019", "TargetCharacterCodePoint": "0199"}, {"SourceCharacterCodePoint": "0092", "TargetCharacterCodePoint": "019A"}, {"SourceCharacterCodePoint": "008F", "TargetCharacterCodePoint": "019B"}, {"SourceCharacterCodePoint": "001C", "TargetCharacterCodePoint": "019C"}, {"SourceCharacterCodePoint": "001D", "TargetCharacterCodePoint": "019D"}, {"SourceCharacterCodePoint": "001E", "TargetCharacterCodePoint": "019E"}, {"SourceCharacterCodePoint": "001F", "TargetCharacterCodePoint": "019F"}, {"SourceCharacterCodePoint": "0080", "TargetCharacterCodePoint": "01A0"}, {"SourceCharacterCodePoint": "0081", "TargetCharacterCodePoint": "01A1"}, {"SourceCharacterCodePoint": "0082", "TargetCharacterCodePoint": "01A2"}, {"SourceCharacterCodePoint": "0083", "TargetCharacterCodePoint": "01A3"}, {"SourceCharacterCodePoint": "0084", "TargetCharacterCodePoint": "01A4"}, {"SourceCharacterCodePoint": "000A", "TargetCharacterCodePoint": "01A5"}, {"SourceCharacterCodePoint": "0017", "TargetCharacterCodePoint": "01A6"}, {"SourceCharacterCodePoint": "001B", "TargetCharacterCodePoint": "01A7"}, {"SourceCharacterCodePoint": "0088", "TargetCharacterCodePoint": "01A8"}, {"SourceCharacterCodePoint": "0089", "TargetCharacterCodePoint": "01A9"}, {"SourceCharacterCodePoint": "008A", "TargetCharacterCodePoint": "01AA"}, {"SourceCharacterCodePoint": "008B", "TargetCharacterCodePoint": "01AB"}, {"SourceCharacterCodePoint": "008C", "TargetCharacterCodePoint": "01AC"}, {"SourceCharacterCodePoint": "0005", "TargetCharacterCodePoint": "01AD"}, {"SourceCharacterCodePoint": "0006", "TargetCharacterCodePoint": "01AE"}, {"SourceCharacterCodePoint": "0007", "TargetCharacterCodePoint": "01AF"}, {"SourceCharacterCodePoint": "0090", "TargetCharacterCodePoint": "01B0"}, {"SourceCharacterCodePoint": "0091", "TargetCharacterCodePoint": "01B1"}, {"SourceCharacterCodePoint": "0016", "TargetCharacterCodePoint": "01B2"}, {"SourceCharacterCodePoint": "0093", "TargetCharacterCodePoint": "01B3"}, {"SourceCharacterCodePoint": "0094", "TargetCharacterCodePoint": "01B4"}, {"SourceCharacterCodePoint": "0095", "TargetCharacterCodePoint": "01B5"}, {"SourceCharacterCodePoint": "0096", "TargetCharacterCodePoint": "01B6"}, {"SourceCharacterCodePoint": "0004", "TargetCharacterCodePoint": "01B7"}, {"SourceCharacterCodePoint": "0098", "TargetCharacterCodePoint": "01B8"}, {"SourceCharacterCodePoint": "0099", "TargetCharacterCodePoint": "01B9"}, {"SourceCharacterCodePoint": "009A", "TargetCharacterCodePoint": "01BA"}, {"SourceCharacterCodePoint": "009B", "TargetCharacterCodePoint": "01BB"}, {"SourceCharacterCodePoint": "0014", "TargetCharacterCodePoint": "01BC"}, {"SourceCharacterCodePoint": "0015", "TargetCharacterCodePoint": "01BD"}, {"SourceCharacterCodePoint": "009E", "TargetCharacterCodePoint": "01BE"}, {"SourceCharacterCodePoint": "001A", "TargetCharacterCodePoint": "01BF"}, {"SourceCharacterCodePoint": "009F", "TargetCharacterCodePoint": "027F"}]
```

### de-DE-273b and de-DE-1141b

#### Code Shift:

0180	0180
0001	0181
0002	0182
0003	0183
009C	0184
0009	0185
0086	0186
007F	0187
0097	0188
008D	0189
008E	018A
000B	018B
000C	018C
000D	018D
000E	018E
000F	018F
0010	0190
0011	0191

0012	0192
0013	0193
009D	0194
0085	0195
0008	0196
0087	0197
0018	0198
0019	0199
0092	019A
008F	019B
001C	019C
001D	019D
001E	019E
001F	019F
0080	01A0
0081	01A1
0082	01A2
0083	01A3
0084	01A4
000A	01A5
0017	01A6
001B	01A7
0088	01A8
0089	01A9
008A	01AA
008B	01AB
008C	01AC
0005	01AD
0006	01AE
0007	01AF
0090	01B0
0091	01B1
0016	01B2
0093	01B3
0094	01B4
0095	01B5
0096	01B6
0004	01B7
0098	01B8
0099	01B9
009A	01BA
009B	01BB
0014	01BC
0015	01BD
009E	01BE
001A	01BF
009F	027F

Corresponding input mapping for an AWS DMS task:

```
{
  "SourceCharacterCodePoint": "0180", "TargetCharacterCodePoint": "0180"},  

  {"SourceCharacterCodePoint": "0001", "TargetCharacterCodePoint": "0181"},  

  {"SourceCharacterCodePoint": "0002", "TargetCharacterCodePoint": "0182"},  

  {"SourceCharacterCodePoint": "0003", "TargetCharacterCodePoint": "0183"},  

  {"SourceCharacterCodePoint": "009C", "TargetCharacterCodePoint": "0184"},  

  {"SourceCharacterCodePoint": "0009", "TargetCharacterCodePoint": "0185"},  

  {"SourceCharacterCodePoint": "0086", "TargetCharacterCodePoint": "0186"},  

  {"SourceCharacterCodePoint": "007F", "TargetCharacterCodePoint": "0187"},  

  {"SourceCharacterCodePoint": "0097", "TargetCharacterCodePoint": "0188"},  

  {"SourceCharacterCodePoint": "008D", "TargetCharacterCodePoint": "0189"},  

  {"SourceCharacterCodePoint": "008E", "TargetCharacterCodePoint": "018A"},  

  {"SourceCharacterCodePoint": "000B", "TargetCharacterCodePoint": "018B"},  

  {"SourceCharacterCodePoint": "000C", "TargetCharacterCodePoint": "018C"},  

  {"SourceCharacterCodePoint": "000D", "TargetCharacterCodePoint": "018D"}}
```

```
[{"SourceCharacterCodePoint": "000E", "TargetCharacterCodePoint": "018E"}, {"SourceCharacterCodePoint": "000F", "TargetCharacterCodePoint": "018F"}, {"SourceCharacterCodePoint": "0010", "TargetCharacterCodePoint": "0190"}, {"SourceCharacterCodePoint": "0011", "TargetCharacterCodePoint": "0191"}, {"SourceCharacterCodePoint": "0012", "TargetCharacterCodePoint": "0192"}, {"SourceCharacterCodePoint": "0013", "TargetCharacterCodePoint": "0193"}, {"SourceCharacterCodePoint": "009D", "TargetCharacterCodePoint": "0194"}, {"SourceCharacterCodePoint": "0085", "TargetCharacterCodePoint": "0195"}, {"SourceCharacterCodePoint": "0008", "TargetCharacterCodePoint": "0196"}, {"SourceCharacterCodePoint": "0087", "TargetCharacterCodePoint": "0197"}, {"SourceCharacterCodePoint": "0018", "TargetCharacterCodePoint": "0198"}, {"SourceCharacterCodePoint": "0019", "TargetCharacterCodePoint": "0199"}, {"SourceCharacterCodePoint": "0092", "TargetCharacterCodePoint": "019A"}, {"SourceCharacterCodePoint": "008F", "TargetCharacterCodePoint": "019B"}, {"SourceCharacterCodePoint": "001C", "TargetCharacterCodePoint": "019C"}, {"SourceCharacterCodePoint": "001D", "TargetCharacterCodePoint": "019D"}, {"SourceCharacterCodePoint": "001E", "TargetCharacterCodePoint": "019E"}, {"SourceCharacterCodePoint": "001F", "TargetCharacterCodePoint": "019F"}, {"SourceCharacterCodePoint": "0080", "TargetCharacterCodePoint": "01A0"}, {"SourceCharacterCodePoint": "0081", "TargetCharacterCodePoint": "01A1"}, {"SourceCharacterCodePoint": "0082", "TargetCharacterCodePoint": "01A2"}, {"SourceCharacterCodePoint": "0083", "TargetCharacterCodePoint": "01A3"}, {"SourceCharacterCodePoint": "0084", "TargetCharacterCodePoint": "01A4"}, {"SourceCharacterCodePoint": "000A", "TargetCharacterCodePoint": "01A5"}, {"SourceCharacterCodePoint": "0017", "TargetCharacterCodePoint": "01A6"}, {"SourceCharacterCodePoint": "001B", "TargetCharacterCodePoint": "01A7"}, {"SourceCharacterCodePoint": "0088", "TargetCharacterCodePoint": "01A8"}, {"SourceCharacterCodePoint": "0089", "TargetCharacterCodePoint": "01A9"}, {"SourceCharacterCodePoint": "008A", "TargetCharacterCodePoint": "01AA"}, {"SourceCharacterCodePoint": "008B", "TargetCharacterCodePoint": "01AB"}, {"SourceCharacterCodePoint": "008C", "TargetCharacterCodePoint": "01AC"}, {"SourceCharacterCodePoint": "0005", "TargetCharacterCodePoint": "01AD"}, {"SourceCharacterCodePoint": "0006", "TargetCharacterCodePoint": "01AE"}, {"SourceCharacterCodePoint": "0007", "TargetCharacterCodePoint": "01AF"}, {"SourceCharacterCodePoint": "0090", "TargetCharacterCodePoint": "01B0"}, {"SourceCharacterCodePoint": "0091", "TargetCharacterCodePoint": "01B1"}, {"SourceCharacterCodePoint": "0016", "TargetCharacterCodePoint": "01B2"}, {"SourceCharacterCodePoint": "0093", "TargetCharacterCodePoint": "01B3"}, {"SourceCharacterCodePoint": "0094", "TargetCharacterCodePoint": "01B4"}, {"SourceCharacterCodePoint": "0095", "TargetCharacterCodePoint": "01B5"}, {"SourceCharacterCodePoint": "0096", "TargetCharacterCodePoint": "01B6"}, {"SourceCharacterCodePoint": "0004", "TargetCharacterCodePoint": "01B7"}, {"SourceCharacterCodePoint": "0098", "TargetCharacterCodePoint": "01B8"}, {"SourceCharacterCodePoint": "0099", "TargetCharacterCodePoint": "01B9"}, {"SourceCharacterCodePoint": "009A", "TargetCharacterCodePoint": "01BA"}, {"SourceCharacterCodePoint": "009B", "TargetCharacterCodePoint": "01BB"}, {"SourceCharacterCodePoint": "0014", "TargetCharacterCodePoint": "01BC"}, {"SourceCharacterCodePoint": "0015", "TargetCharacterCodePoint": "01BD"}, {"SourceCharacterCodePoint": "009E", "TargetCharacterCodePoint": "01BE"}, {"SourceCharacterCodePoint": "001A", "TargetCharacterCodePoint": "01BF"}, {"SourceCharacterCodePoint": "009F", "TargetCharacterCodePoint": "027F"}]
```

### en-GB-285b and en-GB-1146b

Code Shift:

0180	0180
0001	0181
0002	0182
0003	0183
009C	0184
0009	0185
0086	0186

007F	0187
0097	0188
008D	0189
008E	018A
000B	018B
000C	018C
000D	018D
000E	018E
000F	018F
0010	0190
0011	0191
0012	0192
0013	0193
009D	0194
0085	0195
0008	0196
0087	0197
0018	0198
0019	0199
0092	019A
008F	019B
001C	019C
001D	019D
001E	019E
001F	019F
0080	01A0
0081	01A1
0082	01A2
0083	01A3
0084	01A4
000A	01A5
0017	01A6
001B	01A7
0088	01A8
0089	01A9
008A	01AA
008B	01AB
008C	01AC
0005	01AD
0006	01AE
0007	01AF
0090	01B0
0091	01B1
0016	01B2
0093	01B3
0094	01B4
0095	01B5
0096	01B6
0004	01B7
0098	01B8
0099	01B9
009A	01BA
009B	01BB
0014	01BC
0015	01BD
009E	01BE
001A	01BF
009F	027F

Corresponding input mapping for an AWS DMS task:

```
{"SourceCharacterCodePoint": "0180", "TargetCharacterCodePoint": "0180"}  
, {"SourceCharacterCodePoint": "0001", "TargetCharacterCodePoint": "0181"}  
, {"SourceCharacterCodePoint": "0002", "TargetCharacterCodePoint": "0182"}
```



**en-us-037b and en-us-1140b**

Code Shift:

0180	0180
0001	0181
0002	0182
0003	0183
009C	0184
0009	0185
0086	0186
007F	0187
0097	0188
008D	0189
008E	018A
000B	018B
000C	018C
000D	018D
000E	018E
000F	018F
0010	0190
0011	0191
0012	0192
0013	0193
009D	0194
0085	0195
0008	0196
0087	0197
0018	0198
0019	0199
0092	019A
008F	019B
001C	019C
001D	019D
001E	019E
001F	019F
0080	01A0
0081	01A1
0082	01A2
0083	01A3
0084	01A4
000A	01A5
0017	01A6
001B	01A7
0088	01A8
0089	01A9
008A	01AA
008B	01AB
008C	01AC
0005	01AD
0006	01AE
0007	01AF
0090	01B0
0091	01B1
0016	01B2
0093	01B3
0094	01B4
0095	01B5
0096	01B6
0004	01B7
0098	01B8
0099	01B9
009A	01BA
009B	01BB
0014	01BC

0015	01BD
009E	01BE
001A	01BF
009F	027F

Corresponding input mapping for an AWS DMS task:

```
{"SourceCharacterCodePoint": "0180", "TargetCharacterCodePoint": "0180"}  
, {"SourceCharacterCodePoint": "0001", "TargetCharacterCodePoint": "0181"}  
, {"SourceCharacterCodePoint": "0002", "TargetCharacterCodePoint": "0182"}  
, {"SourceCharacterCodePoint": "0003", "TargetCharacterCodePoint": "0183"}  
, {"SourceCharacterCodePoint": "009C", "TargetCharacterCodePoint": "0184"}  
, {"SourceCharacterCodePoint": "0009", "TargetCharacterCodePoint": "0185"}  
, {"SourceCharacterCodePoint": "0086", "TargetCharacterCodePoint": "0186"}  
, {"SourceCharacterCodePoint": "007F", "TargetCharacterCodePoint": "0187"}  
, {"SourceCharacterCodePoint": "0097", "TargetCharacterCodePoint": "0188"}  
, {"SourceCharacterCodePoint": "008D", "TargetCharacterCodePoint": "0189"}  
, {"SourceCharacterCodePoint": "008E", "TargetCharacterCodePoint": "018A"}  
, {"SourceCharacterCodePoint": "000B", "TargetCharacterCodePoint": "018B"}  
, {"SourceCharacterCodePoint": "000C", "TargetCharacterCodePoint": "018C"}  
, {"SourceCharacterCodePoint": "000D", "TargetCharacterCodePoint": "018D"}  
, {"SourceCharacterCodePoint": "000E", "TargetCharacterCodePoint": "018E"}  
, {"SourceCharacterCodePoint": "000F", "TargetCharacterCodePoint": "018F"}  
, {"SourceCharacterCodePoint": "0010", "TargetCharacterCodePoint": "0190"}  
, {"SourceCharacterCodePoint": "0011", "TargetCharacterCodePoint": "0191"}  
, {"SourceCharacterCodePoint": "0012", "TargetCharacterCodePoint": "0192"}  
, {"SourceCharacterCodePoint": "0013", "TargetCharacterCodePoint": "0193"}  
, {"SourceCharacterCodePoint": "009D", "TargetCharacterCodePoint": "0194"}  
, {"SourceCharacterCodePoint": "0085", "TargetCharacterCodePoint": "0195"}  
, {"SourceCharacterCodePoint": "0008", "TargetCharacterCodePoint": "0196"}  
, {"SourceCharacterCodePoint": "0087", "TargetCharacterCodePoint": "0197"}  
, {"SourceCharacterCodePoint": "0018", "TargetCharacterCodePoint": "0198"}  
, {"SourceCharacterCodePoint": "0019", "TargetCharacterCodePoint": "0199"}  
, {"SourceCharacterCodePoint": "0092", "TargetCharacterCodePoint": "019A"}  
, {"SourceCharacterCodePoint": "008F", "TargetCharacterCodePoint": "019B"}  
, {"SourceCharacterCodePoint": "001C", "TargetCharacterCodePoint": "019C"}  
, {"SourceCharacterCodePoint": "001D", "TargetCharacterCodePoint": "019D"}  
, {"SourceCharacterCodePoint": "001E", "TargetCharacterCodePoint": "019E"}  
, {"SourceCharacterCodePoint": "001F", "TargetCharacterCodePoint": "019F"}  
, {"SourceCharacterCodePoint": "0080", "TargetCharacterCodePoint": "01A0"}  
, {"SourceCharacterCodePoint": "0081", "TargetCharacterCodePoint": "01A1"}  
, {"SourceCharacterCodePoint": "0082", "TargetCharacterCodePoint": "01A2"}  
, {"SourceCharacterCodePoint": "0083", "TargetCharacterCodePoint": "01A3"}  
, {"SourceCharacterCodePoint": "0084", "TargetCharacterCodePoint": "01A4"}  
, {"SourceCharacterCodePoint": "000A", "TargetCharacterCodePoint": "01A5"}  
, {"SourceCharacterCodePoint": "0017", "TargetCharacterCodePoint": "01A6"}  
, {"SourceCharacterCodePoint": "001B", "TargetCharacterCodePoint": "01A7"}  
, {"SourceCharacterCodePoint": "0088", "TargetCharacterCodePoint": "01A8"}  
, {"SourceCharacterCodePoint": "0089", "TargetCharacterCodePoint": "01A9"}  
, {"SourceCharacterCodePoint": "008A", "TargetCharacterCodePoint": "01AA"}  
, {"SourceCharacterCodePoint": "008B", "TargetCharacterCodePoint": "01AB"}  
, {"SourceCharacterCodePoint": "008C", "TargetCharacterCodePoint": "01AC"}  
, {"SourceCharacterCodePoint": "0005", "TargetCharacterCodePoint": "01AD"}  
, {"SourceCharacterCodePoint": "0006", "TargetCharacterCodePoint": "01AE"}  
, {"SourceCharacterCodePoint": "0007", "TargetCharacterCodePoint": "01AF"}  
, {"SourceCharacterCodePoint": "0090", "TargetCharacterCodePoint": "01B0"}  
, {"SourceCharacterCodePoint": "0091", "TargetCharacterCodePoint": "01B1"}  
, {"SourceCharacterCodePoint": "0016", "TargetCharacterCodePoint": "01B2"}  
, {"SourceCharacterCodePoint": "0093", "TargetCharacterCodePoint": "01B3"}  
, {"SourceCharacterCodePoint": "0094", "TargetCharacterCodePoint": "01B4"}  
, {"SourceCharacterCodePoint": "0095", "TargetCharacterCodePoint": "01B5"}  
, {"SourceCharacterCodePoint": "0096", "TargetCharacterCodePoint": "01B6"}  
, {"SourceCharacterCodePoint": "0004", "TargetCharacterCodePoint": "01B7"}  
, {"SourceCharacterCodePoint": "0098", "TargetCharacterCodePoint": "01B8"}
```

```
, {"SourceCharacterCodePoint": "0099", "TargetCharacterCodePoint": "01B9"}  
, {"SourceCharacterCodePoint": "009A", "TargetCharacterCodePoint": "01BA"}  
, {"SourceCharacterCodePoint": "009B", "TargetCharacterCodePoint": "01BB"}  
, {"SourceCharacterCodePoint": "0014", "TargetCharacterCodePoint": "01BC"}  
, {"SourceCharacterCodePoint": "0015", "TargetCharacterCodePoint": "01BD"}  
, {"SourceCharacterCodePoint": "009E", "TargetCharacterCodePoint": "01BE"}  
, {"SourceCharacterCodePoint": "001A", "TargetCharacterCodePoint": "01BF"}  
, {"SourceCharacterCodePoint": "009F", "TargetCharacterCodePoint": "027F"}
```

### es-ES-284b and es-ES-1145b

Code Shift:

0180	0180
0001	0181
0002	0182
0003	0183
009C	0184
0009	0185
0086	0186
007F	0187
0097	0188
008D	0189
008E	018A
000B	018B
000C	018C
000D	018D
000E	018E
000F	018F
0010	0190
0011	0191
0012	0192
0013	0193
009D	0194
0085	0195
0008	0196
0087	0197
0018	0198
0019	0199
0092	019A
008F	019B
001C	019C
001D	019D
001E	019E
001F	019F
0080	01A0
0081	01A1
0082	01A2
0083	01A3
0084	01A4
000A	01A5
0017	01A6
001B	01A7
0088	01A8
0089	01A9
008A	01AA
008B	01AB
008C	01AC
0005	01AD
0006	01AE
0007	01AF
0090	01B0
0091	01B1
0016	01B2

0093	01B3
0094	01B4
0095	01B5
0096	01B6
0004	01B7
0098	01B8
0099	01B9
009A	01BA
009B	01BB
0014	01BC
0015	01BD
009E	01BE
001A	01BF
009F	027F

Corresponding input mapping for an AWS DMS task:

```
{
  "SourceCharacterCodePoint": "0180", "TargetCharacterCodePoint": "0180"},
  {"SourceCharacterCodePoint": "0001", "TargetCharacterCodePoint": "0181"},
  {"SourceCharacterCodePoint": "0002", "TargetCharacterCodePoint": "0182"},
  {"SourceCharacterCodePoint": "0003", "TargetCharacterCodePoint": "0183"},
  {"SourceCharacterCodePoint": "009C", "TargetCharacterCodePoint": "0184"},
  {"SourceCharacterCodePoint": "0009", "TargetCharacterCodePoint": "0185"},
  {"SourceCharacterCodePoint": "0086", "TargetCharacterCodePoint": "0186"},
  {"SourceCharacterCodePoint": "007F", "TargetCharacterCodePoint": "0187"},
  {"SourceCharacterCodePoint": "0097", "TargetCharacterCodePoint": "0188"},
  {"SourceCharacterCodePoint": "008D", "TargetCharacterCodePoint": "0189"},
  {"SourceCharacterCodePoint": "008E", "TargetCharacterCodePoint": "018A"},
  {"SourceCharacterCodePoint": "000B", "TargetCharacterCodePoint": "018B"},
  {"SourceCharacterCodePoint": "000C", "TargetCharacterCodePoint": "018C"},
  {"SourceCharacterCodePoint": "000D", "TargetCharacterCodePoint": "018D"},
  {"SourceCharacterCodePoint": "000E", "TargetCharacterCodePoint": "018E"},
  {"SourceCharacterCodePoint": "000F", "TargetCharacterCodePoint": "018F"},
  {"SourceCharacterCodePoint": "0010", "TargetCharacterCodePoint": "0190"},
  {"SourceCharacterCodePoint": "0011", "TargetCharacterCodePoint": "0191"},
  {"SourceCharacterCodePoint": "0012", "TargetCharacterCodePoint": "0192"},
  {"SourceCharacterCodePoint": "0013", "TargetCharacterCodePoint": "0193"},
  {"SourceCharacterCodePoint": "009D", "TargetCharacterCodePoint": "0194"},
  {"SourceCharacterCodePoint": "0085", "TargetCharacterCodePoint": "0195"},
  {"SourceCharacterCodePoint": "0008", "TargetCharacterCodePoint": "0196"},
  {"SourceCharacterCodePoint": "0087", "TargetCharacterCodePoint": "0197"},
  {"SourceCharacterCodePoint": "0018", "TargetCharacterCodePoint": "0198"},
  {"SourceCharacterCodePoint": "0019", "TargetCharacterCodePoint": "0199"},
  {"SourceCharacterCodePoint": "0092", "TargetCharacterCodePoint": "019A"},
  {"SourceCharacterCodePoint": "008F", "TargetCharacterCodePoint": "019B"},
  {"SourceCharacterCodePoint": "001C", "TargetCharacterCodePoint": "019C"},
  {"SourceCharacterCodePoint": "001D", "TargetCharacterCodePoint": "019D"},
  {"SourceCharacterCodePoint": "001E", "TargetCharacterCodePoint": "019E"},
  {"SourceCharacterCodePoint": "001F", "TargetCharacterCodePoint": "019F"},
  {"SourceCharacterCodePoint": "0080", "TargetCharacterCodePoint": "01A0"},
  {"SourceCharacterCodePoint": "0081", "TargetCharacterCodePoint": "01A1"},
  {"SourceCharacterCodePoint": "0082", "TargetCharacterCodePoint": "01A2"},
  {"SourceCharacterCodePoint": "0083", "TargetCharacterCodePoint": "01A3"},
  {"SourceCharacterCodePoint": "0084", "TargetCharacterCodePoint": "01A4"},
  {"SourceCharacterCodePoint": "000A", "TargetCharacterCodePoint": "01A5"},
  {"SourceCharacterCodePoint": "0017", "TargetCharacterCodePoint": "01A6"},
  {"SourceCharacterCodePoint": "001B", "TargetCharacterCodePoint": "01A7"},
  {"SourceCharacterCodePoint": "0088", "TargetCharacterCodePoint": "01A8"},
  {"SourceCharacterCodePoint": "0089", "TargetCharacterCodePoint": "01A9"},
  {"SourceCharacterCodePoint": "008A", "TargetCharacterCodePoint": "01AA"},
  {"SourceCharacterCodePoint": "008B", "TargetCharacterCodePoint": "01AB"},
  {"SourceCharacterCodePoint": "008C", "TargetCharacterCodePoint": "01AC"},
  {"SourceCharacterCodePoint": "0005", "TargetCharacterCodePoint": "01AD"},
  {"SourceCharacterCodePoint": "0006", "TargetCharacterCodePoint": "01AE"}
}
```

```
[{"SourceCharacterCodePoint": "0007", "TargetCharacterCodePoint": "01AF"}, {"SourceCharacterCodePoint": "0090", "TargetCharacterCodePoint": "01B0"}, {"SourceCharacterCodePoint": "0091", "TargetCharacterCodePoint": "01B1"}, {"SourceCharacterCodePoint": "0016", "TargetCharacterCodePoint": "01B2"}, {"SourceCharacterCodePoint": "0093", "TargetCharacterCodePoint": "01B3"}, {"SourceCharacterCodePoint": "0094", "TargetCharacterCodePoint": "01B4"}, {"SourceCharacterCodePoint": "0095", "TargetCharacterCodePoint": "01B5"}, {"SourceCharacterCodePoint": "0096", "TargetCharacterCodePoint": "01B6"}, {"SourceCharacterCodePoint": "0004", "TargetCharacterCodePoint": "01B7"}, {"SourceCharacterCodePoint": "0098", "TargetCharacterCodePoint": "01B8"}, {"SourceCharacterCodePoint": "0099", "TargetCharacterCodePoint": "01B9"}, {"SourceCharacterCodePoint": "009A", "TargetCharacterCodePoint": "01BA"}, {"SourceCharacterCodePoint": "009B", "TargetCharacterCodePoint": "01BB"}, {"SourceCharacterCodePoint": "0014", "TargetCharacterCodePoint": "01BC"}, {"SourceCharacterCodePoint": "0015", "TargetCharacterCodePoint": "01BD"}, {"SourceCharacterCodePoint": "009E", "TargetCharacterCodePoint": "01BE"}, {"SourceCharacterCodePoint": "001A", "TargetCharacterCodePoint": "01BF"}, {"SourceCharacterCodePoint": "009F", "TargetCharacterCodePoint": "027F"}]
```

### fi\_FI-278b and fi\_FI-1143b

#### Code Shift:

0180	0180
0001	0181
0002	0182
0003	0183
009C	0184
0009	0185
0086	0186
007F	0187
0097	0188
008D	0189
008E	018A
000B	018B
000C	018C
000D	018D
000E	018E
000F	018F
0010	0190
0011	0191
0012	0192
0013	0193
009D	0194
0085	0195
0008	0196
0087	0197
0018	0198
0019	0199
0092	019A
008F	019B
001C	019C
001D	019D
001E	019E
001F	019F
0080	01A0
0081	01A1
0082	01A2
0083	01A3
0084	01A4
000A	01A5
0017	01A6
001B	01A7

0088	01A8
0089	01A9
008A	01AA
008B	01AB
008C	01AC
0005	01AD
0006	01AE
0007	01AF
0090	01B0
0091	01B1
0016	01B2
0093	01B3
0094	01B4
0095	01B5
0096	01B6
0004	01B7
0098	01B8
0099	01B9
009A	01BA
009B	01BB
0014	01BC
0015	01BD
009E	01BE
001A	01BF
009F	027F

Corresponding input mapping for an AWS DMS task:

```
{
  "SourceCharacterCodePoint": "0180", "TargetCharacterCodePoint": "0180"},  

  {"SourceCharacterCodePoint": "0001", "TargetCharacterCodePoint": "0181"},  

  {"SourceCharacterCodePoint": "0002", "TargetCharacterCodePoint": "0182"},  

  {"SourceCharacterCodePoint": "0003", "TargetCharacterCodePoint": "0183"},  

  {"SourceCharacterCodePoint": "009C", "TargetCharacterCodePoint": "0184"},  

  {"SourceCharacterCodePoint": "0009", "TargetCharacterCodePoint": "0185"},  

  {"SourceCharacterCodePoint": "0086", "TargetCharacterCodePoint": "0186"},  

  {"SourceCharacterCodePoint": "007F", "TargetCharacterCodePoint": "0187"},  

  {"SourceCharacterCodePoint": "0097", "TargetCharacterCodePoint": "0188"},  

  {"SourceCharacterCodePoint": "008D", "TargetCharacterCodePoint": "0189"},  

  {"SourceCharacterCodePoint": "008E", "TargetCharacterCodePoint": "018A"},  

  {"SourceCharacterCodePoint": "000B", "TargetCharacterCodePoint": "018B"},  

  {"SourceCharacterCodePoint": "000C", "TargetCharacterCodePoint": "018C"},  

  {"SourceCharacterCodePoint": "000D", "TargetCharacterCodePoint": "018D"},  

  {"SourceCharacterCodePoint": "000E", "TargetCharacterCodePoint": "018E"},  

  {"SourceCharacterCodePoint": "000F", "TargetCharacterCodePoint": "018F"},  

  {"SourceCharacterCodePoint": "0010", "TargetCharacterCodePoint": "0190"},  

  {"SourceCharacterCodePoint": "0011", "TargetCharacterCodePoint": "0191"},  

  {"SourceCharacterCodePoint": "0012", "TargetCharacterCodePoint": "0192"},  

  {"SourceCharacterCodePoint": "0013", "TargetCharacterCodePoint": "0193"},  

  {"SourceCharacterCodePoint": "009D", "TargetCharacterCodePoint": "0194"},  

  {"SourceCharacterCodePoint": "0085", "TargetCharacterCodePoint": "0195"},  

  {"SourceCharacterCodePoint": "0008", "TargetCharacterCodePoint": "0196"},  

  {"SourceCharacterCodePoint": "0087", "TargetCharacterCodePoint": "0197"},  

  {"SourceCharacterCodePoint": "0018", "TargetCharacterCodePoint": "0198"},  

  {"SourceCharacterCodePoint": "0019", "TargetCharacterCodePoint": "0199"},  

  {"SourceCharacterCodePoint": "0092", "TargetCharacterCodePoint": "019A"},  

  {"SourceCharacterCodePoint": "008F", "TargetCharacterCodePoint": "019B"},  

  {"SourceCharacterCodePoint": "001C", "TargetCharacterCodePoint": "019C"},  

  {"SourceCharacterCodePoint": "001D", "TargetCharacterCodePoint": "019D"},  

  {"SourceCharacterCodePoint": "001E", "TargetCharacterCodePoint": "019E"},  

  {"SourceCharacterCodePoint": "001F", "TargetCharacterCodePoint": "019F"},  

  {"SourceCharacterCodePoint": "0080", "TargetCharacterCodePoint": "01A0"},  

  {"SourceCharacterCodePoint": "0081", "TargetCharacterCodePoint": "01A1"},  

  {"SourceCharacterCodePoint": "0082", "TargetCharacterCodePoint": "01A2"},  

  {"SourceCharacterCodePoint": "0083", "TargetCharacterCodePoint": "01A3"}}
```

```
[{"SourceCharacterCodePoint": "0084", "TargetCharacterCodePoint": "01A4"}, {"SourceCharacterCodePoint": "000A", "TargetCharacterCodePoint": "01A5"}, {"SourceCharacterCodePoint": "0017", "TargetCharacterCodePoint": "01A6"}, {"SourceCharacterCodePoint": "001B", "TargetCharacterCodePoint": "01A7"}, {"SourceCharacterCodePoint": "0088", "TargetCharacterCodePoint": "01A8"}, {"SourceCharacterCodePoint": "0089", "TargetCharacterCodePoint": "01A9"}, {"SourceCharacterCodePoint": "008A", "TargetCharacterCodePoint": "01AA"}, {"SourceCharacterCodePoint": "008B", "TargetCharacterCodePoint": "01AB"}, {"SourceCharacterCodePoint": "008C", "TargetCharacterCodePoint": "01AC"}, {"SourceCharacterCodePoint": "0005", "TargetCharacterCodePoint": "01AD"}, {"SourceCharacterCodePoint": "0006", "TargetCharacterCodePoint": "01AE"}, {"SourceCharacterCodePoint": "0007", "TargetCharacterCodePoint": "01AF"}, {"SourceCharacterCodePoint": "0090", "TargetCharacterCodePoint": "01B0"}, {"SourceCharacterCodePoint": "0091", "TargetCharacterCodePoint": "01B1"}, {"SourceCharacterCodePoint": "0016", "TargetCharacterCodePoint": "01B2"}, {"SourceCharacterCodePoint": "0093", "TargetCharacterCodePoint": "01B3"}, {"SourceCharacterCodePoint": "0094", "TargetCharacterCodePoint": "01B4"}, {"SourceCharacterCodePoint": "0095", "TargetCharacterCodePoint": "01B5"}, {"SourceCharacterCodePoint": "0096", "TargetCharacterCodePoint": "01B6"}, {"SourceCharacterCodePoint": "0004", "TargetCharacterCodePoint": "01B7"}, {"SourceCharacterCodePoint": "0098", "TargetCharacterCodePoint": "01B8"}, {"SourceCharacterCodePoint": "0099", "TargetCharacterCodePoint": "01B9"}, {"SourceCharacterCodePoint": "009A", "TargetCharacterCodePoint": "01BA"}, {"SourceCharacterCodePoint": "009B", "TargetCharacterCodePoint": "01BB"}, {"SourceCharacterCodePoint": "0014", "TargetCharacterCodePoint": "01BC"}, {"SourceCharacterCodePoint": "0015", "TargetCharacterCodePoint": "01BD"}, {"SourceCharacterCodePoint": "009E", "TargetCharacterCodePoint": "01BE"}, {"SourceCharacterCodePoint": "001A", "TargetCharacterCodePoint": "01BF"}, {"SourceCharacterCodePoint": "009F", "TargetCharacterCodePoint": "027F"}]
```

### **fr-FR-297b and fr-FR-1147b**

Code Shift:

0180	0180
0001	0181
0002	0182
0003	0183
009C	0184
0009	0185
0086	0186
007F	0187
0097	0188
008D	0189
008E	018A
000B	018B
000C	018C
000D	018D
000E	018E
000F	018F
0010	0190
0011	0191
0012	0192
0013	0193
009D	0194
0085	0195
0008	0196
0087	0197
0018	0198
0019	0199
0092	019A
008F	019B
001C	019C
001D	019D

001E	019E
001F	019F
0080	01A0
0081	01A1
0082	01A2
0083	01A3
0084	01A4
000A	01A5
0017	01A6
001B	01A7
0088	01A8
0089	01A9
008A	01AA
008B	01AB
008C	01AC
0005	01AD
0006	01AE
0007	01AF
0090	01B0
0091	01B1
0016	01B2
0093	01B3
0094	01B4
0095	01B5
0096	01B6
0004	01B7
0098	01B8
0099	01B9
009A	01BA
009B	01BB
0014	01BC
0015	01BD
009E	01BE
001A	01BF
009F	027F

Corresponding input mapping for an AWS DMS task:

```
{"SourceCharacterCodePoint": "0180", "TargetCharacterCodePoint": "0180"},  
{"SourceCharacterCodePoint": "0001", "TargetCharacterCodePoint": "0181"},  
{"SourceCharacterCodePoint": "0002", "TargetCharacterCodePoint": "0182"},  
{"SourceCharacterCodePoint": "0003", "TargetCharacterCodePoint": "0183"},  
{"SourceCharacterCodePoint": "009C", "TargetCharacterCodePoint": "0184"},  
{"SourceCharacterCodePoint": "0009", "TargetCharacterCodePoint": "0185"},  
{"SourceCharacterCodePoint": "0086", "TargetCharacterCodePoint": "0186"},  
{"SourceCharacterCodePoint": "007F", "TargetCharacterCodePoint": "0187"},  
 {"SourceCharacterCodePoint": "0097", "TargetCharacterCodePoint": "0188"},  
 {"SourceCharacterCodePoint": "008D", "TargetCharacterCodePoint": "0189"},  
 {"SourceCharacterCodePoint": "008E", "TargetCharacterCodePoint": "018A"},  
 {"SourceCharacterCodePoint": "000B", "TargetCharacterCodePoint": "018B"},  
 {"SourceCharacterCodePoint": "000C", "TargetCharacterCodePoint": "018C"},  
 {"SourceCharacterCodePoint": "000D", "TargetCharacterCodePoint": "018D"},  
 {"SourceCharacterCodePoint": "000E", "TargetCharacterCodePoint": "018E"},  
 {"SourceCharacterCodePoint": "000F", "TargetCharacterCodePoint": "018F"},  
 {"SourceCharacterCodePoint": "0010", "TargetCharacterCodePoint": "0190"},  
 {"SourceCharacterCodePoint": "0011", "TargetCharacterCodePoint": "0191"},  
 {"SourceCharacterCodePoint": "0012", "TargetCharacterCodePoint": "0192"},  
 {"SourceCharacterCodePoint": "0013", "TargetCharacterCodePoint": "0193"},  
 {"SourceCharacterCodePoint": "009D", "TargetCharacterCodePoint": "0194"},  
 {"SourceCharacterCodePoint": "0085", "TargetCharacterCodePoint": "0195"},  
 {"SourceCharacterCodePoint": "0008", "TargetCharacterCodePoint": "0196"},  
 {"SourceCharacterCodePoint": "0087", "TargetCharacterCodePoint": "0197"},  
 {"SourceCharacterCodePoint": "0018", "TargetCharacterCodePoint": "0198"},  
 {"SourceCharacterCodePoint": "0019", "TargetCharacterCodePoint": "0199"}
```

```
[{"SourceCharacterCodePoint": "0092", "TargetCharacterCodePoint": "019A"}, {"SourceCharacterCodePoint": "008F", "TargetCharacterCodePoint": "019B"}, {"SourceCharacterCodePoint": "001C", "TargetCharacterCodePoint": "019C"}, {"SourceCharacterCodePoint": "001D", "TargetCharacterCodePoint": "019D"}, {"SourceCharacterCodePoint": "001E", "TargetCharacterCodePoint": "019E"}, {"SourceCharacterCodePoint": "001F", "TargetCharacterCodePoint": "019F"}, {"SourceCharacterCodePoint": "0080", "TargetCharacterCodePoint": "01A0"}, {"SourceCharacterCodePoint": "0081", "TargetCharacterCodePoint": "01A1"}, {"SourceCharacterCodePoint": "0082", "TargetCharacterCodePoint": "01A2"}, {"SourceCharacterCodePoint": "0083", "TargetCharacterCodePoint": "01A3"}, {"SourceCharacterCodePoint": "0084", "TargetCharacterCodePoint": "01A4"}, {"SourceCharacterCodePoint": "000A", "TargetCharacterCodePoint": "01A5"}, {"SourceCharacterCodePoint": "0017", "TargetCharacterCodePoint": "01A6"}, {"SourceCharacterCodePoint": "001B", "TargetCharacterCodePoint": "01A7"}, {"SourceCharacterCodePoint": "0088", "TargetCharacterCodePoint": "01A8"}, {"SourceCharacterCodePoint": "0089", "TargetCharacterCodePoint": "01A9"}, {"SourceCharacterCodePoint": "008A", "TargetCharacterCodePoint": "01AA"}, {"SourceCharacterCodePoint": "008B", "TargetCharacterCodePoint": "01AB"}, {"SourceCharacterCodePoint": "008C", "TargetCharacterCodePoint": "01AC"}, {"SourceCharacterCodePoint": "0005", "TargetCharacterCodePoint": "01AD"}, {"SourceCharacterCodePoint": "0006", "TargetCharacterCodePoint": "01AE"}, {"SourceCharacterCodePoint": "0007", "TargetCharacterCodePoint": "01AF"}, {"SourceCharacterCodePoint": "0090", "TargetCharacterCodePoint": "01B0"}, {"SourceCharacterCodePoint": "0091", "TargetCharacterCodePoint": "01B1"}, {"SourceCharacterCodePoint": "0016", "TargetCharacterCodePoint": "01B2"}, {"SourceCharacterCodePoint": "0093", "TargetCharacterCodePoint": "01B3"}, {"SourceCharacterCodePoint": "0094", "TargetCharacterCodePoint": "01B4"}, {"SourceCharacterCodePoint": "0095", "TargetCharacterCodePoint": "01B5"}, {"SourceCharacterCodePoint": "0096", "TargetCharacterCodePoint": "01B6"}, {"SourceCharacterCodePoint": "0004", "TargetCharacterCodePoint": "01B7"}, {"SourceCharacterCodePoint": "0098", "TargetCharacterCodePoint": "01B8"}, {"SourceCharacterCodePoint": "0099", "TargetCharacterCodePoint": "01B9"}, {"SourceCharacterCodePoint": "009A", "TargetCharacterCodePoint": "01BA"}, {"SourceCharacterCodePoint": "009B", "TargetCharacterCodePoint": "01BB"}, {"SourceCharacterCodePoint": "0014", "TargetCharacterCodePoint": "01BC"}, {"SourceCharacterCodePoint": "0015", "TargetCharacterCodePoint": "01BD"}, {"SourceCharacterCodePoint": "009E", "TargetCharacterCodePoint": "01BE"}, {"SourceCharacterCodePoint": "001A", "TargetCharacterCodePoint": "01BF"}, {"SourceCharacterCodePoint": "009F", "TargetCharacterCodePoint": "027F"}]
```

### it-IT-280b and it-IT-1144b

#### Code Shift:

0180	0180
0001	0181
0002	0182
0003	0183
009C	0184
0009	0185
0086	0186
007F	0187
0097	0188
008D	0189
008E	018A
000B	018B
000C	018C
000D	018D
000E	018E
000F	018F
0010	0190
0011	0191
0012	0192

0013	0193
009D	0194
0085	0195
0008	0196
0087	0197
0018	0198
0019	0199
0092	019A
008F	019B
001C	019C
001D	019D
001E	019E
001F	019F
0080	01A0
0081	01A1
0082	01A2
0083	01A3
0084	01A4
000A	01A5
0017	01A6
001B	01A7
0088	01A8
0089	01A9
008A	01AA
008B	01AB
008C	01AC
0005	01AD
0006	01AE
0007	01AF
0090	01B0
0091	01B1
0016	01B2
0093	01B3
0094	01B4
0095	01B5
0096	01B6
0004	01B7
0098	01B8
0099	01B9
009A	01BA
009B	01BB
0014	01BC
0015	01BD
009E	01BE
001A	01BF
009F	027F

Corresponding input mapping for an AWS DMS task:

```
{
    "SourceCharacterCodePoint": "0180", "TargetCharacterCodePoint": "0180"},  

    {"SourceCharacterCodePoint": "0001", "TargetCharacterCodePoint": "0181"},  

    {"SourceCharacterCodePoint": "0002", "TargetCharacterCodePoint": "0182"},  

    {"SourceCharacterCodePoint": "0003", "TargetCharacterCodePoint": "0183"},  

    {"SourceCharacterCodePoint": "009C", "TargetCharacterCodePoint": "0184"},  

    {"SourceCharacterCodePoint": "0009", "TargetCharacterCodePoint": "0185"},  

    {"SourceCharacterCodePoint": "0086", "TargetCharacterCodePoint": "0186"},  

    {"SourceCharacterCodePoint": "007F", "TargetCharacterCodePoint": "0187"},  

    {"SourceCharacterCodePoint": "0097", "TargetCharacterCodePoint": "0188"},  

    {"SourceCharacterCodePoint": "008D", "TargetCharacterCodePoint": "0189"},  

    {"SourceCharacterCodePoint": "008E", "TargetCharacterCodePoint": "018A"},  

    {"SourceCharacterCodePoint": "000B", "TargetCharacterCodePoint": "018B"},  

    {"SourceCharacterCodePoint": "000C", "TargetCharacterCodePoint": "018C"},  

    {"SourceCharacterCodePoint": "000D", "TargetCharacterCodePoint": "018D"},  

    {"SourceCharacterCodePoint": "000E", "TargetCharacterCodePoint": "018E"}}
```

```
[{"SourceCharacterCodePoint": "000F", "TargetCharacterCodePoint": "018F"}, {"SourceCharacterCodePoint": "0010", "TargetCharacterCodePoint": "0190"}, {"SourceCharacterCodePoint": "0011", "TargetCharacterCodePoint": "0191"}, {"SourceCharacterCodePoint": "0012", "TargetCharacterCodePoint": "0192"}, {"SourceCharacterCodePoint": "0013", "TargetCharacterCodePoint": "0193"}, {"SourceCharacterCodePoint": "009D", "TargetCharacterCodePoint": "0194"}, {"SourceCharacterCodePoint": "0085", "TargetCharacterCodePoint": "0195"}, {"SourceCharacterCodePoint": "0008", "TargetCharacterCodePoint": "0196"}, {"SourceCharacterCodePoint": "0087", "TargetCharacterCodePoint": "0197"}, {"SourceCharacterCodePoint": "0018", "TargetCharacterCodePoint": "0198"}, {"SourceCharacterCodePoint": "0019", "TargetCharacterCodePoint": "0199"}, {"SourceCharacterCodePoint": "0092", "TargetCharacterCodePoint": "019A"}, {"SourceCharacterCodePoint": "008F", "TargetCharacterCodePoint": "019B"}, {"SourceCharacterCodePoint": "001C", "TargetCharacterCodePoint": "019C"}, {"SourceCharacterCodePoint": "001D", "TargetCharacterCodePoint": "019D"}, {"SourceCharacterCodePoint": "001E", "TargetCharacterCodePoint": "019E"}, {"SourceCharacterCodePoint": "001F", "TargetCharacterCodePoint": "019F"}, {"SourceCharacterCodePoint": "0080", "TargetCharacterCodePoint": "01A0"}, {"SourceCharacterCodePoint": "0081", "TargetCharacterCodePoint": "01A1"}, {"SourceCharacterCodePoint": "0082", "TargetCharacterCodePoint": "01A2"}, {"SourceCharacterCodePoint": "0083", "TargetCharacterCodePoint": "01A3"}, {"SourceCharacterCodePoint": "0084", "TargetCharacterCodePoint": "01A4"}, {"SourceCharacterCodePoint": "000A", "TargetCharacterCodePoint": "01A5"}, {"SourceCharacterCodePoint": "0017", "TargetCharacterCodePoint": "01A6"}, {"SourceCharacterCodePoint": "001B", "TargetCharacterCodePoint": "01A7"}, {"SourceCharacterCodePoint": "0088", "TargetCharacterCodePoint": "01A8"}, {"SourceCharacterCodePoint": "0089", "TargetCharacterCodePoint": "01A9"}, {"SourceCharacterCodePoint": "008A", "TargetCharacterCodePoint": "01AA"}, {"SourceCharacterCodePoint": "008B", "TargetCharacterCodePoint": "01AB"}, {"SourceCharacterCodePoint": "008C", "TargetCharacterCodePoint": "01AC"}, {"SourceCharacterCodePoint": "0005", "TargetCharacterCodePoint": "01AD"}, {"SourceCharacterCodePoint": "0006", "TargetCharacterCodePoint": "01AE"}, {"SourceCharacterCodePoint": "0007", "TargetCharacterCodePoint": "01AF"}, {"SourceCharacterCodePoint": "0090", "TargetCharacterCodePoint": "01B0"}, {"SourceCharacterCodePoint": "0091", "TargetCharacterCodePoint": "01B1"}, {"SourceCharacterCodePoint": "0016", "TargetCharacterCodePoint": "01B2"}, {"SourceCharacterCodePoint": "0093", "TargetCharacterCodePoint": "01B3"}, {"SourceCharacterCodePoint": "0094", "TargetCharacterCodePoint": "01B4"}, {"SourceCharacterCodePoint": "0095", "TargetCharacterCodePoint": "01B5"}, {"SourceCharacterCodePoint": "0096", "TargetCharacterCodePoint": "01B6"}, {"SourceCharacterCodePoint": "0004", "TargetCharacterCodePoint": "01B7"}, {"SourceCharacterCodePoint": "0098", "TargetCharacterCodePoint": "01B8"}, {"SourceCharacterCodePoint": "0099", "TargetCharacterCodePoint": "01B9"}, {"SourceCharacterCodePoint": "009A", "TargetCharacterCodePoint": "01BA"}, {"SourceCharacterCodePoint": "009B", "TargetCharacterCodePoint": "01BB"}, {"SourceCharacterCodePoint": "0014", "TargetCharacterCodePoint": "01BC"}, {"SourceCharacterCodePoint": "0015", "TargetCharacterCodePoint": "01BD"}, {"SourceCharacterCodePoint": "009E", "TargetCharacterCodePoint": "01BE"}, {"SourceCharacterCodePoint": "001A", "TargetCharacterCodePoint": "01BF"}, {"SourceCharacterCodePoint": "009F", "TargetCharacterCodePoint": "027F"}]
```

### **nl-BE-500b and nl-BE-1148b**

Code Shift:

0180	0180
0001	0181
0002	0182
0003	0183
009C	0184
0009	0185
0086	0186
007F	0187

0097	0188
008D	0189
008E	018A
000B	018B
000C	018C
000D	018D
000E	018E
000F	018F
0010	0190
0011	0191
0012	0192
0013	0193
009D	0194
0085	0195
0008	0196
0087	0197
0018	0198
0019	0199
0092	019A
008F	019B
001C	019C
001D	019D
001E	019E
001F	019F
0080	01A0
0081	01A1
0082	01A2
0083	01A3
0084	01A4
000A	01A5
0017	01A6
001B	01A7
0088	01A8
0089	01A9
008A	01AA
008B	01AB
008C	01AC
0005	01AD
0006	01AE
0007	01AF
0090	01B0
0091	01B1
0016	01B2
0093	01B3
0094	01B4
0095	01B5
0096	01B6
0004	01B7
0098	01B8
0099	01B9
009A	01BA
009B	01BB
0014	01BC
0015	01BD
009E	01BE
001A	01BF
009F	027F

Corresponding input mapping for an AWS DMS task:

```
{"SourceCharacterCodePoint": "0180", "TargetCharacterCodePoint": "0180"},  
 {"SourceCharacterCodePoint": "0001", "TargetCharacterCodePoint": "0181"},  
 {"SourceCharacterCodePoint": "0002", "TargetCharacterCodePoint": "0182"},  
 {"SourceCharacterCodePoint": "0003", "TargetCharacterCodePoint": "0183"}
```



# Targets for data migration

AWS Database Migration Service (AWS DMS) can use many of the most popular databases as a target for data replication. The target can be on an Amazon Elastic Compute Cloud (Amazon EC2) instance, an Amazon Relational Database Service (Amazon RDS) instance, or an on-premises database.

For a comprehensive list of valid targets, see [Targets for AWS DMS \(p. 10\)](#).

**Note**

AWS DMS doesn't support migration across AWS Regions for the following target endpoint types:

- Amazon DynamoDB
- Amazon OpenSearch Service
- Amazon Kinesis Data Streams

**Topics**

- [Using an Oracle database as a target for AWS Database Migration Service \(p. 293\)](#)
- [Using a Microsoft SQL Server database as a target for AWS Database Migration Service \(p. 300\)](#)
- [Using a PostgreSQL database as a target for AWS Database Migration Service \(p. 304\)](#)
- [Using a MySQL-compatible database as a target for AWS Database Migration Service \(p. 311\)](#)
- [Using an Amazon Redshift database as a target for AWS Database Migration Service \(p. 315\)](#)
- [Using a SAP ASE database as a target for AWS Database Migration Service \(p. 330\)](#)
- [Using Amazon S3 as a target for AWS Database Migration Service \(p. 332\)](#)
- [Using an Amazon DynamoDB database as a target for AWS Database Migration Service \(p. 361\)](#)
- [Using Amazon Kinesis Data Streams as a target for AWS Database Migration Service \(p. 376\)](#)
- [Using Apache Kafka as a target for AWS Database Migration Service \(p. 387\)](#)
- [Using an Amazon OpenSearch Service cluster as a target for AWS Database Migration Service \(p. 404\)](#)
- [Using Amazon DocumentDB as a target for AWS Database Migration Service \(p. 408\)](#)
- [Using Amazon Neptune as a target for AWS Database Migration Service \(p. 415\)](#)
- [Using Redis as a target for AWS Database Migration Service \(p. 426\)](#)
- [Using Babelfish as a target for AWS Database Migration Service \(p. 431\)](#)

## Using an Oracle database as a target for AWS Database Migration Service

You can migrate data to Oracle database targets using AWS DMS, either from another Oracle database or from one of the other supported databases. You can use Secure Sockets Layer (SSL) to encrypt connections between your Oracle endpoint and the replication instance. For more information on using SSL with an Oracle endpoint, see [Using SSL with AWS Database Migration Service \(p. 647\)](#). AWS DMS also supports the use of Oracle transparent data encryption (TDE) to encrypt data at rest in the target database because Oracle TDE does not require an encryption key or password to write to the database.

AWS DMS supports Oracle versions 10g, 11g, 12c, 18c, and 19c for on-premises and EC2 instances for the Enterprise, Standard, Standard One, and Standard Two editions as targets. AWS DMS supports Oracle versions 11g (version 11.2.0.3.v1 and later), 12c, 18c, and 19c for Amazon RDS instance databases for the Enterprise, Standard, Standard One, and Standard Two editions.

When you use Oracle as a target, we assume that the data is to be migrated into the schema or user that is used for the target connection. If you want to migrate data to a different schema, use a schema transformation to do so. For example, suppose that your target endpoint connects to the user RDSMASTER and you want to migrate from the user PERFDATA1 to PERFDATA2. In this case, create a transformation like the following.

```
{  
    "rule-type": "transformation",  
    "rule-id": "2",  
    "rule-name": "2",  
    "rule-action": "rename",  
    "rule-target": "schema",  
    "object-locator": {  
        "schema-name": "PERFDATA1"  
    },  
    "value": "PERFDATA2"  
}
```

When using Oracle as a target, AWS DMS migrates all tables and indexes to default table and index tablespaces in the target. If you want to migrate tables and indexes to different table and index tablespaces, use a tablespace transformation to do so. For example, suppose that you have a set of tables in the INVENTORY schema assigned to some tablespaces in the Oracle source. For the migration, you want to assign all of these tables to a single INVENTORYSPACE tablespace in the target. In this case, create a transformation like the following.

```
{  
    "rule-type": "transformation",  
    "rule-id": "3",  
    "rule-name": "3",  
    "rule-action": "rename",  
    "rule-target": "table-tablespace",  
    "object-locator": {  
        "schema-name": "INVENTORY",  
        "table-name": "%",  
        "table-tablespace-name": "%"  
    },  
    "value": "INVENTORYSPACE"  
}
```

For more information about transformations, see [Specifying table selection and transformations rules using JSON \(p. 494\)](#).

If Oracle is both source and target, you can preserve existing table or index tablespace assignments by setting the Oracle source extra connection attribute, enableHomogenousTablespace=true. For more information, see [Extra connection attributes when using Oracle as a source for AWS DMS \(p. 165\)](#)

For additional details on working with Oracle databases as a target for AWS DMS, see the following sections:

### Topics

- [Limitations on Oracle as a target for AWS Database Migration Service \(p. 295\)](#)
- [User account privileges required for using Oracle as a target \(p. 295\)](#)
- [Configuring an Oracle database as a target for AWS Database Migration Service \(p. 297\)](#)
- [Extra connection attributes when using Oracle as a target for AWS DMS \(p. 297\)](#)
- [Target data types for Oracle \(p. 298\)](#)

## Limitations on Oracle as a target for AWS Database Migration Service

Limitations when using Oracle as a target for data migration include the following:

- AWS DMS doesn't create schema on the target Oracle database. You have to create any schemas you want on the target Oracle database. The schema name must already exist for the Oracle target. Tables from source schema are imported to the user or schema, which AWS DMS uses to connect to the target instance. To migrate multiple schemas, you can create multiple replication tasks. You can also migrate data to different schemas on a target. To do this, you need to use schema transformation rules on the AWS DMS table mappings.
- AWS DMS doesn't support the `Use direct path full load` option for tables with `INDEXTYPE CONTEXT`. As a workaround, you can use array load.
- With the batch optimized apply option, loading into the net changes table uses a direct path, which doesn't support XML type. As a workaround, you can use transactional apply mode.
- Empty strings migrated from source databases can be treated differently by the Oracle target (converted to one-space strings, for example). This can result in AWS DMS validation reporting a mismatch.
- You can express the total number of columns per table supported in Batch optimized apply mode, using the following formula:

```
2 * columns_in_original_table + columns_in_primary_key <= 999
```

For example, if the original table has 25 columns and its Primary Key consists of 5 columns, then the total number of columns is 55. If a table exceeds the supported number of columns, then all of the changes are applied in one-by-one mode.

- AWS DMS doesn't support Autonomous DB on Oracle Cloud Infrastructure (OCI).

## User account privileges required for using Oracle as a target

To use an Oracle target in an AWS Database Migration Service task, grant the following privileges in the Oracle database. You grant these to the user account specified in the Oracle database definitions for AWS DMS.

- `SELECT ANY TRANSACTION`
- `SELECT on V$NLS_PARAMETERS`
- `SELECT on V$TIMEZONE_NAMES`
- `SELECT on ALL_INDEXES`
- `SELECT on ALL_OBJECTS`
- `SELECT on DBA_OBJECTS`
- `SELECT on ALL_TABLES`
- `SELECT on ALL_USERS`
- `SELECT on ALL_CATALOG`
- `SELECT on ALL_CONSTRAINTS`
- `SELECT on ALL_CONS_COLUMNS`
- `SELECT on ALL_TAB_COLS`
- `SELECT on ALL_IND_COLUMNS`
- `DROP ANY TABLE`
- `SELECT ANY TABLE`

- INSERT ANY TABLE
- UPDATE ANY TABLE
- CREATE ANY VIEW
- DROP ANY VIEW
- CREATE ANY PROCEDURE
- ALTER ANY PROCEDURE
- DROP ANY PROCEDURE
- CREATE ANY SEQUENCE
- ALTER ANY SEQUENCE
- DROP ANY SEQUENCE
- DELETE ANY TABLE

For the following requirements, grant these additional privileges:

- To use a specific table list, grant SELECT on any replicated table and also ALTER on any replicated table.
- To allow a user to create a table in a default tablespace, grant the privilege GRANT UNLIMITED TABLESPACE.
- For logon, grant the privilege CREATE SESSION.
- If you are using a direct path (which is the default for full load), GRANT LOCK ANY TABLE to *dms\_user*;
- If schema is different when using "DROP and CREATE" table prep mode, GRANT CREATE ANY INDEX to *dms\_user*;
- For some full load scenarios, you might choose the "DROP and CREATE table" or "TRUNCATE before loading" option where a target table schema is different from the DMS user's. In this case, grant DROP ANY TABLE.
- To store changes in change tables or an audit table where the target table schema is different from the DMS user's, grant CREATE ANY TABLE and CREATE ANY INDEX.

## Read privileges required for AWS Database Migration Service on the target database

The AWS DMS user account must be granted read permissions for the following DBA tables:

- SELECT on DBA\_USERS
- SELECT on DBA\_TAB\_PRIVS
- SELECT on DBA\_OBJECTS
- SELECT on DBA\_SYNONYMS
- SELECT on DBA\_SEQUENCES
- SELECT on DBA\_TYPES
- SELECT on DBA\_INDEXES
- SELECT on DBA\_TABLES
- SELECT on DBA\_TRIGGERS
- SELECT on SYS.DBA\_REGISTRY

If any of the required privileges cannot be granted to V\$xxx, then grant them to V\_\$xxx.

## Configuring an Oracle database as a target for AWS Database Migration Service

Before using an Oracle database as a data migration target, you must provide an Oracle user account to AWS DMS. The user account must have read/write privileges on the Oracle database, as specified in [User account privileges required for using Oracle as a target \(p. 295\)](#).

### Extra connection attributes when using Oracle as a target for AWS DMS

You can use extra connection attributes to configure your Oracle target. You specify these settings when you create the target endpoint. If you have multiple connection attribute settings, separate them from each other by semicolons with no additional white space.

The following table shows the extra connection attributes available when using Oracle as a target.

Name	Description
escapeCharacter	<p>Set this attribute to an escape character. This escape character allows you to make a single wildcard character behave as a normal character in table mapping expressions. For more information, see <a href="#">Wildcards in table mapping (p. 499)</a>.</p> <p>Default value: Null</p> <p>Valid values: Any character other than a wildcard character</p> <p>Example: escapeCharacter=#</p>
useDirectPathFullLoad	<p>When set to Y, AWS DMS uses a direct path full load. Specify this value to enable the direct path protocol in the Oracle Call Interface (OCI). This OCI protocol enables the bulk loading of Oracle target tables during a full load.</p> <p>Default value: true</p> <p>Valid values: true/false</p> <p>Example: useDirectPathFullLoad=false;</p>
directPathParallelLoad	<p>When set to true, this attribute specifies a parallel load when useDirectPathFullLoad is set to Y. This attribute also only applies when you use the AWS DMS parallel load feature. For more information, see the description of the parallel-load operation in <a href="#">Table and collection settings rules and operations (p. 523)</a>.</p> <p>A limitation on specifying this parallel load setting is that the target table cannot have any constraints or indexes. For more information on this limitation, see <a href="#">Enabling Constraints After a Parallel Direct Path Load</a>. If constraints or indexes are enabled, setting this attribute to true has no effect.</p> <p>Default value: false</p>

Name	Description
	Valid values: true/false  Example: directPathParallelLoad=true;
directPathNoLog	When set to true, this attribute helps to increase the commit rate on the Oracle target database by writing directly to tables and not writing a trail to database logs. For more information, see <a href="#">Direct-Load INSERT</a> . This attribute also only applies when you set useDirectPathFullLoad to Y.  Default value: false  Valid values: true/false  Example: directPathNoLog=true;
charLengthSemantics	Specifies whether the length of a character column is in bytes or in characters. To indicate that the character column length is in characters, set this attribute to CHAR. Otherwise, the character column length is in bytes.  Default value: Not set to CHAR  Valid values: CHAR  Example: charLengthSemantics=CHAR;
alwaysReplaceEmptyString	AWS DMS adds an extra space to replicate an empty string when migrating to an Oracle target. In general, Oracle doesn't have a notation for an empty string. When you insert an empty string on varchar2, you load empty strings as NULL. If you want to insert the data as NULL on Oracle, set this attribute to FALSE.  Default value: true  Valid values: true/false  Example: alwaysReplaceEmptyString=false;

## Target data types for Oracle

A target Oracle database used with AWS DMS supports most Oracle data types. The following table shows the Oracle target data types that are supported when using AWS DMS and the default mapping from AWS DMS data types. For more information about how to view the data type that is mapped from the source, see the section for the source you are using.

AWS DMS data type	Oracle data type
BOOLEAN	NUMBER (1)
BYTES	RAW (length)
DATE	DATETIME

AWS DMS data type	Oracle data type
TIME	TIMESTAMP (0)
DATETIME	TIMESTAMP (scale)
INT1	NUMBER (3)
INT2	NUMBER (5)
INT4	NUMBER (10)
INT8	NUMBER (19)
NUMERIC	NUMBER (p,s)
REAL4	FLOAT
REAL8	FLOAT
STRING	With date indication: DATE With time indication: TIMESTAMP With timestamp indication: TIMESTAMP With timestamp_with_timezone indication: TIMESTAMP WITH TIMEZONE With timestamp_with_local_timezone indication: TIMESTAMP WITH LOCAL TIMEZONE With interval_year_to_month indication: INTERVAL YEAR TO MONTH With interval_day_to_second indication: INTERVAL DAY TO SECOND If length > 4000: CLOB In all other cases: VARCHAR2 (length)
UINT1	NUMBER (3)
UINT2	NUMBER (5)
UINT4	NUMBER (10)
UINT8	NUMBER (19)
WSTRING	If length > 2000: NCLOB In all other cases: NVARCHAR2 (length)
BLOB	BLOB To use this data type with AWS DMS, you must enable the use of BLOBS for a specific task. BLOB data types are supported only in tables that include a primary key

AWS DMS data type	Oracle data type
CLOB	<p>CLOB</p> <p>To use this data type with AWS DMS, you must enable the use of CLOBs for a specific task. During change data capture (CDC), CLOB data types are supported only in tables that include a primary key.</p> <p>STRING</p> <p>An Oracle VARCHAR2 data type on the source with a declared size greater than 4000 bytes maps through the AWS DMS CLOB to a STRING on the Oracle target.</p>
NCLOB	<p>NCLOB</p> <p>To use this data type with AWS DMS, you must enable the use of NCLOBs for a specific task. During CDC, NCLOB data types are supported only in tables that include a primary key.</p> <p>WSTRING</p> <p>An Oracle VARCHAR2 data type on the source with a declared size greater than 4000 bytes maps through the AWS DMS NCLOB to a WSTRING on the Oracle target.</p>
XMLTYPE	<p>The XMLTYPE target data type is only relevant in Oracle-to-Oracle replication tasks.</p> <p>When the source database is Oracle, the source data types are replicated as-is to the Oracle target. For example, an XMLTYPE data type on the source is created as an XMLTYPE data type on the target.</p>

## Using a Microsoft SQL Server database as a target for AWS Database Migration Service

You can migrate data to Microsoft SQL Server databases using AWS DMS. With an SQL Server database as a target, you can migrate data from either another SQL Server database or one of the other supported databases.

For on-premises and Amazon EC2 instance databases, AWS DMS supports as a target SQL Server versions 2005, 2008, 2008R2, 2012, 2014, 2016, 2017, and 2019. The Enterprise, Standard, Workgroup, Developer, and Web editions are supported by AWS DMS.

For Amazon RDS instance databases, AWS DMS supports as a target SQL Server versions 2008R2, 2012, 2014, 2016, 2017, and 2019. The Enterprise, Standard, Workgroup, Developer, and Web editions are supported by AWS DMS.

For additional details on working with AWS DMS and SQL Server target databases, see the following.

### Limitations on using SQL Server as a target for AWS Database Migration Service

The following limitations apply when using a SQL Server database as a target for AWS DMS:

- When you manually create a SQL Server target table with a computed column, full load replication is not supported when using the BCP bulk-copy utility. To use full load replication, disable BCP loading by setting the extra connection attribute (ECA) 'useBCPFullLoad=false' on the endpoint. For information about setting ECAs on endpoints, see [Creating source and target endpoints \(p. 131\)](#). For more information on working with BCP, see the [Microsoft SQL Server documentation](#).
- When replicating tables with SQL Server spatial data types (GEOMETRY and GEOGRAPHY), AWS DMS replaces any spatial reference identifier (SRID) that you might have inserted with the default SRID. The default SRID is 0 for GEOMETRY and 4326 for GEOGRAPHY.
- Temporal tables are not supported. Migrating temporal tables may work with a replication-only task in transactional apply mode if those tables are manually created on the target.
- Currently, boolean data types in a PostgreSQL source are migrated to a SQLServer target as the bit data type with inconsistent values. As a workaround, precreate the table with a VARCHAR(1) data type for the column (or let AWS DMS create the table). Then have downstream processing treat an "F" as False and a "T" as True.
- AWS DMS doesn't support change processing to set column nullability (using the ALTER COLUMN [SET|DROP] NOT NULL clause with ALTER TABLE statements).

## Security requirements when using SQL Server as a target for AWS Database Migration Service

The following describes the security requirements for using AWS DMS with a Microsoft SQL Server target:

- The AWS DMS user account must have at least the db\_owner user role on the SQL Server database that you are connecting to.
- A SQL Server system administrator must provide this permission to all AWS DMS user accounts.

## Extra connection attributes when using SQL Server as a target for AWS DMS

You can use extra connection attributes to configure your SQL Server target. You specify these settings when you create the target endpoint. If you have multiple connection attribute settings, separate them from each other by semicolons with no additional white space.

The following table shows the extra connection attributes that you can use when SQL Server is the target.

Name	Description
useBCPFullLoad	<p>Use this to attribute to transfer data for full-load operations using BCP. When the target table contains an identity column that does not exist in the source table, you must disable the <b>use BCP for loading table</b> option.</p> <p>Default value: true</p> <p>Valid values: true/false</p> <p>Example:</p> <pre>{ "UseBCPFullLoad": false }</pre>

Name	Description
}	
controlTablesFileGroup	<p>Specify a filegroup for the AWS DMS internal tables. When the replication task starts, all the internal AWS DMS control tables (awsdms_apply_exception, awsdms_apply, awsdms_changes) are created on the specified filegroup.</p> <p>Default value: n/a</p> <p>Valid values: String</p> <p>Example: controlTablesFileGroup=filegroup1</p> <p>The following is an example of a command for creating a filegroup.</p> <pre>ALTER DATABASE replicate ADD FILEGROUP Test1FG1; GO ALTER DATABASE replicate ADD FILE (     NAME = test1dat5,     FILENAME = 'C:\temp\DATA\t1dat5.ndf',     SIZE = 5MB,     MAXSIZE = 100MB,     FILEGROWTH = 5MB ) TO FILEGROUP Test1FG1; GO</pre>

## Target data types for Microsoft SQL Server

The following table shows the Microsoft SQL Server target data types that are supported when using AWS DMS and the default mapping from AWS DMS data types. For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

AWS DMS data type	SQL Server data type
BOOLEAN	TINYINT
BYTES	VARBINARY(length)
DATE	<p>For SQL Server 2008 and later, use DATE.</p> <p>For earlier versions, if the scale is 3 or less use DATETIME. In all other cases, use VARCHAR (37).</p>
TIME	<p>For SQL Server 2008 and later, use DATETIME2 (%d).</p> <p>For earlier versions, if the scale is 3 or less use DATETIME. In all other cases, use VARCHAR (37).</p>
DATETIME	<p>For SQL Server 2008 and later, use DATETIME2 (scale).</p> <p>For earlier versions, if the scale is 3 or less use DATETIME. In all other cases, use VARCHAR (37).</p>

AWS DMS data type	SQL Server data type
INT1	SMALLINT
INT2	SMALLINT
INT4	INT
INT8	BIGINT
NUMERIC	NUMERIC (p,s)
REAL4	REAL
REAL8	FLOAT
STRING	<p>If the column is a date or time column, then do the following:</p> <ul style="list-style-type: none"> <li>For SQL Server 2008 and later, use DATETIME2.</li> <li>For earlier versions, if the scale is 3 or less use DATETIME. In all other cases, use VARCHAR (37).</li> </ul> <p>If the column is not a date or time column, use VARCHAR (length).</p>
UINT1	TINYINT
UINT2	SMALLINT
UINT4	INT
UINT8	BIGINT
WSTRING	NVARCHAR (length)
BLOB	<p>VARBINARY(max)</p> <p>IMAGE</p> <p>To use this data type with AWS DMS, you must enable the use of BLOBs for a specific task. AWS DMS supports BLOB data types only in tables that include a primary key.</p>
CLOB	<p>VARCHAR(max)</p> <p>To use this data type with AWS DMS, you must enable the use of CLOBs for a specific task. During change data capture (CDC), AWS DMS supports CLOB data types only in tables that include a primary key.</p>
NCLOB	<p>NVARCHAR(max)</p> <p>To use this data type with AWS DMS, you must enable the use of NCLOBs for a specific task. During CDC, AWS DMS supports NCLOB data types only in tables that include a primary key.</p>

# Using a PostgreSQL database as a target for AWS Database Migration Service

You can migrate data to PostgreSQL databases using AWS DMS, either from another PostgreSQL database or from one of the other supported databases. AWS DMS supports a PostgreSQL version 9.4 and later (for versions 9.x), 10.x, 11.x, 12.x, 13.x, and 14.x database as a target for these types of databases:

- On-premises databases
- Databases on an EC2 instance
- Databases on an Amazon RDS DB instance
- Databases on an Amazon Aurora DB instance with PostgreSQL compatibility

## Note

- PostgreSQL 14.x requires AWS DMS 3.4.7 and later.
- Amazon Aurora Serverless is available as a TARGET for Amazon Aurora with PostgreSQL compatibility. For more information about Amazon Aurora Serverless, see [Using Amazon Aurora Serverless](#) in the [Amazon Aurora User Guide](#).
- Aurora Serverless DB clusters are accessible only from an Amazon VPC and can't use a [public IP address](#). So, if you intend to have a replication instance in a different region than Aurora PostgreSQL Serverless, you must configure [vpc peering](#). Otherwise, check the availability of Aurora PostgreSQL Serverless [regions](#), and decide to use one of those regions for both Aurora PostgreSQL Serverless and your replication instance.
- Babelfish capability is built into Amazon Aurora and doesn't have an additional cost. For more information, see [Using Babelfish for Aurora PostgreSQL as a target for AWS Database Migration Service \(p. 308\)](#).

AWS DMS takes a table-by-table approach when migrating data from source to target in the Full Load phase. Table order during the full load phase cannot be guaranteed. Tables are out of sync during the full load phase and while cached transactions for individual tables are being applied. As a result, active referential integrity constraints can result in task failure during the full load phase.

In PostgreSQL, foreign keys (referential integrity constraints) are implemented using triggers. During the full load phase, AWS DMS loads each table one at a time. We strongly recommend that you disable foreign key constraints during a full load, using one of the following methods:

- Temporarily disable all triggers from the instance, and finish the full load.
- Use the `session_replication_role` parameter in PostgreSQL.

At any given time, a trigger can be in one of the following states: `origin`, `replica`, `always`, or `disabled`. When the `session_replication_role` parameter is set to `replica`, only triggers in the `replica` state are active, and they are fired when they are called. Otherwise, the triggers remain inactive.

PostgreSQL has a failsafe mechanism to prevent a table from being truncated, even when `session_replication_role` is set. You can use this as an alternative to disabling triggers, to help the full load run to completion. To do this, set the target table preparation mode to `DO NOTHING`. Otherwise, `DROP` and `TRUNCATE` operations fail when there are foreign key constraints.

In Amazon RDS, you can control set this parameter using a parameter group. For a PostgreSQL instance running on Amazon EC2, you can set the parameter directly.

For additional details on working with a PostgreSQL database as a target for AWS DMS, see the following sections:

#### Topics

- [Limitations on using PostgreSQL as a target for AWS Database Migration Service \(p. 305\)](#)
- [Security requirements when using a PostgreSQL database as a target for AWS Database Migration Service \(p. 305\)](#)
- [Extra connection attributes when using PostgreSQL as a target for AWS DMS \(p. 306\)](#)
- [Target data types for PostgreSQL \(p. 307\)](#)
- [Using Babelfish for Aurora PostgreSQL as a target for AWS Database Migration Service \(p. 308\)](#)

## Limitations on using PostgreSQL as a target for AWS Database Migration Service

The following limitations apply when using a PostgreSQL database as a target for AWS DMS:

- For heterogeneous migrations, the JSON data type is converted to the Native CLOB data type internally.
- In an Oracle to PostgreSQL migration, if a column in Oracle contains a NULL character (hex value U+0000), AWS DMS converts the NULL character to a space (hex value U+0020). This is due to a PostgreSQL limitation.
- AWS DMS doesn't support replication to a table with a unique index created with coalesce function.

## Security requirements when using a PostgreSQL database as a target for AWS Database Migration Service

For security purposes, the user account used for the data migration must be a registered user in any PostgreSQL database that you use as a target.

Your PostgreSQL target endpoint requires minimum user permissions to run an AWS DMS migration, see the following examples.

```
CREATE USER newuser WITH PASSWORD 'your-password';
ALTER SCHEMA schema_name OWNER TO newuser;
```

Or,

```
GRANT USAGE ON SCHEMA schema_name TO myuser;
GRANT CONNECT ON DATABASE postgres TO myuser;
GRANT CREATE ON DATABASE postgres TO myuser;
GRANT CREATE ON SCHEMA schema_name TO myuser;
GRANT UPDATE, INSERT, SELECT, DELETE, TRUNCATE ON ALL TABLES IN SCHEMA schema_name TO
myuser;
GRANT TRUNCATE ON schema_name."BasicFeed" TO myuser;
```

## Extra connection attributes when using PostgreSQL as a target for AWS DMS

You can use extra connection attributes to configure your PostgreSQL target. You specify these settings when you create the target endpoint. If you have multiple connection attribute settings, separate them from each other by semicolons with no additional white space.

The following table shows the extra connection attributes you can use to configure PostgreSQL as a target for AWS DMS.

Name	Description
maxFileSize	<p>Specifies the maximum size (in KB) of any .csv file used to transfer data to PostgreSQL.</p> <p>Default value: 32,768 KB (32 MB)</p> <p>Valid values: 1–1,048,576 KB (up to 1.1 GB)</p> <p>Example: maxFileSize=512</p>
executeTimeout	<p>Sets the client statement timeout for the PostgreSQL instance, in seconds. The default value is 60 seconds.</p> <p>Example: executeTimeout=100</p>
afterConnectScript=SET session_replication_role='replica'	<p>For use with change data capture (CDC) only, this attribute bypasses AWS DMS bypass foreign keys and user triggers to reduce the time it takes to bulk load data.</p> <p><b>Note</b> This attribute is only effective in change data capture mode.</p>
mapUnboundedNumericAsString	<p>This parameter treats columns with unbounded NUMERIC data types as STRING in order to successfully migrate without losing precision of the numeric value. Use this parameter only for replication from PostgreSQL source to PostgreSQL target, or databases with PostgreSQL compatibility.</p> <p>Default value: false</p> <p>Valid values: false/true</p> <p>Example: mapUnboundedNumericAsString=true;</p> <p>Using this parameter might result in some replication performance degradation because of transformation from numeric to string and back to numeric. This parameter is supported for use by DMS version 3.4.4 and higher</p> <p><b>Note</b> Only use mapUnboundedNumericAsString in PostgreSQL source and target endpoints together. Use of mapUnboundedNumericAsString on source PostgreSQL endpoints restricts precision to 28 during CDC. Use of</p>

Name	Description
	mapUnboundedNumericAsString on target endpoints, migrates data with Precision 28 Scale 6. Do not use mapUnboundedNumericAsString with non-PostgreSQL targets.

## Target data types for PostgreSQL

The PostgreSQL database endpoint for AWS DMS supports most PostgreSQL database data types. The following table shows the PostgreSQL database target data types that are supported when using AWS DMS and the default mapping from AWS DMS data types.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

AWS DMS data type	PostgreSQL data type
BOOLEAN	BOOLEAN
BLOB	BYTEA
BYTES	BYTEA
DATE	DATE
TIME	TIME
DATETIME	If the scale is from 0 through 6, then use TIMESTAMP. If the scale is from 7 through 9, then use VARCHAR (37).
INT1	SMALLINT
INT2	SMALLINT
INT4	INTEGER
INT8	BIGINT
NUMERIC	DECIMAL (P,S)
REAL4	FLOAT4
REAL8	FLOAT8
STRING	If the length is from 1 through 21,845, then use VARCHAR (length in bytes). If the length is 21,846 through 2,147,483,647, then use VARCHAR (65535).
UINT1	SMALLINT
UINT2	INTEGER
UINT4	BIGINT
UINT8	BIGINT

AWS DMS data type	PostgreSQL data type
WSTRING	If the length is from 1 through 21,845, then use VARCHAR (length in bytes).  If the length is 21,846 through 2,147,483,647, then use VARCHAR (65535).
NCLOB	TEXT
CLOB	TEXT

**Note**

When replicating from a PostgreSQL source, AWS DMS creates the target table with the same data types for all columns, apart from columns with user-defined data types. In such cases, the data type is created as "character varying" in the target.

## Using Babelfish for Aurora PostgreSQL as a target for AWS Database Migration Service

You can migrate SQL Server source tables to a Babelfish for Amazon Aurora PostgreSQL target using AWS Database Migration Service. With Babelfish, Aurora PostgreSQL understands T-SQL, Microsoft SQL Server's proprietary SQL dialect, and supports the same communications protocol. So, applications written for SQL Server can now work with Aurora with fewer code changes. Babelfish capability is built into Amazon Aurora and doesn't have an additional cost. You can activate Babelfish on your Amazon Aurora cluster from the Amazon RDS console.

When you create your AWS DMS target endpoint using the AWS DMS console, API, or CLI commands, specify the target engine as **Amazon Aurora PostgreSQL**, and name the database, **babelfish\_db**.

### Adding transformation rules to your migration task

When you define a migration task for a Babelfish target, you need to include transformation rules that ensure DMS uses the pre-created T-SQL Babelfish tables in the target database.

First, add a transformation rule to your migration task that makes all table names lowercase. Babelfish stores as lowercase in the PostgreSQL pg\_class catalog the names of tables that you create using T-SQL. However, when you have SQL Server tables with mixed-case names, DMS creates the tables using PostgreSQL native data types instead of the T-SQL compatible data types. For that reason, be sure to add a transformation rule that makes all table names lowercase.

Next, if you used the multidatabase migration mode when you defined your cluster, add a transformation rule that renames the original SQL Server schema. Make sure to rename the SQL Server schema name to include the name of the T-SQL database. For example, if the original SQL Server schema name is dbo, and your T-SQL database name is mydb, rename the schema to mydb\_dbo using a transformation rule.

If you use single database mode, you don't need a transformation rule to rename schema names. Schema names have a one-to-one mapping with the target T-SQL database in Babelfish.

The following sample transformation rule makes all table names lowercase, and renames the original SQL Server schema name from dbo to mydb\_dbo.

```
{
  "rules": [
    {
      "rule-type": "transformation",
      "rule-id": "lowercase-table-names",
      "target": "mydb_dbo"
    }
  ]
}
```

```

    "rule-id": "566251737",
    "rule-name": "566251737",
    "rule-target": "schema",
    "object-locator": {
        "schema-name": "dbo"
    },
    "rule-action": "rename",
    "value": "mydb_dbo",
    "old-value": null
},
{
    "rule-type": "transformation",
    "rule-id": "566139410",
    "rule-name": "566139410",
    "rule-target": "table",
    "object-locator": {
        "schema-name": "%",
        "table-name": "%"
    },
    "rule-action": "convert-lowercase",
    "value": null,
    "old-value": null
},
{
    "rule-type": "selection",
    "rule-id": "566111704",
    "rule-name": "566111704",
    "object-locator": {
        "schema-name": "dbo",
        "table-name": "%"
    },
    "rule-action": "include",
    "filters": []
}
]
}

```

## Limitations to using a PostgreSQL target endpoint with Babelfish tables

The following limitations apply when using a PostgreSQL target endpoint with Babelfish tables:

- For **Target table preparation** mode, use only the **Do nothing** or **Truncate** modes. Don't use the **Drop tables on target** mode. In that mode, DMS creates the tables as PostgreSQL tables that T-SQL might not recognize.
- Babelfish only supports migrating BINARY, VARBINARY, and IMAGE data types with Aurora PostgreSQL version 14.3 and later, using the BYTEA data type. For earlier versions of Aurora PostgreSQL, you can use DMS to migrate these tables to a [Babelfish target endpoint \(p. 431\)](#). You don't have to specify a length for the BYTEA data type, as shown in the example following.

```
[Picture] [VARBINARY](max) NULL
```

The preceding becomes the following.

```
[Picture] BYTEA NULL
```

- If you create a migration task for ongoing replication from SQL Server to Babelfish using the PostgreSQL target endpoint, you need to assign the SERIAL data type to any tables that use IDENTITY columns. For more information, see [SERIAL Usage](#) in the Sequences and Identity section of the [SQL Server to Aurora PostgreSQL Migration Playbook](#). Then, when you create the table in Babelfish, change the column definition from the following.

```
[IDCol] [INT] IDENTITY(1,1) NOT NULL PRIMARY KEY
```

Change the preceding into the following.

```
[IDCol] SERIAL PRIMARY KEY
```

Babelfish-compatible Aurora PostgreSQL creates a sequence using the default configuration and adds a NOT NULL constraint to the column. The newly created sequence behaves like a regular sequence (incremented by 1) and has no composite SERIAL option.

- After migrating data with tables that use IDENTITY columns or the SERIAL data type, reset the PostgreSQL-based sequence object based on the maximum value for the column. After performing a full load of the tables, use the following T-SQL query to generate statements to seed the associated sequence object.

```
DECLARE @schema_prefix NVARCHAR(200) = ''

IF current_setting('babelfishpg_tsquery.migration_mode') = 'multi-db'
    SET @schema_prefix = db_name() + '_'

SELECT 'SELECT setval(pg_get_serial_sequence(''' + @schema_prefix +
    schema_name(tables.schema_id) + '.' + tables.name + ''', ''' + columns.name + ''')
        ,(select max(' + columns.name + ') from ' + schema_name(tables.schema_id)
        + '.' + tables.name + '));'
FROM sys.tables tables
JOIN sys.columns columns ON tables.object_id = columns.object_id
WHERE columns.is_identity = 1

UNION ALL

SELECT 'SELECT setval(pg_get_serial_sequence(''' + @schema_prefix + table_schema + '.' +
    table_name + ''',
        ''' + column_name + '''),(select max(' + column_name + ') from ' + table_schema + '.' +
    table_name + '));'
FROM information_schema.columns
WHERE column_default LIKE 'nextval(%');
```

The query generates a series of SELECT statements that you execute in order to update the maximum IDENTITY and SERIAL values.

- In some cases, **Full LOB mode** might result in a table error. If that happens, create a separate task for the tables that failed to load. Then use **Limited LOB mode** to specify the appropriate value for the **Maximum LOB size (KB)**.
- Performing data validation with Babelfish tables that don't use integer based primary keys generates a message that a suitable unique key can't be found.
- Because of precision differences in the number of decimal places for seconds, DMS reports data validation failures for Babelfish tables that use DATETIME data types. To suppress those failures, you can add the following validation rule type for DATETIME data types.

```
{
    "rule-type": "validation",
    "rule-id": "3",
    "rule-name": "3",
    "rule-target": "column",
    "object-locator": {
        "schema-name": "dbo",
        "table-name": "%",
        "column-name": "%",
        "data-type": "datetime"
}
```

```
        },
        "rule-action": "override-validation-function",
        "source-function": "case when ${column-name} is NULL then NULL else 0 end",
        "target-function": "case when ${column-name} is NULL then NULL else 0 end"
    }
```

## Using a MySQL-compatible database as a target for AWS Database Migration Service

You can migrate data to any MySQL-compatible database using AWS DMS, from any of the source data engines that AWS DMS supports. If you are migrating to an on-premises MySQL-compatible database, then AWS DMS requires that your source engine reside within the AWS ecosystem. The engine can be on an AWS-managed service such as Amazon RDS, Amazon Aurora, or Amazon S3. Or the engine can be on a self-managed database on Amazon EC2.

You can use SSL to encrypt connections between your MySQL-compatible endpoint and the replication instance. For more information on using SSL with a MySQL-compatible endpoint, see [Using SSL with AWS Database Migration Service \(p. 647\)](#).

AWS DMS supports versions 5.5, 5.6, 5.7, and 8.0 of MySQL and Aurora MySQL. In addition, AWS DMS supports MariaDB versions 10.0.24 to 10.0.28, 10.1, 10.2, 10.3, 10.4, and 10.5.

You can use the following MySQL-compatible databases as targets for AWS DMS:

- MySQL Community Edition
- MySQL Standard Edition
- MySQL Enterprise Edition
- MySQL Cluster Carrier Grade Edition
- MariaDB Community Edition
- MariaDB Enterprise Edition
- MariaDB Column Store
- Amazon Aurora MySQL

### Note

Regardless of the source storage engine (MyISAM, MEMORY, and so on), AWS DMS creates a MySQL-compatible target table as an InnoDB table by default.

If you need a table in a storage engine other than InnoDB, you can manually create the table on the MySQL-compatible target and migrate the table using the **Do nothing** option. For more information, see [Full-load task settings \(p. 454\)](#).

For additional details on working with a MySQL-compatible database as a target for AWS DMS, see the following sections.

### Topics

- [Using any MySQL-compatible database as a target for AWS Database Migration Service \(p. 312\)](#)
- [Limitations on using a MySQL-compatible database as a target for AWS Database Migration Service \(p. 312\)](#)
- [Extra connection attributes when using a MySQL-compatible database as a target for AWS DMS \(p. 313\)](#)
- [Target data types for MySQL \(p. 314\)](#)

## Using any MySQL-compatible database as a target for AWS Database Migration Service

Before you begin to work with a MySQL-compatible database as a target for AWS DMS, make sure that you have completed the following prerequisites:

- Provide a user account to AWS DMS that has read/write privileges to the MySQL-compatible database. To create the necessary privileges, run the following commands.

```
CREATE USER '<user acct>'@'%' IDENTIFIED BY '<user password>';
GRANT ALTER, CREATE, DROP, INDEX, INSERT, UPDATE, DELETE, SELECT ON <schema>.* TO
'<user acct>'@'%';
GRANT ALL PRIVILEGES ON awsdms_control.* TO '<user acct>'@'%';
```

- During the full-load migration phase, you must disable foreign keys on your target tables. To disable foreign key checks on a MySQL-compatible database during a full load, you can add the following command to the **Extra Connection Attributes** in the **Advanced** section of the target endpoint.

```
initstmt=SET FOREIGN_KEY_CHECKS=0
```

- Set the database parameter `local_infile = 1` to enable AWS DMS to load data into the target database.

## Limitations on using a MySQL-compatible database as a target for AWS Database Migration Service

When using a MySQL database as a target, AWS DMS doesn't support the following:

- The data definition language (DDL) statements TRUNCATE PARTITION, DROP TABLE, and RENAME TABLE.
- Using an ALTER TABLE `table_name` ADD COLUMN `column_name` statement to add columns to the beginning or the middle of a table.
- When loading data to a MySQL-compatible target in a full load task, AWS DMS doesn't report duplicate key errors in the task log. This is because of the way that MySQL handles CSV load data.
- When you update a column's value to its existing value, MySQL-compatible databases return a `0 rows affected` warning. Although this behavior isn't technically an error, it is different from how the situation is handled by other database engines. For example, Oracle performs an update of one row. For MySQL-compatible databases, AWS DMS generates an entry in the `awsdms_apply_exceptions` control table and logs the following warning.

```
Some changes from the source database had no impact when applied to
the target database. See awsdms_apply_exceptions table for details.
```

- Aurora Serverless is available as a target for Amazon Aurora version 1, compatible with MySQL version 5.6. Aurora Serverless is available as a target for Amazon Aurora version 2, compatible with MySQL version 5.7. (Select Aurora MySQL version 2.07.1 to be able to use Aurora Serverless with MySQL 5.7 compatibility.) For more information about Aurora Serverless, see [Using Amazon Aurora Serverless](#) in the [Amazon Aurora User Guide](#).
- Using an Aurora Reader endpoint. You can't use an Aurora for MySQL reader endpoint because the `read_only` parameter can't be changed. You can use a Amazon RDS for MySQL reader endpoint if the `read_only` parameter is set to 0.

## Extra connection attributes when using a MySQL-compatible database as a target for AWS DMS

You can use extra connection attributes to configure your MySQL-compatible target. You specify these settings when you create the target endpoint. If you have multiple connection attribute settings, separate them from each other by semicolons with no additional white space.

The following table shows extra configuration settings that you can use when creating a MySQL-compatible target for AWS DMS.

Name	Description
targetDbType	<p>Specifies where to migrate source tables on the target, either to a single database or multiple databases. If you specify SPECIFIC_DATABASE, you need to specify the database name, either when using the AWS CLI or the AWS Management Console.</p> <p>Default value: MULTIPLE DATABASES</p> <p>Valid values: {SPECIFIC_DATABASE, MULTIPLE DATABASES}</p> <p>Example: targetDbType=MULTIPLE DATABASES</p>
parallelLoadThreads	<p>Improves performance when loading data into the MySQL-compatible target database. Specifies how many threads to use to load the data into the MySQL-compatible target database. Setting a large number of threads can have an adverse effect on database performance, because a separate connection is required for each thread.</p> <p>Default value: 1</p> <p>Valid values: 1–5</p> <p>Example: parallelLoadThreads=1</p>
initstmt=SET FOREIGN_KEY_CHECKS=0	Disables foreign key checks.
initstmt=SET time_zone	<p>Specifies the time zone for the target MySQL-compatible database.</p> <p>Default value: UTC</p> <p>Valid values: The time zone names available in the target MySQL database.</p> <p>Example: initstmt=SET time_zone='US/Pacific'</p>
afterConnectScript=SET character_set_connection='latin1'	<p>Specifies that the MySQL-compatible target should translate received statements into the latin1 character set, which is the default compiled-in character set of the database. This parameter typically improves performance when converting from UTF8 clients.</p>
maxFileSize	Specifies the maximum size (in KB) of any .csv file used to transfer data to a MySQL-compatible database.

Name	Description
	<p>Default value: 32,768 KB (32 MB)</p> <p>Valid values: 1–1,048,576</p> <p>Example: <code>maxFileSize=512</code></p>
CleanSrcMetadataOnMismatch	<p>Cleans and recreates table metadata information on the replication instance when a mismatch occurs. An example is a situation where running an alter DDL statement on a table might result in different information about the table cached in the replication instance. Boolean.</p> <p>Default value: <code>false</code></p> <p>Example: <code>CleanSrcMetadataOnMismatch=false</code></p>

## Target data types for MySQL

The following table shows the MySQL database target data types that are supported when using AWS DMS and the default mapping from AWS DMS data types.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

AWS DMS data types	MySQL data types
BOOLEAN	BOOLEAN
BYTES	If the length is from 1 through 65,535, then use VARBINARY (length).  If the length is from 65,536 through 2,147,483,647, then use LONGLOB.
DATE	DATE
TIME	TIME
TIMESTAMP	"If scale is => 0 and =< 6, then: DATETIME (Scale)  If scale is => 7 and =< 9, then: VARCHAR (37)"
INT1	TINYINT
INT2	SMALLINT
INT4	INTEGER
INT8	BIGINT
NUMERIC	DECIMAL (p,s)
REAL4	FLOAT
REAL8	DOUBLE PRECISION
STRING	If the length is from 1 through 21,845, then use VARCHAR (length).

AWS DMS data types	MySQL data types
	If the length is from 21,846 through 2,147,483,647, then use LONGTEXT.
UINT1	UNSIGNED TINYINT
UINT2	UNSIGNED SMALLINT
UINT4	UNSIGNED INTEGER
UINT8	UNSIGNED BIGINT
WSTRING	If the length is from 1 through 32,767, then use VARCHAR (length).  If the length is from 32,768 through 2,147,483,647, then use LONGTEXT.
BLOB	If the length is from 1 through 65,535, then use BLOB.  If the length is from 65,536 through 2,147,483,647, then use LONGBLOB.  If the length is 0, then use LONGBLOB (full LOB support).
NCLOB	If the length is from 1 through 65,535, then use TEXT.  If the length is from 65,536 through 2,147,483,647, then use LONGTEXT with ucs2 for CHARACTER SET.  If the length is 0, then use LONGTEXT (full LOB support) with ucs2 for CHARACTER SET.
CLOB	If the length is from 1 through 65,535, then use TEXT.  If the length is from 65,536 through 2147483647, then use LONGTEXT.  If the length is 0, then use LONGTEXT (full LOB support).

## Using an Amazon Redshift database as a target for AWS Database Migration Service

You can migrate data to Amazon Redshift databases using AWS Database Migration Service. Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. With an Amazon Redshift database as a target, you can migrate data from all of the other supported source databases.

The Amazon Redshift cluster must be in the same AWS account and same AWS Region as the replication instance.

During a database migration to Amazon Redshift, AWS DMS first moves data to an Amazon S3 bucket. When the files reside in an Amazon S3 bucket, AWS DMS then transfers them to the proper tables in the Amazon Redshift data warehouse. AWS DMS creates the S3 bucket in the same AWS Region as the Amazon Redshift database. The AWS DMS replication instance must be located in that same AWS Region.

If you use the AWS CLI or DMS API to migrate data to Amazon Redshift, set up an AWS Identity and Access Management (IAM) role to allow S3 access. For more information about creating this IAM role, see [Creating the IAM roles to use with the AWS CLI and AWS DMS API \(p. 632\)](#).

The Amazon Redshift endpoint provides full automation for the following:

- Schema generation and data type mapping
- Full load of source database tables
- Incremental load of changes made to source tables
- Application of schema changes in data definition language (DDL) made to the source tables
- Synchronization between full load and change data capture (CDC) processes.

AWS Database Migration Service supports both full load and change processing operations. AWS DMS reads the data from the source database and creates a series of comma-separated value (.csv) files. For full-load operations, AWS DMS creates files for each table. AWS DMS then copies the table files for each table to a separate folder in Amazon S3. When the files are uploaded to Amazon S3, AWS DMS sends a copy command and the data in the files are copied into Amazon Redshift. For change-processing operations, AWS DMS copies the net changes to the .csv files. AWS DMS then uploads the net change files to Amazon S3 and copies the data to Amazon Redshift.

For additional details on working with Amazon Redshift as a target for AWS DMS, see the following sections:

#### Topics

- [Prerequisites for using an Amazon Redshift database as a target for AWS Database Migration Service \(p. 316\)](#)
- [Privileges required for using Redshift as a target \(p. 317\)](#)
- [Limitations on using Amazon Redshift as a target for AWS Database Migration Service \(p. 318\)](#)
- [Configuring an Amazon Redshift database as a target for AWS Database Migration Service \(p. 319\)](#)
- [Using enhanced VPC routing with an Amazon Redshift as a target for AWS Database Migration Service \(p. 319\)](#)
- [Creating and using AWS KMS keys to encrypt Amazon Redshift target data \(p. 319\)](#)
- [Endpoint settings when using Amazon Redshift as a target for AWS DMS \(p. 322\)](#)
- [Extra connection attributes when using Amazon Redshift as a target for AWS DMS \(p. 324\)](#)
- [Multithreaded task settings for Amazon Redshift \(p. 327\)](#)
- [Target data types for Amazon Redshift \(p. 329\)](#)

## Prerequisites for using an Amazon Redshift database as a target for AWS Database Migration Service

The following list describes the prerequisites necessary for working with Amazon Redshift as a target for data migration:

- Use the AWS Management Console to launch an Amazon Redshift cluster. Note the basic information about your AWS account and your Amazon Redshift cluster, such as your password, user name, and database name. You need these values when creating the Amazon Redshift target endpoint.

- The Amazon Redshift cluster must be in the same AWS account and the same AWS Region as the replication instance.
- The AWS DMS replication instance needs network connectivity to the Amazon Redshift endpoint (hostname and port) that your cluster uses.
- AWS DMS uses an Amazon S3 bucket to transfer data to the Amazon Redshift database. For AWS DMS to create the bucket, the console uses an IAM role, `dms-access-for-endpoint`. If you use the AWS CLI or DMS API to create a database migration with Amazon Redshift as the target database, you must create this IAM role. For more information about creating this role, see [Creating the IAM roles to use with the AWS CLI and AWS DMS API \(p. 632\)](#).
- AWS DMS converts BLOBS, CLOBS, and NCLOBs to a VARCHAR on the target Amazon Redshift instance. Amazon Redshift doesn't support VARCHAR data types larger than 64 KB, so you can't store traditional LOBs on Amazon Redshift.
- Set the target metadata task setting [BatchApplyEnabled \(p. 465\)](#) to true for AWS DMS to handle changes to Amazon Redshift target tables during CDC. A Primary Key on both the source and target table is required. Without a Primary Key, changes are applied statement by statement. And that can adversely affect task performance during CDC by causing target latency and impacting the cluster commit queue.

## Privileges required for using Redshift as a target

Use the GRANT command to define access privileges for a user or user group. Privileges include access options such as being able to read data in tables and views, write data, and create tables. For more information about using GRANT with Amazon Redshift, see [GRANT](#) in the *Amazon Redshift Database Developer Guide*.

The following is the syntax to give specific privileges for a table, database, schema, function, procedure, or language-level privileges on Amazon Redshift tables and views.

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | REFERENCES } [, ...] | ALL [ PRIVILEGES ] }
    ON { [ TABLE ] table_name [, ...] | ALL TABLES IN SCHEMA schema_name [, ...] }
    TO { username [ WITH GRANT OPTION ] | GROUP group_name | PUBLIC } [, ...]

GRANT { { CREATE | TEMPORARY | TEMP } [, ...] | ALL [ PRIVILEGES ] }
    ON DATABASE db_name [, ...]
    TO { username [ WITH GRANT OPTION ] | GROUP group_name | PUBLIC } [, ...]

GRANT { { CREATE | USAGE } [, ...] | ALL [ PRIVILEGES ] }
    ON SCHEMA schema_name [, ...]
    TO { username [ WITH GRANT OPTION ] | GROUP group_name | PUBLIC } [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { FUNCTION function_name ( [ [ argname ] argtype [, ...] ] ) [, ...] | ALL FUNCTIONS
        IN SCHEMA schema_name [, ...] }
    TO { username [ WITH GRANT OPTION ] | GROUP group_name | PUBLIC } [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { PROCEDURE procedure_name ( [ [ argname ] argtype [, ...] ] ) [, ...] | ALL
        PROCEDURES IN SCHEMA schema_name [, ...] }
    TO { username [ WITH GRANT OPTION ] | GROUP group_name | PUBLIC } [, ...]

GRANT USAGE
    ON LANGUAGE language_name [, ...]
    TO { username [ WITH GRANT OPTION ] | GROUP group_name | PUBLIC } [, ...]
```

The following is the syntax for column-level privileges on Amazon Redshift tables and views.

```
GRANT { { SELECT | UPDATE } ( column_name [, ...] ) [, ...] | ALL [ PRIVILEGES ]
    ( column_name [, ...] ) }
```

```
ON { [ TABLE ] table_name [, ...] }
TO { username | GROUP group_name | PUBLIC } [, ...]
```

The following is the syntax for the ASSUMEROLE privilege granted to users and groups with a specified role.

```
GRANT ASSUMEROLE
ON { 'iam_role' [, ...] | ALL }
TO { username | GROUP group_name | PUBLIC } [, ...]
FOR { ALL | COPY | UNLOAD } [, ...]
```

## Limitations on using Amazon Redshift as a target for AWS Database Migration Service

The following limitations apply when using an Amazon Redshift database as a target:

- Don't enable versioning for the S3 bucket you use as intermediate storage for your Amazon Redshift target. If you need S3 versioning, use lifecycle policies to actively delete old versions. Otherwise, you might encounter endpoint test connection failures because of an S3 list-object call timeout. To create a lifecycle policy for an S3 bucket, see [Managing your storage lifecycle](#). To delete a version of an S3 object, see [Deleting object versions from a versioning-enabled bucket](#).
- The following DDL is not supported:

```
ALTER TABLE table name MODIFY COLUMN column name data type;
```

- AWS DMS cannot migrate or replicate changes to a schema with a name that begins with underscore (\_). If you have schemas that have a name that begins with an underscore, use mapping transformations to rename the schema on the target.
- Amazon Redshift doesn't support VARCHARs larger than 64 KB. LOBs from traditional databases can't be stored in Amazon Redshift.
- Applying a DELETE statement to a table with a multi-column primary key is not supported when any of the primary key column names use a reserved word. Go [here](#) to see a list of Amazon Redshift reserved words.
- You may experience performance issues if your source system performs UPDATE operations on the primary key of a source table. These performance issues occur when applying changes to the target. This is because UPDATE (and DELETE) operations depend on the primary key value to identify the target row. If you update the primary key of a source table, your task log will contain messages like the following:

```
Update on table 1 changes PK to a PK that was previously updated in the same bulk update.
```

- DMS doesn't support custom DNS names when configuring an endpoint for a Redshift cluster, and you need to use the Amazon provided DNS name. Since the Amazon Redshift cluster must be in the same AWS account and Region as the replication instance, validation fails if you use a custom DNS endpoint.
- Amazon Redshift has a default 4-hour idle session timeout. When there isn't any activity within the DMS replication task, Redshift disconnects the session after 4 hours. Errors can result from DMS being unable to connect and potentially needing to restart. As a workaround, set a SESSION TIMEOUT limit greater than 4 hours for the DMS replication user. Or, see the description of [ALTER USER](#) in the *Amazon Redshift Database Developer Guide*.
- When AWS DMS replicates source table data without a primary or unique key, CDC latency might be high resulting in an unacceptable level of performance.

## Configuring an Amazon Redshift database as a target for AWS Database Migration Service

AWS Database Migration Service must be configured to work with the Amazon Redshift instance. The following table describes the configuration properties available for the Amazon Redshift endpoint.

Property	Description
server	The name of the Amazon Redshift cluster you are using.
port	The port number for Amazon Redshift. The default value is 5439.
username	An Amazon Redshift user name for a registered user.
password	The password for the user named in the username property.
database	The name of the Amazon Redshift data warehouse (service) you are working with.

If you want to add extra connection string attributes to your Amazon Redshift endpoint, you can specify the `maxFileSize` and `fileTransferUploadStreams` attributes. For more information on these attributes, see [Extra connection attributes when using Amazon Redshift as a target for AWS DMS \(p. 324\)](#).

## Using enhanced VPC routing with an Amazon Redshift as a target for AWS Database Migration Service

If you use Enhanced VPC Routing with your Amazon Redshift target, all COPY traffic between your Amazon Redshift cluster and your data repositories goes through your VPC. Because Enhanced VPC Routing affects the way that Amazon Redshift accesses other resources, COPY commands might fail if you haven't configured your VPC correctly.

AWS DMS can be affected by this behavior because it uses the COPY command to move data in S3 to an Amazon Redshift cluster.

Following are the steps AWS DMS takes to load data into an Amazon Redshift target:

1. AWS DMS copies data from the source to .csv files on the replication server.
2. AWS DMS uses the AWS SDK to copy the .csv files into an S3 bucket on your account.
3. AWS DMS then uses the COPY command in Amazon Redshift to copy data from the .csv files in S3 to an appropriate table in Amazon Redshift.

If Enhanced VPC Routing is not enabled, Amazon Redshift routes traffic through the internet, including traffic to other services within the AWS network. If the feature is not enabled, you do not have to configure the network path. If the feature is enabled, you must specifically create a network path between your cluster's VPC and your data resources. For more information on the configuration required, see [Enhanced VPC routing](#) in the Amazon Redshift documentation.

## Creating and using AWS KMS keys to encrypt Amazon Redshift target data

You can encrypt your target data pushed to Amazon S3 before it is copied to Amazon Redshift. To do so, you can create and use custom AWS KMS keys. You can use the key you created to encrypt your target data using one of the following mechanisms when you create the Amazon Redshift target endpoint:

- Use the following option when you run the `create-endpoint` command using the AWS CLI.

```
--redshift-settings '{"EncryptionMode": "SSE_KMS", "ServerSideEncryptionKmsKeyId": "your-kms-key-ARN"}'
```

Here, `your-kms-key-ARN` is the Amazon Resource Name (ARN) for your KMS key. For more information, see [Endpoint settings when using Amazon Redshift as a target for AWS DMS \(p. 322\)](#).

- Set the extra connection attribute `encryptionMode` to the value `SSE_KMS` and the extra connection attribute `serverSideEncryptionKmsKeyId` to the ARN for your KMS key. For more information, see [Extra connection attributes when using Amazon Redshift as a target for AWS DMS \(p. 324\)](#).

To encrypt Amazon Redshift target data using a KMS key, you need an AWS Identity and Access Management (IAM) role that has permissions to access Amazon Redshift data. This IAM role is then accessed in a policy (a key policy) attached to the encryption key that you create. You can do this in your IAM console by creating the following:

- An IAM role with an AWS-managed policy.
- A KMS key with a key policy that references this role.

The following procedures describe how to do this.

### To create an IAM role with the required AWS-managed policy

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Roles**. The **Roles** page opens.
3. Choose **Create role**. The **Create role** page opens.
4. With **AWS service** chosen as the trusted entity, choose **DMS** as the service to use the role.
5. Choose **Next: Permissions**. The **Attach permissions policies** page appears.
6. Find and select the `AmazonDMSRedshiftS3Role` policy.
7. Choose **Next: Tags**. The **Add tags** page appears. Here, you can add any tags you want.
8. Choose **Next: Review** and review your results.
9. If the settings are what you need, enter a name for the role (for example, `DMS-Redshift-endpoint-access-role`), and any additional description, then choose **Create role**. The **Roles** page opens with a message indicating that your role has been created.

You have now created the new role to access Amazon Redshift resources for encryption with a specified name, for example `DMS-Redshift-endpoint-access-role`.

### To create an AWS KMS encryption key with a key policy that references your IAM role

#### Note

For more information about how AWS DMS works with AWS KMS encryption keys, see [Setting an encryption key and specifying AWS KMS permissions \(p. 643\)](#).

1. Sign in to the AWS Management Console and open the AWS Key Management Service (AWS KMS) console at <https://console.aws.amazon.com/kms>.
2. To change the AWS Region, use the Region selector in the upper-right corner of the page.
3. In the navigation pane, choose **Customer managed keys**.
4. Choose **Create key**. The **Configure key** page opens.
5. For **Key type**, choose **Symmetric**.

## Note

When you create this key, you can only create a symmetric key, because all AWS services, such as Amazon Redshift, only work with symmetric encryption keys.

6. Choose **Advanced Options**. For **Key material origin**, make sure that **KMS** is chosen, then choose **Next**. The **Add labels** page opens.
  7. For **Create alias and description**, enter an alias for the key (for example, **DMS-Redshift-endpoint-encryption-key**) and any additional description.
  8. For **Tags**, add any tags that you want to help identify the key and track its usage, then choose **Next**. The **Define key administrative permissions** page opens showing a list of users and roles that you can choose from.
  9. Add the users and roles that you want to manage the key. Make sure that these users and roles have the required permissions to manage the key.
  10. For **Key deletion**, choose whether key administrators can delete the key, then choose **Next**. The **Define key usage permissions** page opens showing an additional list of users and roles that you can choose from.
  11. For **This account**, choose the available users you want to perform cryptographic operations on Amazon Redshift targets. Also choose the role that you previously created in **Roles** to enable access to encrypt Amazon Redshift target objects, for example **DMS-Redshift-endpoint-access-role**.
  12. If you want to add other accounts not listed to have this same access, for **Other AWS accounts**, choose **Add another AWS account**, then choose **Next**. The **Review and edit key policy** page opens, showing the JSON for the key policy that you can review and edit by typing into the existing JSON. Here, you can see where the key policy references the role and users (for example, Admin and User1) that you chose in the previous step. You can also see the different key actions permitted for the different principals (users and roles), as shown in the following example.

```
{
  "Id": "key-consolepolicy-3",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Enable IAM User Permissions",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:root"
        ]
      },
      "Action": "kms:*",
      "Resource": "*"
    },
    {
      "Sid": "Allow access for Key Administrators",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:role/Admin"
        ]
      },
      "Action": [
        "kms>Create*",
        "kms>Describe*",
        "kms>Enable*",
        "kms>List*",
        "kms>Put*",
        "kms>Update*",
        "kms>Revoke*",
        "kms>Disable*",
        "kms>Get*"
      ]
    }
  ]
}
```

```

    "kms>Delete*",
    "kms>TagResource",
    "kms>UntagResource",
    "kms>ScheduleKeyDeletion",
    "kms>CancelKeyDeletion"
],
"Resource": "*"
},
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::111122223333:role/DMS-Redshift-endpoint-access-role",
      "arn:aws:iam::111122223333:role/Admin",
      "arn:aws:iam::111122223333:role/User1"
    ]
  },
  "Action": [
    "kms>Encrypt",
    "kms>Decrypt",
    "kms>ReEncrypt*",
    "kms>GenerateDataKey*",
    "kms>DescribeKey"
  ],
  "Resource": "*"
},
{
  "Sid": "Allow attachment of persistent resources",
  "Effect": "Allow",
  "Principal": {
    "AWS": [
      "arn:aws:iam::111122223333:role/DMS-Redshift-endpoint-access-role",
      "arn:aws:iam::111122223333:role/Admin",
      "arn:aws:iam::111122223333:role/User1"
    ]
  },
  "Action": [
    "kms>CreateGrant",
    "kms>ListGrants",
    "kms>RevokeGrant"
  ],
  "Resource": "*",
  "Condition": {
    "Bool": {
      "kms>GrantIsForAWSResource": true
    }
  }
}
]

```

13. Choose **Finish**. The **Encryption keys** page opens with a message indicating that your AWS KMS key has been created.

You have now created a new KMS key with a specified alias (for example, DMS-Redshift-endpoint-encryption-key). This key enables AWS DMS to encrypt Amazon Redshift target data.

## Endpoint settings when using Amazon Redshift as a target for AWS DMS

You can use endpoint settings to configure your Amazon Redshift target similar to using extra connection attributes. You can specify these settings when you create the target endpoint using the

create-endpoint command in the AWS CLI, with the --redshift-settings "json-settings" option. Here, *json-settings* is a JSON object containing parameters to specify the settings. You can also specify a .json file containing the same *json-settings* object, for example, as in the following: --redshift-settings file:///your-file-path/my\_redshift\_settings.json. Here, my\_redshift\_settings.json is the name of a .json file that contains the same *json-settings* object.

The parameter names for endpoint settings are the same as the names for equivalent extra connections attributes, except that the parameter names for endpoint settings have initial caps. Also, not all Amazon Redshift target endpoint settings using extra connection attributes are available using the --redshift-settings option of the create-endpoint command. For more information about the available settings in an AWS CLI call to create-endpoint, see [create-endpoint](#) in the *AWS CLI Command Reference* for AWS DMS. For more information on these settings, see the equivalent extra connection attributes in [Extra connection attributes when using Amazon Redshift as a target for AWS DMS \(p. 324\)](#).

You can use Amazon Redshift target endpoint settings to configure the following:

- A custom AWS KMS data encryption key. You can then use this key to encrypt your data pushed to Amazon S3 before it is copied to Amazon Redshift.
- A custom S3 bucket as intermediate storage for data migrated to Amazon Redshift.
- Map a boolean as a boolean from a PostgreSQL source. By default, a BOOLEAN type is migrated as varchar(1). You can specify MapBooleanAsBoolean to let your Redshift target migrate the boolean type as boolean, as shown in the example following.

```
redshift-settings='{"MapBooleanAsBoolean": "true"}'
```

## KMS key settings for data encryption

The following examples show configuring a custom KMS key to encrypt your data pushed to S3. To start, you might make the following create-endpoint call using the AWS CLI.

```
aws dms create-endpoint --endpoint-identifier redshift-target-endpoint --endpoint-type target
--engine-name redshift --username your-username --password your-password
--server-name your-server-name --port 5439 --database-name your-db-name
--redshift-settings '{"EncryptionMode": "SSE_KMS",
"ServerSideEncryptionKmsKeyId": "arn:aws:kms:us-east-1:111122223333:key/24c3c5a1-f34a-4519-a85b-2debbeff226d1"}'
```

Here, the JSON object specified by --redshift-settings option defines two parameters. One is an EncryptionMode parameter with the value SSE\_KMS. The other is an ServerSideEncryptionKmsKeyId parameter with the value arn:aws:kms:us-east-1:111122223333:key/24c3c5a1-f34a-4519-a85b-2debbeff226d1. This value is an Amazon Resource Name (ARN) for your custom KMS key.

By default, S3 data encryption occurs using S3 server-side encryption. For the previous example's Amazon Redshift target, this is also equivalent of specifying its endpoint settings, as in the following example.

```
aws dms create-endpoint --endpoint-identifier redshift-target-endpoint --endpoint-type target
--engine-name redshift --username your-username --password your-password
--server-name your-server-name --port 5439 --database-name your-db-name
--redshift-settings '{"EncryptionMode": "SSE_S3"}'
```

For more information about working with S3 server-side encryption, see [Protecting data using server-side encryption](#) in the *Amazon Simple Storage Service User Guide*.

**Note**

You can also use the CLI `modify-endpoint` command to change the value of the `EncryptionMode` parameter for an existing endpoint from `SSE_KMS` to `SSE_S3`. But you can't change the `EncryptionMode` value from `SSE_S3` to `SSE_KMS`.

## Amazon S3 bucket settings

When you migrate data to an Amazon Redshift target endpoint, AWS DMS uses a default Amazon S3 bucket as intermediate task storage before copying the migrated data to Amazon Redshift. For example, the examples shown for creating an Amazon Redshift target endpoint with a AWS KMS data encryption key use this default S3 bucket (see [KMS key settings for data encryption \(p. 323\)](#)).

You can instead specify a custom S3 bucket for this intermediate storage by including the following parameters in the value of your `--redshift-settings` option on the AWS CLI `create-endpoint` command:

- `BucketName` – A string you specify as the name of the S3 bucket storage.
- `BucketFolder` – (Optional) A string you can specify as the name of the storage folder in the specified S3 bucket.
- `ServiceAccessRoleArn` – The ARN of an IAM role that permits administrative access to the S3 bucket. Typically, you create this role based on the `AmazonDMSRedshiftS3Role` policy. For an example, see the procedure to create an IAM role with the required AWS-managed policy in [Creating and using AWS KMS keys to encrypt Amazon Redshift target data \(p. 319\)](#).

**Note**

If you specify the ARN of a different IAM role using the `--service-access-role-arn` option of the `create-endpoint` command, this IAM role option takes precedence.

The following example shows how you might use these parameters to specify a custom Amazon S3 bucket in the following `create-endpoint` call using the AWS CLI.

```
aws dms create-endpoint --endpoint-identifier redshift-target-endpoint --endpoint-type target
--engine-name redshift --username your-username --password your-password
--server-name your-server-name --port 5439 --database-name your-db-name
--redshift-settings '{"ServiceAccessRoleArn": "your-service-access-ARN", "BucketName": "your-bucket-name", "BucketFolder": "your-bucket-folder-name"}'
```

## Extra connection attributes when using Amazon Redshift as a target for AWS DMS

You can use extra connection attributes to configure your Amazon Redshift target. You specify these settings when you create the target endpoint. If you have multiple connection attribute settings, separate them from each other by semicolons with no additional white space.

The following table shows the extra connection attributes available when Amazon Redshift is the target.

Name	Description
<code>maxFileSize</code>	Specifies the maximum size (in KB) of any .csv file used to transfer data to Amazon Redshift.  Default value: 32768 KB (32 MB)

Name	Description
	<p>Valid values: 1–1,048,576</p> <p>Example: <code>maxFileSize=512</code></p>
<code>fileTransferUploadStreams</code>	<p>Specifies the number of threads used to upload a single file.</p> <p>Default value: 10</p> <p>Valid values: 1–64</p> <p>Example: <code>fileTransferUploadStreams=20</code></p>
<code>acceptanydate</code>	<p>Specifies if any date format is accepted, including invalid dates formats such as 0000-00-00. Boolean value.</p> <p>Default value: <code>false</code></p> <p>Valid values: <code>true</code>   <code>false</code></p> <p>Example: <code>acceptanydate=true</code></p>
<code>dateformat</code>	<p>Specifies the date format. This is a string input and is empty by default. The default format is YYYY-MM-DD but you can change it to, for example, DD-MM-YYYY. If your date or time values use different formats, use the auto argument with the dateformat parameter. The auto argument recognizes several formats that are not supported when using a dateformat string. The auto keyword is case-sensitive.</p> <p>Default value: empty</p> <p>Valid values: '<code>dateformat_string</code>' or <code>auto</code></p> <p>Example: <code>dateformat=auto</code></p>
<code>timeformat</code>	<p>Specifies the time format. This is a string input and is empty by default. The auto argument recognizes several formats that aren't supported when using a timeformat string. If your date and time values use formats different from each other, use the auto argument with the timeformat parameter.</p> <p>Default value: 10</p> <p>Valid values: '<code>timeformat_string</code>'   <code>auto</code>   <code>'epochsecs'</code>   <code>'epochmillisecs'</code></p> <p>Example: <code>timeformat=auto</code></p>

Name	Description
emptyasnull	<p>Specifies whether AWS DMS should migrate empty CHAR and VARCHAR fields as null. A value of true sets empty CHAR and VARCHAR fields as null.</p> <p>Default value: false</p> <p>Valid values: true   false</p> <p>Example: emptyasnull=true</p>
truncateColumns	<p>Truncates data in columns to the appropriate number of characters so that it fits the column specification. Applies only to columns with a VARCHAR or CHAR data type, and rows 4 MB or less in size.</p> <p>Default value: false</p> <p>Valid values: true   false</p> <p>Example:</p> <pre>truncateColumns=true;</pre>
removeQuotes	<p>Removes surrounding quotation marks from strings in the incoming data. All characters within the quotation marks, including delimiters, are retained. For more information about removing quotes for an Amazon Redshift target, see the <a href="#">Amazon Redshift Database Developer Guide</a>.</p> <p>Default value: false</p> <p>Valid values: true   false</p> <p>Example:</p> <pre>removeQuotes=true;</pre>
trimBlanks	<p>Removes the trailing white-space characters from a VARCHAR string. This parameter applies only to columns with a VARCHAR data type.</p> <p>Default value: false</p> <p>Valid values: true   false</p> <p>Example:</p> <pre>trimBlanks=false;</pre>

Name	Description
encryptionMode	<p>Specifies the server-side encryption mode that you want to use to push your data to S3 before it is copied to Amazon Redshift. The valid values are SSE_S3 (S3 server-side encryption) or SSE_KMS (KMS key encryption). If you choose SSE_KMS, set the <code>serverSideEncryptionKmsKeyId</code> parameter to the Amazon Resource Name (ARN) for the KMS key to be used for encryption.</p> <p><b>Note</b> You can also use the CLI <code>modify-endpoint</code> command to change the value of the <code>encryptionMode</code> attribute for an existing endpoint from SSE_KMS to SSE_S3. But you can't change the <code>encryptionMode</code> value from SSE_S3 to SSE_KMS.</p> <p>Default value: SSE_S3</p> <p>Valid values: SSE_S3 or SSE_KMS</p> <p>Example:<code>encryptionMode=SSE_S3;</code></p>
serverSideEncryptionKmsKeyId	<p>If you set <code>encryptionMode</code> to SSE_KMS, set this parameter to the ARN for the KMS key. You can find this ARN by selecting the key alias in the list of AWS KMS keys created for your account. When you create the key, you must associate specific policies and roles with it. For more information, see <a href="#">Creating and using AWS KMS keys to encrypt Amazon Redshift target data (p. 319)</a>.</p> <p>Example:</p> <pre>serverSideEncryptionKmsKeyId=arn:aws:kms:us-east-1:111122223333:key/24c3c5a1-f34a-4519-a85b-2debef226d1;</pre>
enableParallelBatchInMemoryCSVFiles	<p>Set this attribute to false when using the <code>ParallelLoadThreads</code> task setting. The <code>enableParallelBatchInMemoryCSVFiles</code> attribute improves performance of larger multithreaded full load tasks by having DMS write to disk instead of memory. The default value is true.</p>
compressCsvFile	<p>Use this attribute to compress data sent to a Redshift target during migration. The default value is true, and compression is enabled by default.</p>

## Multithreaded task settings for Amazon Redshift

You can improve performance of full load and change data capture (CDC) tasks for an Amazon Redshift target endpoint by using multithreaded task settings. They enable you to specify the number of concurrent threads and the number of records to store in a buffer.

## Multithreaded full load task settings for Amazon Redshift

To promote full load performance, you can use the following `ParallelLoad*` task settings:

- `ParallelLoadThreads` – Specifies the number of concurrent threads that DMS uses during a full load to push data records to an Amazon Redshift target endpoint. The default value is zero (0) and the maximum value is 32.

You can use the `enableParallelBatchInMemoryCSVFiles` attribute set to `false` when using the `ParallelLoadThreads` task setting. The attribute improves performance of larger multithreaded full load tasks by having DMS write to disk instead of memory. The default value is `true`.

- `ParallelLoadBufferSize` – Specifies the maximum data record requests while using parallel load threads with Redshift target. The default value is 100 and the maximum value is 1,000. We recommend you use this option when `ParallelLoadThreads > 1` (greater than one).

### Note

Support for the use of `ParallelLoad*` task settings during FULL LOAD to Amazon Redshift target endpoints is available in AWS DMS versions 3.4.5 and later.

The `ReplaceInvalidChars` Redshift endpoint setting is not supported for use during change data capture (CDC) or during a parallel load enabled FULL LOAD migration task. It is supported for FULL LOAD migration when parallel load isn't enabled. For more information see [RedshiftSettings](#) in the *AWS Database Migration Service API Reference*

## Multithreaded CDC task settings for Amazon Redshift

To promote CDC performance, you can use the following `ParallelApply*` task settings:

- `ParallelApplyThreads` – Specifies the number of concurrent threads that AWS DMS uses during a CDC load to push data records to a Amazon Redshift target endpoint. The default value is zero (0) and the maximum value is 32.
- `ParallelApplyBufferSize` – Specifies the maximum data record requests while using parallel apply threads with Redshift target. The default value is 100 and the maximum value is 1,000. We recommend to use this option when `ParallelApplyThreads > 1` (greater than one).

To obtain the most benefit for Redshift as a target, we recommend that the value of `ParallelApplyBufferSize` be at least two times (double or more) the number of `ParallelApplyThreads`.

### Note

Support for the use of `ParallelApply*` task settings during CDC to Amazon Redshift target endpoints is available in AWS DMS versions 3.4.3 and later.

The level of parallelism applied depends on the correlation between the total *batch size* and the *maximum file size* used to transfer data. When using multithreaded CDC task settings with a Redshift target, benefits are gained when batch size is large in relation to the maximum file size. For example, you can use the following combination of endpoint and task settings to tune for optimal performance.

```
// Redshift endpoint setting
    MaxFileSize=250000;
// Task settings
    BatchApplyEnabled=true;
    BatchSplitSize =8000;
    BatchApplyTimeoutMax =1800;
    BatchApplyTimeoutMin =1800;
```

```
ParallelApplyThreads=32;
ParallelApplyBufferSize=100;
```

Using the settings above, a customer with a heavy transactional workload benefits by their 8000 MB batch buffer getting filled in 1800 seconds, utilizing 32 parallel threads with a 250 MB maximum file size.

For more information, see [Change processing tuning settings \(p. 465\)](#).

**Note**

DMS queries that run during ongoing replication to a Redshift cluster can share the same WLM (workload management) queue with other application queries that are running. So, consider properly configuring WLM properties to influence performance during ongoing replication to a Redshift target. For example, if other parallel ETL queries are running, DMS runs slower and performance gains are lost.

## Target data types for Amazon Redshift

The Amazon Redshift endpoint for AWS DMS supports most Amazon Redshift data types. The following table shows the Amazon Redshift target data types that are supported when using AWS DMS and the default mapping from AWS DMS data types.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

AWS DMS data types	Amazon Redshift data types
BOOLEAN	BOOL
BYTES	VARCHAR (Length)
DATE	DATE
TIME	VARCHAR(20)
DATETIME	If the scale is => 0 and =< 6, depending on Redshift target column type, then one of the following:  TIMESTAMP (s)  TIMESTAMPTZ (s) — If source timestamp contains a zone offset (such as in SQL Server or Oracle) it converts to UTC on insert/update. If it doesn't contain an offset, then time is considered in UTC already.  If the scale is => 7 and =< 9, then:  VARCHAR (37)
INT1	INT2
INT2	INT2
INT4	INT4
INT8	INT8
NUMERIC	If the scale is => 0 and =< 37, then:

AWS DMS data types	Amazon Redshift data types
	NUMERIC (p,s)  If the scale is => 38 and =< 127, then:  VARCHAR (Length)
REAL4	FLOAT4
REAL8	FLOAT8
STRING	If the length is 1–65,535, then use VARCHAR (length in bytes)  If the length is 65,536–2,147,483,647, then use VARCHAR (65535)
UINT1	INT2
UINT2	INT2
UINT4	INT4
UINT8	NUMERIC (20,0)
WSTRING	If the length is 1–65,535, then use NVARCHAR (length in bytes)  If the length is 65,536–2,147,483,647, then use NVARCHAR (65535)
BLOB	VARCHAR (maximum LOB size *2)  The maximum LOB size cannot exceed 31 KB. Amazon Redshift doesn't support VARCHARs larger than 64 KB.
NCLOB	NVARCHAR (maximum LOB size)  The maximum LOB size cannot exceed 63 KB. Amazon Redshift doesn't support VARCHARs larger than 64 KB.
CLOB	VARCHAR (maximum LOB size)  The maximum LOB size cannot exceed 63 KB. Amazon Redshift doesn't support VARCHARs larger than 64 KB.

## Using a SAP ASE database as a target for AWS Database Migration Service

You can migrate data to SAP Adaptive Server Enterprise (ASE)—formerly known as Sybase—databases using AWS DMS, either from any of the supported database sources.

SAP ASE versions 15, 15.5, 15.7, 16 and later are supported.

## Prerequisites for using a SAP ASE database as a target for AWS Database Migration Service

Before you begin to work with a SAP ASE database as a target for AWS DMS, make sure that you have the following prerequisites:

- Provide SAP ASE account access to the AWS DMS user. This user must have read/write privileges in the SAP ASE database.
- In some cases, you might replicate to SAP ASE version 15.7 installed on an Amazon EC2 instance on Microsoft Windows that is configured with non-Latin characters (for example, Chinese). In such cases, AWS DMS requires SAP ASE 15.7 SP121 to be installed on the target SAP ASE machine.

## Limitations when using a SAP ASE database as a target for AWS DMS

The following limitations apply when using an SAP ASE database as a target for AWS DMS:

- AWS DMS doesn't support tables that include fields with the following data types. Replicated columns with these data types show as null.
  - User-defined type (UDT)

## Extra connection attributes when using SAP ASE as a target for AWS DMS

You can use extra connection attributes to configure your SAP ASE target. You specify these settings when you create the target endpoint. If you have multiple connection attribute settings, separate them from each other by semicolons with no additional white space.

The following table shows the extra connection attributes available when using SAP ASE as a target:

Name	Description
driver	Set this attribute if you want to use TLS for versions of ASE 15.7 and later.  Default value: Adaptive Server Enterprise  Example: driver=Adaptive Server Enterprise 16.03.06;  Valid values: Adaptive Server Enterprise 16.03.06
additionalConnectionProperties	Any additional ODBC connection parameters that you want to specify.

## Target data types for SAP ASE

The following table shows the SAP ASE database target data types that are supported when using AWS DMS and the default mapping from AWS DMS data types.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

AWS DMS data types	SAP ASE data types
BOOLEAN	BIT
BYTES	VARBINARY (Length)
DATE	DATE
TIME	TIME
TIMESTAMP	If scale is => 0 and =< 6, then: BIGDATETIME If scale is => 7 and =< 9, then: VARCHAR (37)
INT1	TINYINT
INT2	SMALLINT
INT4	INTEGER
INT8	BIGINT
NUMERIC	NUMERIC (p,s)
REAL4	REAL
REAL8	DOUBLE PRECISION
STRING	VARCHAR (Length)
UINT1	TINYINT
UINT2	UNSIGNED SMALLINT
UINT4	UNSIGNED INTEGER
UINT8	UNSIGNED BIGINT
WSTRING	VARCHAR (Length)
BLOB	IMAGE
CLOB	UNITEXT
NCLOB	TEXT

## Using Amazon S3 as a target for AWS Database Migration Service

You can migrate data to Amazon S3 using AWS DMS from any of the supported database sources. When using Amazon S3 as a target in an AWS DMS task, both full load and change data capture (CDC) data is written to comma-separated value (.csv) format by default. For more compact storage and faster query options, you also have the option to have the data written to Apache Parquet (.parquet) format.

AWS DMS names files created during a full load using an incremental hexadecimal counter—for example LOAD00001.csv, LOAD00002..., LOAD00009, LOAD0000A, and so on for .csv files. AWS DMS names CDC files using timestamps, for example 20141029-1134010000.csv. For each source table that contains records, AWS DMS creates a folder under the specified target folder (if the source table is not empty). AWS DMS writes all full load and CDC files to the specified Amazon S3 bucket.

The parameter `bucketFolder` contains the location where the .csv or .parquet files are stored before being uploaded to the S3 bucket. With .csv files, table data is stored in the following format in the S3 bucket, shown with full-load files.

```
database_schema_name/table_name/LOAD00000001.csv
database_schema_name/table_name/LOAD00000002.csv
...
database_schema_name/table_name/LOAD00000009.csv
database_schema_name/table_name/LOAD0000000A.csv
database_schema_name/table_name/LOAD0000000B.csv
...database_schema_name/table_name/LOAD0000000F.csv
database_schema_name/table_name/LOAD00000010.csv
...
```

You can specify the column delimiter, row delimiter, and other parameters using the extra connection attributes. For more information on the extra connection attributes, see [Extra connection attributes when using Amazon S3 as a target for AWS DMS \(p. 350\)](#) at the end of this section.

You can specify a bucket owner and prevent sniping by using the `ExpectedBucketOwner` Amazon S3 endpoint setting, as shown following. Then, when you make a request to test a connection or perform a migration, S3 checks the account ID of the bucket owner against the specified parameter.

```
--s3-settings='{"ExpectedBucketOwner": "AWS_Account_ID"}'
```

When you use AWS DMS to replicate data changes using a CDC task, the first column of the .csv or .parquet output file indicates how the row data was changed as shown for the following .csv file.

```
I,101,Smith,Bob,4-Jun-14,New York
U,101,Smith,Bob,8-Oct-15,Los Angeles
U,101,Smith,Bob,13-Mar-17,Dallas
D,101,Smith,Bob,13-Mar-17,Dallas
```

For this example, suppose that there is an `EMPLOYEE` table in the source database. AWS DMS writes data to the .csv or .parquet file, in response to the following events:

- A new employee (Bob Smith, employee ID 101) is hired on 4-Jun-14 at the New York office. In the .csv or .parquet file, the I in the first column indicates that a new row was INSERTed into the `EMPLOYEE` table at the source database.
- On 8-Oct-15, Bob transfers to the Los Angeles office. In the .csv or .parquet file, the U indicates that the corresponding row in the `EMPLOYEE` table was UPDATED to reflect Bob's new office location. The rest of the line reflects the row in the `EMPLOYEE` table as it appears after the UPDATE.
- On 13-Mar-17, Bob transfers again to the Dallas office. In the .csv or .parquet file, the U indicates that this row was UPDATED again. The rest of the line reflects the row in the `EMPLOYEE` table as it appears after the UPDATE.
- After some time working in Dallas, Bob leaves the company. In the .csv or .parquet file, the D indicates that the row was DELETED in the source table. The rest of the line reflects how the row in the `EMPLOYEE` table appeared before it was deleted.

Note that by default for CDC, AWS DMS stores the row changes for each database table without regard to transaction order. If you want to store the row changes in CDC files according to transaction order, you need to use S3 endpoint settings to specify this and the folder path where you want the CDC transaction files to be stored on the S3 target. For more information, see [Capturing data changes \(CDC\) including transaction order on the S3 target \(p. 349\)](#).

To control the frequency of writes to an Amazon S3 target during a data replication task, you can configure the `cdcMaxBatchInterval` and `cdcMinFileSize` extra connection attributes. This can result in better performance when analyzing the data without any additional overhead operations. For more information, see [Extra connection attributes when using Amazon S3 as a target for AWS DMS \(p. 350\)](#)

### Topics

- [Prerequisites for using Amazon S3 as a target \(p. 334\)](#)
- [Limitations to using Amazon S3 as a target \(p. 335\)](#)
- [Security \(p. 335\)](#)
- [Using Apache Parquet to store Amazon S3 objects \(p. 335\)](#)
- [Amazon S3 object tagging \(p. 336\)](#)
- [Creating AWS KMS keys to encrypt Amazon S3 target objects \(p. 341\)](#)
- [Using date-based folder partitioning \(p. 344\)](#)
- [Parallel load of partitioned sources when using Amazon S3 as a target for AWS DMS \(p. 345\)](#)
- [Endpoint settings when using Amazon S3 as a target for AWS DMS \(p. 347\)](#)
- [Extra connection attributes when using Amazon S3 as a target for AWS DMS \(p. 350\)](#)
- [Indicating source DB operations in migrated S3 data \(p. 359\)](#)
- [Target data types for S3 Parquet \(p. 360\)](#)

## Prerequisites for using Amazon S3 as a target

Before using Amazon S3 as a target, check that the following are true:

- The S3 bucket that you're using as a target is in the same AWS Region as the DMS replication instance you are using to migrate your data.
- The AWS account that you use for the migration has an IAM role with write and delete access to the S3 bucket you are using as a target.
- This role has tagging access so you can tag any S3 objects written to the target bucket.
- The IAM role has DMS (`dms.amazonaws.com`) added as *trusted entity*.

To set up this account access, ensure that the role assigned to the user account used to create the migration task has the following set of permissions.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3:PutObject",  
                "s3>DeleteObject",  
                "s3:PutObjectTagging"  
            ],  
            "Resource": [  
                "arn:aws:s3:::buckettest2/*"  
            ]  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "s3>ListBucket"  
            ],  
            "Resource": [  
                "arn:aws:s3:::buckettest2"  
            ]  
        }  
    ]  
}
```

```
        "arn:aws:s3:::buckettest2"
    ]
}
}
```

## Limitations to using Amazon S3 as a target

The following limitations apply when using Amazon S3 as a target:

- Don't enable versioning for S3. If you need S3 versioning, use lifecycle policies to actively delete old versions. Otherwise, you might encounter endpoint test connection failures because of an S3 list-object call timeout. To create a lifecycle policy for an S3 bucket, see [Managing your storage lifecycle](#). To delete a version of an S3 object, see [Deleting object versions from a versioning-enabled bucket](#).
- A VPC-enabled (gateway VPC) S3 bucket is supported in versions 3.4.7 and higher.
- The following data definition language (DDL) commands are supported for change data capture (CDC): Truncate Table, Drop Table, Create Table, Rename Table, Add Column, Drop Column, Rename Column, and Change Column Data Type.

**Note**

A truncate DDL operation removes all files and corresponding table folders from an S3 bucket.

You can use task settings to disable that behavior and configure the way DMS handles DDL behavior during change data capture (CDC). For more information, see [Task settings for change processing DDL handling \(p. 468\)](#).

- Full LOB mode is not supported.
- Changes to the source table structure during full load are not supported. Changes to data are supported during full load.
- Multiple tasks that replicate data from the same source table to the same target S3 endpoint bucket result in those tasks writing to the same file. We recommend that you specify different target endpoints (buckets) if your data source is from the same table.
- BatchApply is not supported for an S3 endpoint. Using Batch Apply (for example, the BatchApplyEnabled target metadata task setting) for an S3 target might result in loss of data.
- You can't use DatePartitionEnabled or addColumnName together with PreserveTransactions or CdcPath.

## Security

To use Amazon S3 as a target, the account used for the migration must have write and delete access to the Amazon S3 bucket that is used as the target. Specify the Amazon Resource Name (ARN) of an IAM role that has the permissions required to access Amazon S3.

AWS DMS supports a set of predefined grants for Amazon S3, known as canned access control lists (ACLs). Each canned ACL has a set of grantees and permissions that you can use to set permissions for the Amazon S3 bucket. You can specify a canned ACL using the cannedAclForObjects on the connection string attribute for your S3 target endpoint. For more information about using the extra connection attribute cannedAclForObjects, see [Extra connection attributes when using Amazon S3 as a target for AWS DMS \(p. 350\)](#). For more information about Amazon S3 canned ACLs, see [Canned ACL](#).

The IAM role that you use for the migration must be able to perform the s3:PutObjectAcl API operation.

## Using Apache Parquet to store Amazon S3 objects

The comma-separated value (.csv) format is the default storage format for Amazon S3 target objects. For more compact storage and faster queries, you can instead use Apache Parquet (.parquet) as the storage format.

Apache Parquet is an open-source file storage format originally designed for Hadoop. For more information on Apache Parquet, see <https://parquet.apache.org/>.

To set .parquet as the storage format for your migrated S3 target objects, you can use the following mechanisms:

- Endpoint settings that you provide as parameters of a JSON object when you create the endpoint using the AWS CLI or the API for AWS DMS. For more information, see [Endpoint settings when using Amazon S3 as a target for AWS DMS \(p. 347\)](#).
- Extra connection attributes that you provide as a semicolon-separated list when you create the endpoint. For more information, see [Extra connection attributes when using Amazon S3 as a target for AWS DMS \(p. 350\)](#).

## Amazon S3 object tagging

You can tag Amazon S3 objects that a replication instance creates by specifying appropriate JSON objects as part of task-table mapping rules. For more information about requirements and options for S3 object tagging, including valid tag names, see [Object tagging](#) in the *Amazon Simple Storage Service User Guide*. For more information about table mapping using JSON, see [Specifying table selection and transformations rules using JSON \(p. 494\)](#).

You tag S3 objects created for specified tables and schemas by using one or more JSON objects of the selection rule type. You then follow this selection object (or objects) by one or more JSON objects of the post-processing rule type with add-tag action. These post-processing rules identify the S3 objects that you want to tag and specify the names and values of the tags that you want to add to these S3 objects.

You can find the parameters to specify in JSON objects of the post-processing rule type in the following table.

Parameter	Possible values	Description
rule-type	post-processing	A value that applies post-processing actions to the generated target objects. You can specify one or more post-processing rules to tag selected S3 objects.
rule-id	A numeric value.	A unique numeric value to identify the rule.
rule-name	An alphanumeric value.	A unique name to identify the rule.
rule-action	add-tag	The post-processing action that you want to apply to the S3 object. You can add one or more tags using a single JSON post-processing object for the add-tag action.
object-locator	schema-name – The name of the table schema. table-name – The name of the table.	The name of each schema and table to which the rule applies. You can use the "%" percent sign as a wildcard for all or part of the value of each object-locator parameter. Thus, you can match these items: <ul style="list-style-type: none"><li>• A single table in a single schema</li></ul>

Parameter	Possible values	Description
		<ul style="list-style-type: none"> <li>• A single table in some or all schemas</li> <li>• Some or all tables in a single schema</li> <li>• Some or all tables in some or all schemas</li> </ul>
tag-set	<p>key – Any valid name for a single tag.</p> <p>value – Any valid JSON value for this tag.</p>	<p>The names and values for one or more tags that you want to set on each created S3 object that matches the specified object-locator. You can specify up to 10 key-value pairs in a single tag-set parameter object. For more information on S3 object tagging, see <a href="#">Object tagging in the Amazon Simple Storage Service User Guide</a>.</p> <p>You can also specify a dynamic value for all or part of the value for both the key and value parameters of a tag using \${dyn-value}. Here, \${dyn-value} can be either \${schema-name} or \${table-name}. Thus, you can insert the name of the currently selected schema or table as the whole or any part of the parameter value.</p> <p><b>Note</b></p> <p><b>Important</b> If you insert a dynamic value for the key parameter, you can generate tags with duplicate names for an S3 object, depending on how you use it. In this case, only one of the duplicate tag settings is added to the object.</p>

When you specify multiple post-processing rule types to tag a selection of S3 objects, each S3 object is tagged using only one tag-set object from one post-processing rule. The particular tag set used to tag a given S3 object is the one from the post-processing rule whose associated object locator best matches that S3 object.

For example, suppose that two post-processing rules identify the same S3 object. Suppose also that the object locator from one rule uses wildcards and the object locator from the other rule uses an exact match to identify the S3 object (without wildcards). In this case, the tag set associated with the post-processing rule with the exact match is used to tag the S3 object. If multiple post-processing rules match a given S3 object equally well, the tag set associated with the first such post-processing rule is used to tag the object.

### Example Adding static tags to an S3 object created for a single table and schema

The following selection and post-processing rules add three tags (`tag_1`, `tag_2`, and `tag_3` with corresponding static values `value_1`, `value_2`, and `value_3`) to a created S3 object. This S3 object corresponds to a single table in the source named STOCK with a schema named aat2.

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "5",
      "rule-name": "5",
      "object-locator": {
        "schema-name": "aat2",
        "table-name": "STOCK"
      },
      "rule-action": "include"
    },
    {
      "rule-type": "post-processing",
      "rule-id": "41",
      "rule-name": "41",
      "rule-action": "add-tag",
      "object-locator": {
        "schema-name": "aat2",
        "table-name": "STOCK"
      },
      "tag-set": [
        {
          "key": "tag_1",
          "value": "value_1"
        },
        {
          "key": "tag_2",
          "value": "value_2"
        },
        {
          "key": "tag_3",
          "value": "value_3"
        }
      ]
    }
  ]
}
```

### Example Adding static and dynamic tags to S3 objects created for multiple tables and schemas

The following example has one selection and two post-processing rules, where input from the source includes all tables and all of their schemas.

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "%",
        "table-name": "%"
      },
      "rule-action": "include"
    }
  ]
}
```

```

},
{
    "rule-type": "post-processing",
    "rule-id": "21",
    "rule-name": "21",
    "rule-action": "add-tag",
    "object-locator": {
        "schema-name": "%",
        "table-name": "%",
    },
    "tag-set": [
        {
            "key": "dw-schema-name",
            "value": "${schema-name}"
        },
        {
            "key": "dw-schema-table",
            "value": "my_prefix_${table-name}"
        }
    ]
},
{
    "rule-type": "post-processing",
    "rule-id": "41",
    "rule-name": "41",
    "rule-action": "add-tag",
    "object-locator": {
        "schema-name": "aat",
        "table-name": "ITEM",
    },
    "tag-set": [
        {
            "key": "tag_1",
            "value": "value_1"
        },
        {
            "key": "tag_2",
            "value": "value_2"
        }
    ]
}
]
}

```

The first post-processing rule adds two tags (dw-schema-name and dw-schema-table) with corresponding dynamic values (\${schema-name} and my\_prefix\_\${table-name}) to almost all S3 objects created in the target. The exception is the S3 object identified and tagged with the second post-processing rule. Thus, each target S3 object identified by the wildcard object locator is created with tags that identify the schema and table to which it corresponds in the source.

The second post-processing rule adds tag\_1 and tag\_2 with corresponding static values value\_1 and value\_2 to a created S3 object that is identified by an exact-match object locator. This created S3 object thus corresponds to the single table in the source named ITEM with a schema named aat. Because of the exact match, these tags replace any tags on this object added from the first post-processing rule, which matches S3 objects by wildcard only.

### Example Adding both dynamic tag names and values to S3 objects

The following example has two selection rules and one post-processing rule. Here, input from the source includes just the ITEM table in either the retail or wholesale schema.

```
{
    "rules": [

```

```
{
    "rule-type": "selection",
    "rule-id": "1",
    "rule-name": "1",
    "object-locator": {
        "schema-name": "retail",
        "table-name": "ITEM"
    },
    "rule-action": "include"
},
{
    "rule-type": "selection",
    "rule-id": "1",
    "rule-name": "1",
    "object-locator": {
        "schema-name": "wholesale",
        "table-name": "ITEM"
    },
    "rule-action": "include"
},
{
    "rule-type": "post-processing",
    "rule-id": "21",
    "rule-name": "21",
    "rule-action": "add-tag",
    "object-locator": {
        "schema-name": "%",
        "table-name": "ITEM",
    },
    "tag-set": [
        {
            "key": "dw-schema-name",
            "value": "${schema-name}"
        },
        {
            "key": "dw-schema-table",
            "value": "my_prefix_ITEM"
        },
        {
            "key": "${schema-name}_ITEM_tag_1",
            "value": "value_1"
        },
        {
            "key": "${schema-name}_ITEM_tag_2",
            "value": "value_2"
        }
    ]
}
}
```

The tag set for the post-processing rule adds two tags (dw-schema-name and dw-schema-table) to all S3 objects created for the ITEM table in the target. The first tag has the dynamic value "\${schema-name}" and the second tag has a static value, "my\_prefix\_ITEM". Thus, each target S3 object is created with tags that identify the schema and table to which it corresponds in the source.

In addition, the tag set adds two additional tags with dynamic names (\${schema-name}\_ITEM\_tag\_1 and \${schema-name}\_ITEM\_tag\_2). These have the corresponding static values value\_1 and value\_2. Thus, these tags are each named for the current schema, retail or wholesale. You can't create a duplicate dynamic tag name in this object, because each object is created for a single unique schema name. The schema name is used to create an otherwise unique tag name.

## Creating AWS KMS keys to encrypt Amazon S3 target objects

You can create and use custom AWS KMS keys to encrypt your Amazon S3 target objects. After you create a KMS key, you can use it to encrypt objects using one of the following approaches when you create the S3 target endpoint:

- Use the following options for S3 target objects (with the default .csv file storage format) when you run the `create-endpoint` command using the AWS CLI.

```
--s3-settings '{"ServiceAccessRoleArn": "your-service-access-ARN",  
"CsvRowDelimiter": "\n", "CsvDelimiter": ",", "BucketFolder": "your-bucket-folder",  
"BucketName": "your-bucket-name", "EncryptionMode": "SSE_KMS",  
"ServerSideEncryptionKmsKeyId": "your-KMS-key-ARN"}'
```

Here, `your-KMS-key-ARN` is the Amazon Resource Name (ARN) for your KMS key. For more information, see [Endpoint settings when using Amazon S3 as a target for AWS DMS \(p. 347\)](#).

- Set the extra connection attribute `encryptionMode` to the value `SSE_KMS` and the extra connection attribute `serverSideEncryptionKmsKeyId` to the ARN for your KMS key. For more information, see [Extra connection attributes when using Amazon S3 as a target for AWS DMS \(p. 350\)](#).

To encrypt Amazon S3 target objects using a KMS key, you need an IAM role that has permissions to access the Amazon S3 bucket. This IAM role is then accessed in a policy (a key policy) attached to the encryption key that you create. You can do this in your IAM console by creating the following:

- A policy with permissions to access the Amazon S3 bucket.
- An IAM role with this policy.
- A KMS key encryption key with a key policy that references this role.

The following procedures describe how to do this.

### To create an IAM policy with permissions to access the Amazon S3 bucket

1. Open the IAM console at <https://console.aws.amazon.com/iam/>.
2. In the navigation pane, choose **Policies** in the navigation pane. The **Policies** page opens.
3. Choose **Create policy**. The **Create policy** page opens.
4. Choose **Service** and choose **S3**. A list of action permissions appears.
5. Choose **Expand all** to expand the list and choose the following permissions at a minimum:
  - **ListBucket**
  - **PutObject**
  - **DeleteObject**
- Choose any other permissions you need, and then choose **Collapse all** to collapse the list.
6. Choose **Resources** to specify the resources that you want to access. At a minimum, choose **All resources** to provide general Amazon S3 resource access.
7. Add any other conditions or permissions you need, then choose **Review policy**. Check your results on the **Review policy** page.
8. If the settings are what you need, enter a name for the policy (for example, `DMS-S3-endpoint-access`), and any description, then choose **Create policy**. The **Policies** page opens with a message indicating that your policy has been created.
9. Search for and choose the policy name in the **Policies** list. The **Summary** page appears displaying JSON for the policy similar to the following.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "VisualEditor0",  
            "Effect": "Allow",  
            "Action": [  
                "s3:PutObject",  
                "s3>ListBucket",  
                "s3>DeleteObject"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

You have now created the new policy to access Amazon S3 resources for encryption with a specified name, for example DMS-S3-endpoint-access.

### To create an IAM role with this policy

1. On your IAM console, choose **Roles** in the navigation pane. The **Roles** detail page opens.
2. Choose **Create role**. The **Create role** page opens.
3. With AWS service selected as the trusted entity, choose **DMS** as the service to use the IAM role.
4. Choose **Next: Permissions**. The **Attach permissions policies** view appears in the **Create role** page.
5. Find and select the IAM policy for the IAM role that you created in the previous procedure (DMS-S3-endpoint-access).
6. Choose **Next: Tags**. The **Add tags** view appears in the **Create role** page. Here, you can add any tags you want.
7. Choose **Next: Review**. The **Review** view appears in the **Create role** page. Here, you can verify the results.
8. If the settings are what you need, enter a name for the role (required, for example, DMS-S3-endpoint-access-role), and any additional description, then choose **Create role**. The **Roles** detail page opens with a message indicating that your role has been created.

You have now created the new role to access Amazon S3 resources for encryption with a specified name, for example, DMS-S3-endpoint-access-role.

### To create a KMS key encryption key with a key policy that references your IAM role

#### Note

For more information about how AWS DMS works with AWS KMS encryption keys, see [Setting an encryption key and specifying AWS KMS permissions \(p. 643\)](#).

1. Sign in to the AWS Management Console and open the AWS Key Management Service (AWS KMS) console at <https://console.aws.amazon.com/kms>.
2. To change the AWS Region, use the Region selector in the upper-right corner of the page.
3. In the navigation pane, choose **Customer managed keys**.
4. Choose **Create key**. The **Configure key** page opens.
5. For **Key type**, choose **Symmetric**.

#### Note

When you create this key, you can only create a symmetric key, because all AWS services, such as Amazon S3, only work with symmetric encryption keys.

6. Choose **Advanced Options**. For **Key material origin**, make sure that **KMS** is chosen, then choose **Next**. The **Add labels** page opens.
7. For **Create alias and description**, enter an alias for the key (for example, `DMS-S3-endpoint-encryption-key`) and any additional description.
8. For **Tags**, add any tags that you want to help identify the key and track its usage, then choose **Next**. The **Define key administrative permissions** page opens showing a list of users and roles that you can choose from.
9. Add the users and roles that you want to manage the key. Make sure that these users and roles have the required permissions to manage the key.
10. For **Key deletion**, choose whether key administrators can delete the key, then choose **Next**. The **Define key usage permissions** page opens showing an additional list of users and roles that you can choose from.
11. For **This account**, choose the available users you want to perform cryptographic operations on Amazon S3 targets. Also choose the role that you previously created in **Roles** to enable access to encrypt Amazon S3 target objects, for example `DMS-S3-endpoint-access-role`.
12. If you want to add other accounts not listed to have this same access, for **Other AWS accounts**, choose **Add another AWS account**, then choose **Next**. The **Review and edit key policy** page opens, showing the JSON for the key policy that you can review and edit by typing into the existing JSON. Here, you can see where the key policy references the role and users (for example, `Admin` and `User1`) that you chose in the previous step. You can also see the different key actions permitted for the different principals (users and roles), as shown in the example following.

```
{  
  "Id": "key-consolepolicy-3",  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "Enable IAM User Permissions",  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": [  
          "arn:aws:iam::111122223333:root"  
        ]  
      },  
      "Action": "kms:*",  
      "Resource": "*"  
    },  
    {  
      "Sid": "Allow access for Key Administrators",  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": [  
          "arn:aws:iam::111122223333:role/Admin"  
        ]  
      },  
      "Action": [  
        "kms>Create*",  
        "kmsDescribe*",  
        "kmsEnable*",  
        "kmsList*",  
        "kmsPut*",  
        "kmsUpdate*",  
        "kmsRevoke*",  
        "kmsDisable*",  
        "kmsGet*",  
        "kmsDelete*",  
        "kmsTagResource",  
        "kmsUntagResource",  
        "kmsScheduleKeyDeletion",  
        "kmsCancelKeyDeletion"  
      ]  
    }  
  ]  
}
```

```

        ],
        "Resource": "*"
    },
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::111122223333:role/DMS-S3-endpoint-access-role",
            "arn:aws:iam::111122223333:role/Admin",
            "arn:aws:iam::111122223333:role/User1"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
},
{
    "Sid": "Allow attachment of persistent resources",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::111122223333:role/DMS-S3-endpoint-access-role",
            "arn:aws:iam::111122223333:role/Admin",
            "arn:aws:iam::111122223333:role/User1"
        ]
    },
    "Action": [
        "kms>CreateGrant",
        "kms>ListGrants",
        "kms:RevokeGrant"
    ],
    "Resource": "*",
    "Condition": {
        "Bool": {
            "kms:GrantIsForAWSResource": true
        }
    }
}
]

```

13. Choose **Finish**. The **Encryption keys** page opens with a message indicating that your KMS key has been created.

You have now created a new KMS key with a specified alias (for example, DMS-S3-endpoint-encryption-key). This key enables AWS DMS to encrypt Amazon S3 target objects.

## Using date-based folder partitioning

AWS DMS supports S3 folder partitions based on a transaction commit date when you use Amazon S3 as your target endpoint. Using date-based folder partitioning, you can write data from a single source table to a time-hierarchy folder structure in an S3 bucket. By partitioning folders when creating an S3 target endpoint, you can do the following:

- Better manage your S3 objects
- Limit the size of each S3 folder
- Optimize data lake queries or other subsequent operations

You can enable date-based folder partitioning when you create an S3 target endpoint. You can enable it when you either migrate existing data and replicate ongoing changes (full load + CDC), or replicate data changes only (CDC only). Use the following target endpoint settings:

- **DatePartitionEnabled** – Specifies partitioning based on dates. Set this Boolean option to `true` to partition S3 bucket folders based on transaction commit dates.

You can't use this setting with `PreserveTransactions` or `CdcPath`.

The default value is `false`.

- **DatePartitionSequence** – Identifies the sequence of the date format to use during folder partitioning. Set this ENUM option to `YYYYMMDD`, `YYYYMMDDHH`, `YYYYMM`, `MMYYYYDD`, or `DDMMYYYY`. The default value is `YYYYMMDD`. Use this setting when `DatePartitionEnabled` is set to `true`.
- **DatePartitionDelimiter** – Specifies a date separation delimiter to use during folder partitioning. Set this ENUM option to `SLASH`, `DASH`, `UNDERSCORE`, or `NONE`. The default value is `SLASH`. Use this setting when `DatePartitionEnabled` is set to `true`.

The following example shows how to enable date-based folder partitioning, with default values for the data partition sequence and the delimiter. It uses the `--s3-settings '{json-settings}'` option of the AWS CLI `create-endpoint` command.

```
--s3-settings '{"DatePartitionEnabled": true, "DatePartitionSequence": "YYYYMMDD", "DatePartitionDelimiter": "SLASH"}'
```

## Parallel load of partitioned sources when using Amazon S3 as a target for AWS DMS

You can configure a parallel full load of partitioned data sources to Amazon S3 targets. This approach improves the load times for migrating partitioned data from supported source database engines to the S3 target. To improve the load times of partitioned source data, you create S3 target subfolders mapped to the partitions of every table in the source database. These partition-bound subfolders allow AWS DMS to run parallel processes to populate each subfolder on the target.

To configure a parallel full load of an S3 target, S3 supports three parallel-load rule types for the `table-settings` rule of table mapping:

- `partitions-auto`
- `partitions-list`
- `ranges`

For more information on these parallel-load rule types, see [Table and collection settings rules and operations \(p. 523\)](#).

For the `partitions-auto` and `partitions-list` rule types, AWS DMS uses each partition name from the source endpoint to identify the target subfolder structure, as follows.

```
bucket_name/bucket_folder/database_schema_name/table_name/partition_name/LOADseq_num.csv
```

Here, the subfolder path where data is migrated and stored on the S3 target includes an additional `partition_name` subfolder that corresponds to a source partition with the same name. This `partition_name` subfolder then stores one or more `LOADseq_num.csv` files containing data migrated

from the specified source partition. Here, *seq\_num* is the sequence number postfix on the .csv file name, such as 00000001 in the .csv file with the name, LOAD00000001.csv.

However, some database engines, such as MongoDB and DocumentDB, don't have the concept of partitions. For these database engines, AWS DMS adds the running source segment index as a prefix to the target .csv file name, as follows.

```
.../database_schema_name/table_name/SEGMENT1_LOAD00000001.csv  
.../database_schema_name/table_name/SEGMENT1_LOAD00000002.csv  
...  
.../database_schema_name/table_name/SEGMENT2_LOAD00000009.csv  
.../database_schema_name/table_name/SEGMENT3_LOAD0000000A.csv
```

Here, the files SEGMENT1\_LOAD00000001.csv and SEGMENT1\_LOAD00000002.csv are named with the same running source segment index prefix, SEGMENT1. They're named as so because the migrated source data for these two .csv files is associated with the same running source segment index. On the other hand, the migrated data stored in each of the target SEGMENT2\_LOAD00000009.csv and SEGMENT3\_LOAD0000000A.csv files is associated with different running source segment indexes. Each file has its file name prefixed with the name of its running segment index, SEGMENT2 and SEGMENT3.

For the ranges parallel-load type, you define the column names and column values using the columns and boundaries settings of the table-settings rules. With these rules, you can specify partitions corresponding to segment names, as follows.

```
"parallel-load": {  
    "type": "ranges",  
    "columns": [  
        "region",  
        "sale"  
    ],  
    "boundaries": [  
        [  
            "NORTH",  
            "1000"  
        ],  
        [  
            "WEST",  
            "3000"  
        ]  
    ],  
    "segment-names": [  
        "custom_segment1",  
        "custom_segment2",  
        "custom_segment3"  
    ]  
}
```

Here, the segment-names setting defines names for three partitions to migrate data in parallel on the S3 target. The migrated data is parallel-loaded and stored in .csv files under the partition subfolders in order, as follows.

```
.../database_schema_name/table_name/custom_segment1/LOAD[00000001...].csv  
.../database_schema_name/table_name/custom_segment2/LOAD[00000001...].csv  
.../database_schema_name/table_name/custom_segment3/LOAD[00000001...].csv
```

Here, AWS DMS stores a series of .csv files in each of the three partition subfolders. The series of .csv files in each partition subfolder is named incrementally starting from LOAD00000001.csv until all the data is migrated.

In some cases, you might not explicitly name partition subfolders for a ranges parallel-load type using the segment-names setting. In these case, AWS DMS applies the default of creating each series of .csv files under its *table\_name* subfolder. Here, AWS DMS prefixes the file names of each series of .csv files with the name of the running source segment index, as follows.

```
.../database_schema_name/table_name/SEGMENT1_LOAD[00000001...].csv  
.../database_schema_name/table_name/SEGMENT2_LOAD[00000001...].csv  
.../database_schema_name/table_name/SEGMENT3_LOAD[00000001...].csv  
...  
.../database_schema_name/table_name/SEGMENTZ_LOAD[00000001...].csv
```

## Endpoint settings when using Amazon S3 as a target for AWS DMS

You can use endpoint settings to configure your Amazon S3 target similar to using extra connection attributes. You can specify these settings when you create the target endpoint using the `create-endpoint` command in the AWS CLI, with the `--s3-settings` '*json-settings*' option. Here, *json-settings* is a JSON object containing parameters to specify the settings.

You can also specify a .json file containing the same *json-settings* object, for example, as in the following: `--s3-settings` `file:///your-file-path/my_s3_settings.json`. Here, `my_s3_settings.json` is the name of a .json file that contains the same *json-settings* object.

The parameter names for endpoint settings are the same as the names for equivalent extra connections attributes, except that the parameter names for endpoint settings have initial caps. However, not all S3 target endpoint settings using extra connection attributes are available using the `--s3-settings` option of the `create-endpoint` command. For more information about the available settings for the `create-endpoint` CLI command, see [create-endpoint](#) in the *AWS CLI Command Reference* for AWS DMS. For general information about these settings, see the equivalent extra connection attributes in [Extra connection attributes when using Amazon S3 as a target for AWS DMS \(p. 350\)](#).

You can use S3 target endpoint settings to configure the following:

- A custom KMS key to encrypt your S3 target objects.
- Parquet files as the storage format for S3 target objects.
- Change data capture (CDC) including transaction order on the S3 target.

### AWS KMS key settings for data encryption

The following examples show configuring a custom KMS key to encrypt your S3 target objects. To start, you might run the following `create-endpoint` CLI command.

```
aws dms create-endpoint --endpoint-identifier s3-target-endpoint --engine-name s3 --  
endpoint-type target  
--s3-settings '{"ServiceAccessRoleArn": "your-service-access-ARN", "CsvRowDelimiter":  
"\n",  
"CsvDelimiter": ",", "BucketFolder": "your-bucket-folder",  
"BucketName": "your-bucket-name",  
"EncryptionMode": "SSE_KMS",  
"ServerSideEncryptionKmsKeyId": "arn:aws:kms:us-  
east-1:111122223333:key/72abb6fb-1e49-4ac1-9aed-c803dfcc0480"}'
```

Here, the JSON object specified by `--s3-settings` option defines two parameters. One is an `EncryptionMode` parameter with the value `SSE_KMS`. The other is an `ServerSideEncryptionKmsKeyId` parameter with the value of `arn:aws:kms:us-`

east-1:111122223333:key/72abb6fb-1e49-4ac1-9aed-c803dfcc0480. This value is an Amazon Resource Name (ARN) for your custom KMS key. For an S3 target, you also specify additional settings. These identify the server access role, provide delimiters for the default CSV object storage format, and give the bucket location and name to store S3 target objects.

By default, S3 data encryption occurs using S3 server-side encryption. For the previous example's S3 target, this is also equivalent to specifying its endpoint settings as in the following example.

```
aws dms create-endpoint --endpoint-identifier s3-target-endpoint --engine-name s3 --  
endpoint-type target  
--s3-settings '{"ServiceAccessRoleArn": "your-service-access-ARN", "CsvRowDelimiter":  
"\n",  
"CsvDelimiter": ",", "BucketFolder": "your-bucket-folder",  
"BucketName": "your-bucket-name",  
"EncryptionMode": "SSE_S3"}'
```

For more information about working with S3 server-side encryption, see [Protecting data using server-side encryption](#).

**Note**

You can also use the CLI `modify-endpoint` command to change the value of the `EncryptionMode` parameter for an existing endpoint from SSE\_KMS to SSE\_S3. But you can't change the `EncryptionMode` value from SSE\_S3 to SSE\_KMS.

## Settings for using .parquet files to store S3 target objects

The default format for creating S3 target objects is .csv files. The following examples show some endpoint settings for specifying .parquet files as the format for creating S3 target objects. You can specify the .parquet files format with all the defaults, as in the following example.

```
aws dms create-endpoint --endpoint-identifier s3-target-endpoint --engine-name s3 --  
endpoint-type target  
--s3-settings '{"ServiceAccessRoleArn": "your-service-access-ARN", "DataFormat":  
"parquet"}'
```

Here, the `DataFormat` parameter is set to `parquet` to enable the format with all the S3 defaults. These defaults include a dictionary encoding ("`EncodingType: "rle-dictionary"`) that uses a combination of bit-packing and run-length encoding to more efficiently store repeating values.

You can add additional settings for options other than the defaults as in the following example.

```
aws dms create-endpoint --endpoint-identifier s3-target-endpoint --engine-name s3 --  
endpoint-type target  
--s3-settings '{"ServiceAccessRoleArn": "your-service-access-ARN", "BucketFolder": "your-  
bucket-folder",  
"BucketName": "your-bucket-name", "CompressionType": "GZIP", "DataFormat": "parquet",  
"EncodingType": "plain-dictionary", "DictPageSizeLimit": 3,072,000,  
"EnableStatistics": false }'
```

Here, in addition to parameters for several standard S3 bucket options and the `DataFormat` parameter, the following additional .parquet file parameters are set:

- `EncodingType` – Set to a dictionary encoding (`plain-dictionary`) that stores values encountered in each column in a per-column chunk of the dictionary page.
- `DictPageSizeLimit` – Set to a maximum dictionary page size of 3 MB.
- `EnableStatistics` – Disables the default that enables the collection of statistics about Parquet file pages and row groups.

## Capturing data changes (CDC) including transaction order on the S3 target

By default when AWS DMS runs a CDC task, it stores all the row changes logged in your source database (or databases) in one or more files for each table. Each set of files containing changes for the same table reside in a single target directory associated with that table. AWS DMS creates as many target directories as database tables migrated to the Amazon S3 target endpoint. The files are stored on the S3 target in these directories without regard to transaction order. For more information on the file naming conventions, data contents, and format, see [Using Amazon S3 as a target for AWS Database Migration Service \(p. 332\)](#).

To capture source database changes in a manner that also captures the transaction order, you can specify S3 endpoint settings that direct AWS DMS to store the row changes for *all* database tables in one or more .csv files created depending on transaction size. These .csv *transaction files* contain all row changes listed sequentially in transaction order for all tables involved in each transaction. These transaction files reside together in a single *transaction directory* that you also specify on the S3 target. In each transaction file, the transaction operation and the identity of the database and source table for each row change is stored as part of the row data as follows.

```
operation,table_name,database_schema_name,field_value,...
```

Here, *operation* is the transaction operation on the changed row, *table\_name* is the name of the database table where the row is changed, *database\_schema\_name* is the name of the database schema where the table resides, and *field\_value* is the first of one or more field values that specify the data for the row.

The example following of a transaction file shows changed rows for one or more transactions that involve two tables.

```
I,Names_03cdcad11a,rdsTempsdb,13,Daniel
U,Names_03cdcad11a,rdsTempsdb,23,Kathy
D,Names_03cdcad11a,rdsTempsdb,13,Cathy
I,Names_6d152ce62d,rdsTempsdb,15,Jane
I,Names_6d152ce62d,rdsTempsdb,24,Chris
I,Names_03cdcad11a,rdsTempsdb,16,Mike
```

Here, the transaction operation on each row is indicated by I (insert), U (update), or D (delete) in the first column. The table name is the second column value (for example, Names\_03cdcad11a). The name of the database schema is the value of the third column (for example, rdsTempsdb). And the remaining columns are populated with your own row data (for example, 13, Daniel).

In addition, AWS DMS names the transaction files it creates on the Amazon S3 target using a time stamp according to the following naming convention.

```
CDC_TXN-timestamp.csv
```

Here, *timestamp* is the time when the transaction file was created, as in the following example.

```
CDC_TXN-20201117153046033.csv
```

This time stamp in the file name ensures that the transaction files are created and listed in transaction order when you list them in their transaction directory.

### Note

When capturing data changes in transaction order, AWS DMS always stores the row changes in .csv files regardless of the value of the DataFormat S3 setting on the target. AWS DMS doesn't save data changes in transaction order using .parquet files.

To control the frequency of writes to an Amazon S3 target during a data replication task, you can configure the `cdcMaxBatchInterval` and `cdcMinFileSize` extra connection attributes. This can result in better performance when analyzing the data without any additional overhead operations. For more information, see [Extra connection attributes when using Amazon S3 as a target for AWS DMS \(p. 350\)](#)

#### To tell AWS DMS to store all row changes in transaction order

1. Set the `PreserveTransactions` S3 setting on the target to `true`.
2. Set the `CdcPath` S3 setting on the target to a relative folder path where you want AWS DMS to store the `.csv` transaction files.

AWS DMS creates this path either under the default S3 target bucket and working directory or under the bucket and bucket folder that you specify using the `BucketName` and `BucketFolder` S3 settings on the target.

## Extra connection attributes when using Amazon S3 as a target for AWS DMS

You can specify the following options as extra connection attributes. If you have multiple connection attribute settings, separate them from each other by semicolons with no additional white space.

Option	Description
<code>csvNullValue</code>	An optional parameter that specifies how AWS DMS treats null values. While handling the null value, you can use this parameter to pass a user-defined string as null when writing to the target. For example, when target columns are not nullable, you can use this option to differentiate between the empty string value and the null value. So, if you set this parameter value to the empty string ("" or "), AWS DMS treats the empty string as the null value instead of NULL.  Default value: <code>NULL</code>  Valid values: any valid string  Example:  <code>csvNullValue=' ';</code>
<code>addColumnName</code>	An optional parameter that when set to <code>true</code> or <code>y</code> you can use to add column name information to the <code>.csv</code> output file.  You can't use this parameter with <code>PreserveTransactions</code> or <code>CdcPath</code> .  Default value: <code>false</code>  Valid values: <code>true</code> , <code>false</code> , <code>y</code> , <code>n</code>  Example:  <code>addColumnName=true;</code>
<code>bucketFolder</code>	An optional parameter to set a folder name in the S3 bucket. If provided, target objects are created as <code>.csv</code> or <code>.parquet</code> files in the path <code>bucketFolder/schema_name/table_name/</code> . If this parameter isn't specified, then the path used is <code>schema_name/table_name/</code> .

Option	Description
	<p>Example:</p> <pre>bucketFolder=testFolder;</pre>
bucketName	<p>The name of the S3 bucket where S3 target objects are created as .csv or .parquet files.</p> <p>Example:</p> <pre>bucketName=buckettest;</pre>
cannedAclForObjects	<p>A value that enables AWS DMS to specify a predefined (canned) access control list for objects created in the S3 bucket as .csv or .parquet files. For more information about Amazon S3 canned ACLs, see <a href="#">Canned ACL</a> in the <i>Amazon S3 Developer Guide</i>.</p> <p>Default value: NONE</p> <p>Valid values for this attribute are: NONE; PRIVATE; PUBLIC_READ; PUBLIC_READ_WRITE; AUTHENTICATED_READ; AWS_EXEC_READ; BUCKET_OWNER_READ; BUCKET_OWNER_FULL_CONTROL.</p> <p>Example:</p> <pre>cannedAclForObjects=PUBLIC_READ;</pre>
cdcInsertsOnly	<p>An optional parameter during a change data capture (CDC) load to write only INSERT operations to the comma-separated value (.csv) or columnar storage (.parquet) output files. By default (the false setting), the first field in a .csv or .parquet record contains the letter I (INSERT), U (UPDATE), or D (DELETE). This letter indicates whether the row was inserted, updated, or deleted at the source database for a CDC load to the target. If cdcInsertsOnly is set to true or y, only INSERTs from the source database are migrated to the .csv or .parquet file.</p> <p>For .csv format only, how these INSERTS are recorded depends on the value of includeOpForFullLoad. If includeOpForFullLoad is set to true, the first field of every CDC record is set to I to indicate the INSERT operation at the source. If includeOpForFullLoad is set to false, every CDC record is written without a first field to indicate the INSERT operation at the source. For more information about how these parameters work together, see <a href="#">Indicating source DB operations in migrated S3 data (p. 359)</a>.</p> <p>Default value: false</p> <p>Valid values: true, false, y, n</p> <p>Example:</p> <pre>cdcInsertsOnly=true;</pre>

Option	Description
cdcInsertsAndUpdates	<p>Enables a change data capture (CDC) load to write INSERT and UPDATE operations to .csv or .parquet (columnar storage) output files. The default setting is false, but when cdcInsertsAndUpdates is set to true or y, INSERTs and UPDATEs from the source database are migrated to the .csv or .parquet file.</p> <p>For .csv file format only, how these INSERTs and UPDATEs are recorded depends on the value of the includeOpForFullLoad parameter. If includeOpForFullLoad is set to true, the first field of every CDC record is set to either I or U to indicate INSERT and UPDATE operations at the source. But if includeOpForFullLoad is set to false, CDC records are written without an indication of INSERT or UPDATE operations at the source.</p> <p>For more information about how these parameters work together, see <a href="#">Indicating source DB operations in migrated S3 data (p. 359)</a>.</p> <p><b>Note</b>      cdcInsertsOnly and cdcInsertsAndUpdates can't both be set to true for the same endpoint. Set either cdcInsertsOnly or cdcInsertsAndUpdates to true for the same endpoint, but not both.</p> <p>Default value: false</p> <p>Valid values: true, false, y, n</p> <p>Example:</p> <pre>cdcInsertsAndUpdates=true;</pre>

Option	Description
CdcPath	<p>Specifies the folder path of CDC files. For an S3 source, this setting is required if a task captures change data; otherwise, it's optional. If CdcPath is set, DMS reads CDC files from this path and replicates the data changes to the target endpoint. For an S3 target if you set <code>PreserveTransactions</code> to true, DMS verifies that you have set this parameter to a folder path on your S3 target where DMS can save the transaction order for the CDC load. DMS creates this CDC folder path in either your S3 target working directory or the S3 target location specified by <code>BucketFolder</code> and <code>BucketName</code>.</p> <p>You can't use this parameter with <code>DatePartitionEnabled</code> or <code>addColumnName</code>.</p> <p>Type: String</p> <p>For example, if you specify CdcPath as <code>MyChangedData</code>, and you specify <code>BucketName</code> as <code>MyTargetBucket</code> but do not specify <code>BucketFolder</code>, DMS creates the following CDC folder path: <code>MyTargetBucket/MyChangedData</code>.</p> <p>If you specify the same CdcPath, and you specify <code>BucketName</code> as <code>MyTargetBucket</code> and <code>BucketFolder</code> as <code>MyTargetData</code>, DMS creates the following CDC folder path: <code>MyTargetBucket/MyTargetData/MyChangedData</code>.</p> <p><b>Note</b>  This setting is supported in AWS DMS versions 3.4.2 and later.  When capturing data changes in transaction order, DMS always stores the row changes in .csv files regardless of the value of the <code>DataFormat S3</code> setting on the target. DMS doesn't save data changes in transaction order using .parquet files.</p>
cdcMaxBatchInterval	<p>Maximum interval length condition, defined in seconds, to output a file to Amazon S3.</p> <p>Default Value: 60 seconds</p> <p>When <code>cdcMaxBatchInterval</code> is specified and <code>cdcMinFileSize</code> is specified, the file write is triggered by whichever parameter condition is met first.</p>
cdcMinFileSize	<p>Minimum file size condition as defined in kilobytes to output a file to Amazon S3.</p> <p>Default Value: 32000 KB</p> <p>When <code>cdcMinFileSize</code> is specified and <code>cdcMaxBatchInterval</code> is specified, the file write is triggered by whichever parameter condition is met first.</p>

Option	Description
PreserveTransactions	<p>If set to true, DMS saves the transaction order for change data capture (CDC) on the Amazon S3 target specified by CdcPath.</p> <p>You can't use this parameter with DatePartitionEnabled or addColumnName.</p> <p>Type: Boolean</p> <p>When capturing data changes in transaction order, DMS always stores the row changes in .csv files regardless of the value of the DataFormat S3 setting on the target. DMS doesn't save data changes in transaction order using .parquet files.</p> <p><b>Note</b> This setting is supported in AWS DMS versions 3.4.2 and later</p>
includeOpForFullLoad	<p>An optional parameter during a full load to write the INSERT operations to the comma-separated value (.csv) output files only.</p> <p>For full load, records can only be inserted. By default (the false setting), there is no information recorded in these output files for a full load to indicate that the rows were inserted at the source database. If includeOpForFullLoad is set to true or y, the INSERT is recorded as an I annotation in the first field of the .csv file.</p> <p><b>Note</b> This parameter works together with cdcInsertsOnly or cdcInsertsAndUpdates for output to .csv files only. For more information about how these parameters work together, see <a href="#">Indicating source DB operations in migrated S3 data (p. 359)</a>.</p> <p>Default value: false</p> <p>Valid values: true, false, y, n</p> <p>Example:</p> <pre>includeOpForFullLoad=true;</pre>
compressionType	<p>An optional parameter when set to GZIP uses GZIP to compress the target .csv or .parquet files. When this parameter is set to the default, it leaves the files uncompressed.</p> <p>Default value: NONE</p> <p>Valid values: GZIP or NONE</p> <p>Example:</p> <pre>compressionType=GZIP;</pre>
csvDelimiter	<p>The delimiter used to separate columns in .csv source files. The default is a comma (,).</p> <p>Example:</p> <pre>csvDelimiter=,;</pre>

Option	Description
csvRowDelimiter	<p>The delimiter used to separate rows in the .csv source files. The default is a newline (\n).</p> <p>Example:</p> <pre>csvRowDelimiter=\n;</pre>
maxFileSize	<p>A value that specifies the maximum size (in KB) of any .csv file to be created while migrating to an S3 target during full load.</p> <p>Default value: 1,048,576 KB (1 GB)</p> <p>Valid values: 1–1,048,576</p> <p>Example:</p> <pre>maxFileSize=512</pre>
rfc4180	<p>An optional parameter used to set behavior to comply with RFC for data migrated to Amazon S3 using .csv file format only. When this value is set to true or y using Amazon S3 as a target, if the data has quotation marks, commas, or newline characters in it, AWS DMS encloses the entire column with an additional pair of double quotation marks (""). Every quotation mark within the data is repeated twice. This formatting complies with RFC 4180.</p> <p>Default value: true</p> <p>Valid values: true, false, y, n</p> <p>Example:</p> <pre>rfc4180=false;</pre>
encryptionMode	<p>The server-side encryption mode that you want to encrypt your .csv or .parquet object files copied to S3. The valid values are SSE_S3 (S3 server-side encryption) or SSE_KMS (KMS key encryption). If you choose SSE_KMS, set the serverSideEncryptionKmsKeyId parameter to the Amazon Resource Name (ARN) for the KMS key to be used for encryption.</p> <p><b>Note</b>            You can also use the CLI modify-endpoint command to change the value of the encryptionMode attribute for an existing endpoint from SSE_KMS to SSE_S3. But you can't change the encryptionMode value from SSE_S3 to SSE_KMS.</p> <p>Default value: SSE_S3</p> <p>Valid values: SSE_S3 or SSE_KMS</p> <p>Example:</p> <pre>encryptionMode=SSE_S3;</pre>

Option	Description
serverSideEncryption	If <code>KmsKeyId</code> is <code>None</code> , set this parameter to the Amazon Resource Name (ARN) for the KMS key. You can find this ARN by selecting the key alias in the list of AWS KMS keys created for your account. When you create the key, you must associate specific policies and roles associated with this KMS key. For more information, see <a href="#">Creating AWS KMS keys to encrypt Amazon S3 target objects (p. 341)</a> .
	<p>Example:</p> <pre>serverSideEncryptionKmsKeyId=arn:aws:kms:us-east-1:111122223333:key/72abb6fb-1e49-4ac1-9aed-c803dfcc0480;</pre>
dataFormat	<p>The output format for the files that AWS DMS uses to create S3 objects. For Amazon S3 targets, AWS DMS supports either .csv or .parquet files. The .parquet files have a binary columnar storage format with efficient compression options and faster query performance. For more information about .parquet files, see <a href="https://parquet.apache.org/">https://parquet.apache.org/</a>.</p> <p>Default value: csv</p> <p>Valid values: csv or parquet</p> <p>Example:</p> <pre>dataFormat=parquet;</pre>
encodingType	<p>The Parquet encoding type. The encoding type options include the following:</p> <ul style="list-style-type: none"> <li>• <code>rle-dictionary</code> – This dictionary encoding uses a combination of bit-packing and run-length encoding to more efficiently store repeating values.</li> <li>• <code>plain</code> – No encoding.</li> <li>• <code>plain-dictionary</code> – This dictionary encoding builds a dictionary of values encountered in a given column. The dictionary is stored in a dictionary page for each column chunk.</li> </ul> <p>Default value: rle-dictionary</p> <p>Valid values: rle-dictionary, plain, or plain-dictionary</p> <p>Example:</p> <pre>encodingType=plain-dictionary;</pre>
DictPageSizeLimit	<p>The maximum allowed size, in bytes, for a dictionary page in a .parquet file. If a dictionary page exceeds this value, the page uses plain encoding.</p> <p>Default value: 1,024,000 (1 MB)</p> <p>Valid values: Any valid integer value</p> <p>Example:</p> <pre>DictPageSizeLimit=2,048,000;</pre>

Option	Description
rowGroupLength	<p>The number of rows in one row group of a .parquet file.</p> <p>Default value: 10,024 (10 KB)</p> <p>Valid values: Any valid integer value</p> <p>Example:</p> <pre>rowGroupLength=20,048;</pre>
dataPageSize	<p>The maximum allowed size, in bytes, for a data page in a .parquet file.</p> <p>Default value: 1,024,000 (1 MB)</p> <p>Valid values: Any valid integer value</p> <p>Example:</p> <pre>dataPageSize=2,048,000;</pre>
parquetVersion	<p>The version of the .parquet file format.</p> <p>Default value: PARQUET_1_0</p> <p>Valid values: PARQUET_1_0 or PARQUET_2_0</p> <p>Example:</p> <pre>parquetVersion=PARQUET_2_0;</pre>
enableStatistics	<p>Set to true or y to enable statistics about .parquet file pages and row groups.</p> <p>Default value: true</p> <p>Valid values: true, false, y, n</p> <p>Example:</p> <pre>enableStatistics=false;</pre>

Option	Description
timestampColumnName	<p>An optional parameter to include a timestamp column in the S3 target endpoint data.</p> <p>AWS DMS includes an additional STRING column in the .csv or .parquet object files of your migrated data when you set <code>timestampColumnName</code> to a nonblank value.</p> <p>For a full load, each row of this timestamp column contains a timestamp for when the data was transferred from the source to the target by DMS.</p> <p>For a CDC load, each row of the timestamp column contains the timestamp for the commit of that row in the source database.</p> <p>The string format for this timestamp column value is <code>yyyy-MM-dd HH:mm:ss .SSSSSS</code>. By default, the precision of this value is in microseconds. For a CDC load, the rounding of the precision depends on the commit timestamp supported by DMS for the source database.</p> <p>When the <code>addColumnName</code> parameter is set to <code>true</code>, DMS also includes the name for the timestamp column that you set as the nonblank value of <code>timestampColumnName</code>.</p> <p>Example:</p> <pre>timestampColumnName=TIMESTAMP;</pre>
useTaskStartTimeForFullLoadTimestamp	<p><del>This is a deprecated parameter.</del> This parameter uses the task start time as the timestamp column value instead of the time data is written to target. For full load, when <code>useTaskStartTimeForFullLoadTimestamp</code> is set to <code>true</code>, each row of the timestamp column contains the task start time. For CDC loads, each row of the timestamp column contains the transaction commit time.</p> <p>When <code>useTaskStartTimeForFullLoadTimestamp</code> is set to <code>false</code>, the full load timestamp in the timestamp column increments with the time data arrives at the target.</p> <p>Default value: <code>false</code></p> <p>Valid values: <code>true</code>, <code>false</code></p> <p>Example: <code>useTaskStartTimeForFullLoadTimestamp=true;</code></p> <p><code>useTaskStartTimeForFullLoadTimestamp=true</code> helps make the S3 target <code>timestampColumnName</code> for a full load sortable with <code>timestampColumnName</code> for a CDC load.</p>

Option	Description
parquetTimestampInMillisecond	<p>An optional parameter that specifies the precision of any TIMESTAMP column values written to an S3 object file in .parquet format.</p> <p>When this attribute is set to true or y, AWS DMS writes all TIMESTAMP columns in a .parquet formatted file with millisecond precision. Otherwise, DMS writes them with microsecond precision.</p> <p>Currently, Amazon Athena and AWS Glue can handle only millisecond precision for TIMESTAMP values. Set this attribute to true for .parquet formatted S3 endpoint object files only if you plan to query or process the data with Athena or AWS Glue.</p> <p><b>Note</b></p> <ul style="list-style-type: none"> <li>• AWS DMS writes any TIMESTAMP column values written to an S3 file in .csv format with microsecond precision.</li> <li>• The setting of this attribute has no effect on the string format of the timestamp column value inserted by setting the timestampColumnName attribute.</li> </ul> <p>Default value: false</p> <p>Valid values: true, false, y, n</p> <p>Example:</p> <pre>parquetTimestampInMillisecond=true;</pre>

## Indicating source DB operations in migrated S3 data

When AWS DMS migrates records to an S3 target, it can create an additional field in each migrated record. This additional field indicates the operation applied to the record at the source database. How AWS DMS creates and sets this first field depends on the migration task type and settings of `includeOpForFullLoad`, `cdcInsertsOnly`, and `cdcInsertsAndUpdates`.

For a full load when `includeOpForFullLoad` is true, AWS DMS always creates an additional first field in each .csv record. This field contains the letter I (INSERT) to indicate that the row was inserted at the source database. For a CDC load when `cdcInsertsOnly` is false (the default), AWS DMS also always creates an additional first field in each .csv or .parquet record. This field contains the letter I (INSERT), U (UPDATE), or D (DELETE) to indicate whether the row was inserted, updated, or deleted at the source database.

In the following table, you can see how the settings of the `includeOpForFullLoad` and `cdcInsertsOnly` attributes work together to affect the setting of migrated records.

With these parameter settings		DMS sets target records as follows for .csv and .parquet output	
true	true	Added first field value set to I	Added first field value set to I
false	false	No added field	Added first field value set to I, U, or D

With these parameter settings		DMS sets target records as follows for .csv and .parquet output	
false	true	No added field	No added field
true	false	Added first field value set to I	Added first field value set to I, U, or D

When `includeOpForFullLoad` and `cdcInsertsOnly` are set to the same value, the target records are set according to the attribute that controls record settings for the current migration type. That attribute is `includeOpForFullLoad` for full load and `cdcInsertsOnly` for CDC load.

When `includeOpForFullLoad` and `cdcInsertsOnly` are set to different values, AWS DMS makes the target record settings consistent for both CDC and full load. It does this by making the record settings for a CDC load conform to the record settings for any earlier full load specified by `includeOpForFullLoad`.

In other words, suppose that a full load is set to add a first field to indicate an inserted record. In this case, a following CDC load is set to add a first field that indicates an inserted, updated, or deleted record as appropriate at the source. In contrast, suppose that a full load is set to *not* add a first field to indicate an inserted record. In this case, a CDC load is also set to not add a first field to each record regardless of its corresponding record operation at the source.

Similarly, how DMS creates and sets an additional first field depends on the settings of `includeOpForFullLoad` and `cdcInsertsAndUpdates`. In the following table, you can see how the settings of the `includeOpForFullLoad` and `cdcInsertsAndUpdates` attributes work together to affect the setting of migrated records in this format.

With these parameter settings		DMS sets target records as follows for .csv output	
true	true	Added first field value set to I	Added first field value set to I or U
false	false	No added field	Added first field value set to I, U, or D
false	true	No added field	Added first field value set to I or U
true	false	Added first field value set to I	Added first field value set to I, U, or D

## Target data types for S3 Parquet

The following table shows the Parquet target data types that are supported when using AWS DMS and the default mapping from AWS DMS data types.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

AWS DMS data type	S3 parquet data type
BYTES	BINARY
DATE	DATE32

AWS DMS data type	S3 parquet data type
TIME	TIME32
DATETIME	TIMESTAMP
INT1	INT8
INT2	INT16
INT4	INT32
INT8	INT64
NUMERIC	DECIMAL
REAL4	FLOAT
REAL8	DOUBLE
STRING	STRING
UINT1	UINT8
UINT2	UINT16
UINT4	UINT32
UINT8	UINT64
WSTRING	STRING
BLOB	BINARY
NCLOB	STRING
CLOB	STRING
BOOLEAN	BOOL

## Using an Amazon DynamoDB database as a target for AWS Database Migration Service

You can use AWS DMS to migrate data to an Amazon DynamoDB table. Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. AWS DMS supports using a relational database or MongoDB as a source.

In DynamoDB, tables, items, and attributes are the core components that you work with. A *table* is a collection of items, and each *item* is a collection of attributes. DynamoDB uses primary keys, called partition keys, to uniquely identify each item in a table. You can also use keys and secondary indexes to provide more querying flexibility.

You use object mapping to migrate your data from a source database to a target DynamoDB table. Object mapping enables you to determine where the source data is located in the target.

When AWS DMS creates tables on an DynamoDB target endpoint, it creates as many tables as in the source database endpoint. AWS DMS also sets several DynamoDB parameter values. The cost for the table creation depends on the amount of data and the number of tables to be migrated.

**Note**

The **SSL Mode** option on the AWS DMS console or API doesn't apply to some data streaming and NoSQL services like Kinesis and DynamoDB. They are secure by default, so AWS DMS shows the SSL mode setting is equal to none (**SSL Mode=None**). You don't need to provide any additional configuration for your endpoint to make use of SSL. For example, when using DynamoDB as a target endpoint, it is secure by default. All API calls to DynamoDB use SSL, so there is no need for an additional SSL option in the AWS DMS endpoint. You can securely put data and retrieve data through SSL endpoints using the HTTPS protocol, which AWS DMS uses by default when connecting to a DynamoDB database.

To help increase the speed of the transfer, AWS DMS supports a multithreaded full load to a DynamoDB target instance. DMS supports this multithreading with task settings that include the following:

- **MaxFullLoadSubTasks** – Use this option to indicate the maximum number of source tables to load in parallel. DMS loads each table into its corresponding DynamoDB target table using a dedicated subtask. The default value is 8. The maximum value is 49.
- **ParallelLoadThreads** – Use this option to specify the number of threads that AWS DMS uses to load each table into its DynamoDB target table. The default value is 0 (single-threaded). The maximum value is 200. You can ask to have this maximum limit increased.

**Note**

DMS assigns each segment of a table to its own thread for loading. Therefore, set **ParallelLoadThreads** to the maximum number of segments that you specify for a table in the source.

- **ParallelLoadBufferSize** – Use this option to specify the maximum number of records to store in the buffer that the parallel load threads use to load data to the DynamoDB target. The default value is 50. The maximum value is 1,000. Use this setting with **ParallelLoadThreads**. **ParallelLoadBufferSize** is valid only when there is more than one thread.
- **Table-mapping** settings for individual tables – Use **table-settings** rules to identify individual tables from the source that you want to load in parallel. Also use these rules to specify how to segment the rows of each table for multithreaded loading. For more information, see [Table and collection settings rules and operations \(p. 523\)](#).

**Note**

When AWS DMS sets DynamoDB parameter values for a migration task, the default Read Capacity Units (RCU) parameter value is set to 200.

The Write Capacity Units (WCU) parameter value is also set, but its value depends on several other settings:

- The default value for the WCU parameter is 200.
- If the **ParallelLoadThreads** task setting is set greater than 1 (the default is 0), then the WCU parameter is set to 200 times the **ParallelLoadThreads** value.
- Standard AWS DMS usage fees apply to resources you use.

## Migrating from a relational database to a DynamoDB table

AWS DMS supports migrating data to DynamoDB scalar data types. When migrating from a relational database like Oracle or MySQL to DynamoDB, you might want to restructure how you store this data.

Currently AWS DMS supports single table to single table restructuring to DynamoDB scalar type attributes. If you are migrating data into DynamoDB from a relational database table, you take data from a table and reformat it into DynamoDB scalar data type attributes. These attributes can accept data from multiple columns, and you can map a column to an attribute directly.

AWS DMS supports the following DynamoDB scalar data types:

- String
- Number
- Boolean

**Note**

NULL data from the source are ignored on the target.

## Prerequisites for using DynamoDB as a target for AWS Database Migration Service

Before you begin to work with a DynamoDB database as a target for AWS DMS, make sure that you create an IAM role. This IAM role should allow AWS DMS to assume and grants access to the DynamoDB tables that are being migrated into. The minimum set of access permissions is shown in the following IAM policy.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "dms.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

The role that you use for the migration to DynamoDB must have the following permissions.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb:PutItem",  
                "dynamodb CreateTable",  
                "dynamodb:DescribeTable",  
                "dynamodb>DeleteTable",  
                "dynamodb>DeleteItem",  
                "dynamodb:UpdateItem"  
            ],  
            "Resource": [  
                "arn:aws:dynamodb:us-west-2:account-id:table/name1",  
                "arn:aws:dynamodb:us-west-2:account-id:table/OtherName*",  
                "arn:aws:dynamodb:us-west-2:account-id:table/awsdms_apply_exceptions",  
                "arn:aws:dynamodb:us-west-2:account-id:table/awsdms_full_load_exceptions"  
            ]  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "dynamodb>ListTables"  
            ],  
            "Resource": "*"  
        }  
    ]  
}
```

```
}
```

## Limitations when using DynamoDB as a target for AWS Database Migration Service

The following limitations apply when using DynamoDB as a target:

- DynamoDB limits the precision of the Number data type to 38 places. Store all data types with a higher precision as a String. You need to explicitly specify this using the object-mapping feature.
- Because DynamoDB doesn't have a Date data type, data using the Date data type are converted to strings.
- DynamoDB doesn't allow updates to the primary key attributes. This restriction is important when using ongoing replication with change data capture (CDC) because it can result in unwanted data in the target. Depending on how you have the object mapping, a CDC operation that updates the primary key can do one of two things. It can either fail or insert a new item with the updated primary key and incomplete data.
- AWS DMS only supports replication of tables with noncomposite primary keys. The exception is if you specify an object mapping for the target table with a custom partition key or sort key, or both.
- AWS DMS doesn't support LOB data unless it is a CLOB. AWS DMS converts CLOB data into a DynamoDB string when migrating the data.
- When you use DynamoDB as target, only the Apply Exceptions control table (`dmslogs.awsdms_apply_exceptions`) is supported. For more information about control tables, see [Control table task settings \(p. 461\)](#).

## Using object mapping to migrate data to DynamoDB

AWS DMS uses table-mapping rules to map data from the source to the target DynamoDB table. To map data to a DynamoDB target, you use a type of table-mapping rule called *object-mapping*. Object mapping lets you define the attribute names and the data to be migrated to them. You must have selection rules when you use object mapping.

DynamoDB doesn't have a preset structure other than having a partition key and an optional sort key. If you have a noncomposite primary key, AWS DMS uses it. If you have a composite primary key or you want to use a sort key, define these keys and the other attributes in your target DynamoDB table.

To create an object-mapping rule, you specify the `rule-type` as *object-mapping*. This rule specifies what type of object mapping you want to use.

The structure for the rule is as follows:

```
{ "rules": [
    {
        "rule-type": "object-mapping",
        "rule-id": "<id>",
        "rule-name": "<name>",
        "rule-action": "<valid object-mapping rule action>",
        "object-locator": {
            "schema-name": "<case-sensitive schema name>",
            "table-name": ""
        },
        "target-table-name": "<table_name>"
    }
]}
```

AWS DMS currently supports **map-record-to-record** and **map-record-to-document** as the only valid values for the **rule-action** parameter. These values specify what AWS DMS does by default to records that aren't excluded as part of the **exclude-columns** attribute list. These values don't affect the attribute mappings in any way.

- You can use **map-record-to-record** when migrating from a relational database to DynamoDB. It uses the primary key from the relational database as the partition key in DynamoDB and creates an attribute for each column in the source database. When using **map-record-to-record**, for any column in the source table not listed in the **exclude-columns** attribute list, AWS DMS creates a corresponding attribute on the target DynamoDB instance. It does so regardless of whether that source column is used in an attribute mapping.
- You use **map-record-to-document** to put source columns into a single, flat DynamoDB map on the target using the attribute name **\_doc**. When using **map-record-to-document**, AWS DMS places the data into a single, flat, DynamoDB map attribute on the source. This attribute is called **\_doc**. This placement applies to any column in the source table not listed in the **exclude-columns** attribute list.

One way to understand the difference between the **rule-action** parameters **map-record-to-record** and **map-record-to-document** is to see the two parameters in action. For this example, assume that you are starting with a relational database table row with the following structure and data:

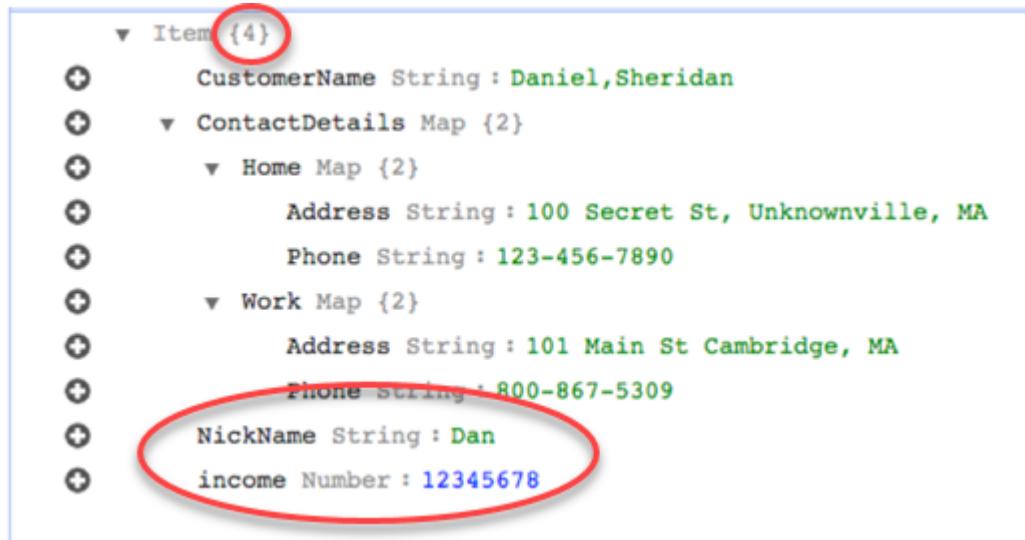
FirstName	LastName	NickName	WorkAddress	WorkPhone	HomeAddress	HomePhone
Daniel	Sheridan	Dan	101 Main St Cambridge, MA	800-867-5309	100 Secret St, Unknownville, MA	123-456-7890

To migrate this information to DynamoDB, you create rules to map the data into a DynamoDB table item. Note the columns listed for the **exclude-columns** parameter. These columns are not directly mapped over to the target. Instead, attribute mapping is used to combine the data into new items, such as where *FirstName* and *LastName* are grouped together to become *CustomerName* on the DynamoDB target. *NickName* and *income* are not excluded.

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "2",
            "rule-name": "TransformToDDB",
            "rule-action": "map-record-to-record",
            "object-locator": {
                "schema-name": "test",
                "table-name": "customer"
            },
            "target-table-name": "customer_t",
            "mapping-parameters": {
                "partition-key-name": "CustomerName",
                "exclude-columns": [
                    "FirstName",
                    "LastName",
                    "HomeAddress",
                    "HomePhone",
                    "WorkAddress",
                    "NickName",
                    "income"
                ]
            }
        }
    ]
}
```

```
"WorkPhone"
],
"attribute-mappings": [
{
  "target-attribute-name": "CustomerName",
  "attribute-type": "scalar",
  "attribute-sub-type": "string",
  "value": "${FirstName},${LastName}"
},
{
  "target-attribute-name": "ContactDetails",
  "attribute-type": "document",
  "attribute-sub-type": "dynamodb-map",
  "value": {
    "M": {
      "Home": {
        "M": {
          "Address": {
            "S": "${HomeAddress}"
          },
          "Phone": {
            "S": "${HomePhone}"
          }
        }
      },
      "Work": {
        "M": {
          "Address": {
            "S": "${WorkAddress}"
          },
          "Phone": {
            "S": "${WorkPhone}"
          }
        }
      }
    }
  }
}
]
```

By using the `rule-action` parameter *map-record-to-record*, the data for *NickName* and *income* are mapped to items of the same name in the DynamoDB target.



However, suppose that you use the same rules but change the rule-action parameter to *map-record-to-document*. In this case, the columns not listed in the exclude-columns parameter, *NickName* and *income*, are mapped to a *\_doc* item.



## Using custom condition expressions with object mapping

You can use a feature of DynamoDB called conditional expressions to manipulate data that is being written to a DynamoDB table. For more information about condition expressions in DynamoDB, see [Condition expressions](#).

A condition expression member consists of:

- an expression (required)
- expression attribute values (optional) . Specifies a DynamoDB json structure of the attribute value
- expression attribute names (optional)
- options for when to use the condition expression (optional). The default is apply-during-cdc = false and apply-during-full-load = true

The structure for the rule is as follows:

```
"target-table-name": "customer_t",
"mapping-parameters": {
    "partition-key-name": "CustomerName",
    "condition-expression": {
        "expression": "<conditional expression>",
        "expression-attribute-values": [
            {
                "name": "<attribute name>",
                "value": "<attribute value>"
            }
        ],
        "apply-during-cdc": <optional Boolean value>,
        "apply-during-full-load": <optional Boolean value>
    }
}
```

The following sample highlights the sections used for condition expression.

```
{
    "rules": [
        {
            "rule-type": "object-mapping",
            "rule-id": "1",
            "rule-name": "TransformToDDB",
            "rule-action": "map-record-to-record",
            "object-locator": {
                "schema-name": "test",
                "table-name": "customer",
            },
            "target-table-name": "customer_t",
            "mapping-parameters": {
                "partition-key-name": "CustomerName",
                "condition-expression": {
                    "expression": "attribute_not_exists(version) or version <= :record_version",
                    "expression-attribute-values": [
                        {
                            "name": ":record_version",
                            "value": {"N": "${version}"}
                        }
                    ],
                    "apply-during-cdc": true,
                    "apply-during-full-load": true
                }
            },
            "attribute-mappings": [
                {
                    "target-attribute-name": "CustomerName",
                    "attribute-type": "scalar",
                    "attribute-sub-type": "string",
                    "value": "${FirstName}, ${LastName}"
                }
            ]
        }
    ]
}
```

Object mapping section defines name, rule-action, and object locator information

Condition expression

Options

## Using attribute mapping with object mapping

Attribute mapping lets you specify a template string using source column names to restructure data on the target. There is no formatting done other than what the user specifies in the template.

The following example shows the structure of the source database and the desired structure of the DynamoDB target. First is shown the structure of the source, in this case an Oracle database, and then the desired structure of the data in DynamoDB. The example concludes with the JSON used to create the desired target structure.

The structure of the Oracle data is as follows:

First	Last	Stn	HomeA	HomeP	WorkAddress	Work	DateOfBirth
Primary Key			N/A				
Randy	Mars	15	221B Baker Street	1234567890	Spooner Street, Quahog		02/29/1988

The structure of the DynamoDB data is as follows:

Custom	StoreId	ContactDetails	DateOfBirth
Partition Key	Sort Key	N/A	
Randy,	Mars	<pre>{     "Name": "Randy",     "Home": {         "Address": "221B Baker Street",         "Phone": 1234567890     },     "Work": {         "Address": "31 Spooner Street,         Quahog",         "Phone": 9876541230     } }</pre>	02/29/1988

The following JSON shows the object mapping and column mapping used to achieve the DynamoDB structure:

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "2",
            "rule-name": "TransformToDDB",
            "transform": {
                "source": {
                    "columns": [
                        "First", "Last", "Stn", "HomeA", "HomeP", "WorkAddress", "Work"
                    ],
                    "row": [
                        "Randy", "Mars", "15", "221B Baker Street", "1234567890", "Spooner Street, Quahog", null
                    ]
                },
                "target": {
                    "columns": [
                        "Custom", "StoreId", "ContactDetails", "DateOfBirth"
                    ],
                    "row": [
                        "Randy", "Mars", {
                            "Name": "Randy",
                            "Home": {
                                "Address": "221B Baker Street",
                                "Phone": 1234567890
                            },
                            "Work": {
                                "Address": "31 Spooner Street, Quahog",
                                "Phone": 9876541230
                            }
                        }, "02/29/1988"
                    ]
                }
            }
        }
    ]
}
```

```

    "rule-action": "map-record-to-record",
    "object-locator": {
        "schema-name": "test",
        "table-name": "customer"
    },
    "target-table-name": "customer_t",
    "mapping-parameters": {
        "partition-key-name": "CustomerName",
        "sort-key-name": "StoreId",
        "exclude-columns": [
            "FirstName",
            "LastName",
            "HomeAddress",
            "HomePhone",
            "WorkAddress",
            "WorkPhone"
        ],
        "attribute-mappings": [
            {
                "target-attribute-name": "CustomerName",
                "attribute-type": "scalar",
                "attribute-sub-type": "string",
                "value": "${FirstName},${LastName}"
            },
            {
                "target-attribute-name": "StoreId",
                "attribute-type": "scalar",
                "attribute-sub-type": "string",
                "value": "${StoreId}"
            },
            {
                "target-attribute-name": "ContactDetails",
                "attribute-type": "scalar",
                "attribute-sub-type": "string",
                "value": "{\"Name\":\"${FirstName}\",\"Home\":{\"Address\":"
                    +"${HomeAddress}\",\"Phone\":\"${HomePhone}\\"}, \"Work\":{\"Address\":\"${WorkAddress}\",
                    \"Phone\":\"${WorkPhone}\\"}}"
            }
        ]
    }
}

```

Another way to use column mapping is to use DynamoDB format as your document type. The following code example uses *dynamodb-map* as the attribute-sub-type for attribute mapping.

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "2",

```

```

"rule-name": "TransformToDDB",
"rule-action": "map-record-to-record",
"object-locator": {
    "schema-name": "test",
    "table-name": "customer"
},
"target-table-name": "customer_t",
"mapping-parameters": {
    "partition-key-name": "CustomerName",
    "sort-key-name": "StoreId",
    "exclude-columns": [
        "FirstName",
        "LastName",
        "HomeAddress",
        "HomePhone",
        "WorkAddress",
        "WorkPhone"
    ],
    "attribute-mappings": [
        {
            "target-attribute-name": "CustomerName",
            "attribute-type": "scalar",
            "attribute-sub-type": "string",
            "value": "${FirstName},${LastName}"
        },
        {
            "target-attribute-name": "StoreId",
            "attribute-type": "scalar",
            "attribute-sub-type": "string",
            "value": "${StoreId}"
        },
        {
            "target-attribute-name": "ContactDetails",
            "attribute-type": "document",
            "attribute-sub-type": "dynamodb-map",
            "value": {
                "M": {
                    "Name": {
                        "S": "${FirstName}"
                    },
                    "Home": {
                        "M": {
                            "Address": {
                                "S": "${HomeAddress}"
                            },
                            "Phone": {
                                "S": "${HomePhone}"
                            }
                        }
                    },
                    "Work": {
                        "M": {
                            "Address": {
                                "S": "${WorkAddress}"
                            },
                            "Phone": {
                                "S": "${WorkPhone}"
                            }
                        }
                    }
                }
            }
        }
    ]
}
}

```

```
    ]
}
```

As an alternative to dynamodb-map, you can use dynamodb-list as the attribute-sub-type for attribute mapping, as shown in the following example.

```
{
  "target-attribute-name": "ContactDetailsList",
  "attribute-type": "document",
  "attribute-sub-type": "dynamodb-list",
  "value": {
    "L": [
      {
        "N": "${FirstName}"
      },
      {
        "N": "${HomeAddress}"
      },
      {
        "N": "${HomePhone}"
      },
      {
        "N": "${WorkAddress}"
      },
      {
        "N": "${WorkPhone}"
      }
    ]
  }
}
```

### Example 1: Using attribute mapping with object mapping

The following example migrates data from two MySQL database tables, *nfl\_data* and *sport\_team*, to two DynamoDB table called *NFLTeams* and *SportTeams*. The structure of the tables and the JSON used to map the data from the MySQL database tables to the DynamoDB tables are shown following.

The structure of the MySQL database table *nfl\_data* is shown below:

```
mysql> desc nfl_data;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| Position | varchar(5) | YES |   | NULL |   |
| player_number | smallint(6) | YES |   | NULL |   |
| Name | varchar(40) | YES |   | NULL |   |
| status | varchar(10) | YES |   | NULL |   |
| stat1 | varchar(10) | YES |   | NULL |   |
| stat1_val | varchar(10) | YES |   | NULL |   |
| stat2 | varchar(10) | YES |   | NULL |   |
| stat2_val | varchar(10) | YES |   | NULL |   |
| stat3 | varchar(10) | YES |   | NULL |   |
| stat3_val | varchar(10) | YES |   | NULL |   |
| stat4 | varchar(10) | YES |   | NULL |   |
| stat4_val | varchar(10) | YES |   | NULL |   |
| team | varchar(10) | YES |   | NULL |   |
+-----+-----+-----+-----+-----+
```

The structure of the MySQL database table *sport\_team* is shown below:

```
mysql> desc sport_team;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| id | mediumint(9) | NO | PRI | NULL | auto_increment |
| name | varchar(30) | NO | | NULL |
| abbreviated_name | varchar(10) | YES | | NULL |
| home_field_id | smallint(6) | YES | MUL | NULL |
| sport_type_name | varchar(15) | NO | MUL | NULL |
| sport_league_short_name | varchar(10) | NO | | NULL |
| sport_division_short_name | varchar(10) | YES | | NULL |
+-----+-----+-----+-----+-----+
```

The table-mapping rules used to map the two tables to the two DynamoDB tables is shown below:

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "dms_sample",
        "table-name": "nfl_data"
      },
      "rule-action": "include"
    },
    {
      "rule-type": "selection",
      "rule-id": "2",
      "rule-name": "2",
      "object-locator": {
        "schema-name": "dms_sample",
        "table-name": "sport_team"
      },
      "rule-action": "include"
    },
    {
      "rule-type": "object-mapping",
      "rule-id": "3",
      "rule-name": "MapNFLData",
      "rule-action": "map-record-to-record",
      "object-locator": {
        "schema-name": "dms_sample",
        "table-name": "nfl_data"
      },
      "target-table-name": "NFLTeams",
      "mapping-parameters": {
        "partition-key-name": "Team",
        "sort-key-name": "PlayerName",
        "exclude-columns": [
          "player_number", "team", "Name"
        ],
        "attribute-mappings": [
          {
            "target-attribute-name": "Team",
            "attribute-type": "scalar",
            "attribute-sub-type": "string",
            "value": "${team}"
          }
        ]
      }
    }
  ]
}
```

```
{
    "target-attribute-name": "PlayerName",
    "attribute-type": "scalar",
    "attribute-sub-type": "string",
    "value": "${Name}"
},
{
    "target-attribute-name": "PlayerInfo",
    "attribute-type": "scalar",
    "attribute-sub-type": "string",
    "value": "{\"Number\": \"${player_number}\", \"Position\": \"${Position}\",
\"Status\": \"${status}\", \"Stats\": {\"Stat1\": \"${stat1}:${stat1_val}\", \"Stat2\":
\"${stat2}:${stat2_val}\", \"Stat3\": \"${stat3}:${stat3_val}\", \"Stat4\": \"${stat4}:${stat4_val}\"
}"
}
]
},
{
    "rule-type": "object-mapping",
    "rule-id": "4",
    "rule-name": "MapSportTeam",
    "rule-action": "map-record-to-record",
    "object-locator": {
        "schema-name": "dms_sample",
        "table-name": "sport_team"
    },
    "target-table-name": "SportTeams",
    "mapping-parameters": {
        "partition-key-name": "TeamName",
        "exclude-columns": [
            "name", "id"
        ],
        "attribute-mappings": [
            {
                "target-attribute-name": "TeamName",
                "attribute-type": "scalar",
                "attribute-sub-type": "string",
                "value": "${name}"
            },
            {
                "target-attribute-name": "TeamInfo",
                "attribute-type": "scalar",
                "attribute-sub-type": "string",
                "value": "{\"League\": \"${sport_league_short_name}\", \"Division\":
\"${sport_division_short_name}\"
}"
            }
        ]
    }
}
}
```

The sample output for the *NFLTeams* DynamoDB table is shown below:

```
"PlayerInfo": "{\"Number\": \"6\", \"Position\": \"P\", \"Status\": \"ACT\", \"Stats\":
{\"Stat1\": \"PUNTS:73\", \"Stat2\": \"AVG:46\", \"Stat3\": \"LNG:67\", \"Stat4\": \"IN
20:31\"}",
"PlayerName": "Allen, Ryan",
"Position": "P",
"stat1": "PUNTS",
"stat1_val": "73",
"stat2": "AVG",
```

```

    "stat2_val": "46",
    "stat3": "LNG",
    "stat3_val": "67",
    "stat4": "IN 20",
    "stat4_val": "31",
    "status": "ACT",
    "Team": "NE"
}

```

The sample output for the *SportsTeams* *DynamoDB* table is shown below:

```

{
    "abbreviated_name": "IND",
    "home_field_id": 53,
    "sport_division_short_name": "AFC South",
    "sport_league_short_name": "NFL",
    "sport_type_name": "football",
    "TeamInfo": "{\"League\": \"NFL\", \"Division\": \"AFC South\"}",
    "TeamName": "Indianapolis Colts"
}

```

## Target data types for DynamoDB

The DynamoDB endpoint for AWS DMS supports most DynamoDB data types. The following table shows the Amazon AWS DMS target data types that are supported when using AWS DMS and the default mapping from AWS DMS data types.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

When AWS DMS migrates data from heterogeneous databases, we map data types from the source database to intermediate data types called AWS DMS data types. We then map the intermediate data types to the target data types. The following table shows each AWS DMS data type and the data type it maps to in DynamoDB:

AWS DMS data type	DynamoDB data type
String	String
WString	String
Boolean	Boolean
Date	String
DateTime	String
INT1	Number
INT2	Number
INT4	Number
INT8	Number
Numeric	Number

AWS DMS data type	DynamoDB data type
Real4	Number
Real8	Number
UINT1	Number
UINT2	Number
UINT4	Number
UINT8	Number
CLOB	String

## Using Amazon Kinesis Data Streams as a target for AWS Database Migration Service

You can use AWS DMS to migrate data to an Amazon Kinesis data stream. Amazon Kinesis data streams are part of the Amazon Kinesis Data Streams service. You can use Kinesis data streams to collect and process large streams of data records in real time.

A Kinesis data stream is made up of shards. *Shards* are uniquely identified sequences of data records in a stream. For more information on shards in Amazon Kinesis Data Streams, see [Shard](#) in the *Amazon Kinesis Data Streams Developer Guide*.

AWS Database Migration Service publishes records to a Kinesis data stream using JSON. During conversion, AWS DMS serializes each record from the source database into an attribute-value pair in JSON format or a JSON\_UNFORMATTED message format. A JSON\_UNFORMATTED message format is a single line JSON string with new line delimiter. It allows Amazon Kinesis Data Firehose to deliver Kinesis data to an Amazon S3 destination, and then query it using various query engines including Amazon Athena.

You use object mapping to migrate your data from any supported data source to a target stream. With object mapping, you determine how to structure the data records in the stream. You also define a partition key for each table, which Kinesis Data Streams uses to group the data into its shards.

When AWS DMS creates tables on an Kinesis Data Streams target endpoint, it creates as many tables as in the source database endpoint. AWS DMS also sets several Kinesis Data Streams parameter values. The cost for the table creation depends on the amount of data and the number of tables to be migrated.

**Note**

The **SSL Mode** option on the AWS DMS console or API doesn't apply to some data streaming and NoSQL services like Kinesis and DynamoDB. They are secure by default, so AWS DMS shows the SSL mode setting is equal to none (**SSL Mode=None**). You don't need to provide any additional configuration for your endpoint to make use of SSL. For example, when using Kinesis as a target endpoint, it is secure by default. All API calls to Kinesis use SSL, so there is no need for an additional SSL option in the AWS DMS endpoint. You can securely put data and retrieve data through SSL endpoints using the HTTPS protocol, which AWS DMS uses by default when connecting to a Kinesis Data Stream.

### Kinesis Data Streams endpoint settings

When you use Kinesis Data Streams target endpoints, you can get transaction and control details using the `KinesisSettings` option in the AWS DMS API.

You can set connection settings in the following ways:

- In the AWS DMS console, using endpoint settings.
- In the CLI, using the `kinesis-settings` option of the [CreateEndpoint](#) command.

In the CLI, use the following request parameters of the `kinesis-settings` option:

**Note**

Support for the `IncludeNullAndEmpty` endpoint setting is available in AWS DMS version 3.4.1 and higher. But support for the other following endpoint settings for Kinesis Data Streams targets is available in AWS DMS.

- `MessageFormat` – The output format for the records created on the endpoint. The message format is `JSON` (default) or `JSON_UNFORMATTED` (a single line with no tab).
- `IncludeControlDetails` – Shows detailed control information for table definition, column definition, and table and column changes in the Kinesis message output. The default is `false`.
- `IncludeNullAndEmpty` – Include NULL and empty columns in the target. The default is `false`.
- `IncludePartitionValue` – Shows the partition value within the Kinesis message output, unless the partition type is `schema-table-type`. The default is `false`.
- `IncludeTableAlterOperations` – Includes any data definition language (DDL) operations that change the table in the control data, such as `rename-table`, `drop-table`, `add-column`, `drop-column`, and `rename-column`. The default is `false`.
- `IncludeTransactionDetails` – Provides detailed transaction information from the source database. This information includes a commit timestamp, a log position, and values for `transaction_id`, `previous_transaction_id`, and `transaction_record_id` (the record offset within a transaction). The default is `false`.
- `PartitionIncludeSchemaTable` – Prefixes schema and table names to partition values, when the partition type is `primary-key-type`. Doing this increases data distribution among Kinesis shards. For example, suppose that a SysBench schema has thousands of tables and each table has only limited range for a primary key. In this case, the same primary key is sent from thousands of tables to the same shard, which causes throttling. The default is `false`.

The following example shows the `kinesis-settings` option in use with an example `create-endpoint` command issued using the AWS CLI.

```
aws dms create-endpoint --endpoint-identifier=$target_name --engine-name kinesis --  
endpoint-type target  
--region us-east-1 --kinesis-settings ServiceAccessRoleArn=arn:aws:iam::333333333333:role/  
dms-kinesis-role,  
StreamArn=arn:aws:kinesis:us-east-1:333333333333:stream/dms-kinesis-target-  
doc,MessageFormat=json-unformatted,  
IncludeControlDetails=true,IncludeTransactionDetails=true,IncludePartitionValue=true,PartitionIncludeSo  
IncludeTableAlterOperations=true
```

### Multithreaded full load task settings

To help increase the speed of the transfer, AWS DMS supports a multithreaded full load to a Kinesis Data Streams target instance. DMS supports this multithreading with task settings that include the following:

- `MaxFullLoadSubTasks` – Use this option to indicate the maximum number of source tables to load in parallel. DMS loads each table into its corresponding Kinesis target table using a dedicated subtask. The default is 8; the maximum value is 49.
- `ParallelLoadThreads` – Use this option to specify the number of threads that AWS DMS uses to load each table into its Kinesis target table. The maximum value for a Kinesis Data Streams target is 32. You can ask to have this maximum limit increased.

- **ParallelLoadBufferSize** – Use this option to specify the maximum number of records to store in the buffer that the parallel load threads use to load data to the Kinesis target. The default value is 50. The maximum value is 1,000. Use this setting with **ParallelLoadThreads**. **ParallelLoadBufferSize** is valid only when there is more than one thread.
- **ParallelLoadQueuesPerThread** – Use this option to specify the number of queues each concurrent thread accesses to take data records out of queues and generate a batch load for the target. The default is 1. However, for Kinesis targets of various payload sizes, the valid range is 5–512 queues per thread.

### Multithreaded CDC load task settings

You can improve the performance of change data capture (CDC) for real-time data streaming target endpoints like Kinesis using task settings to modify the behavior of the PutRecords API call. To do this, you can specify the number of concurrent threads, queues per thread, and the number of records to store in a buffer using **ParallelApply\*** task settings. For example, suppose you want to perform a CDC load and apply 128 threads in parallel. You also want to access 64 queues per thread, with 50 records stored per buffer.

To promote CDC performance, AWS DMS supports these task settings:

- **ParallelApplyThreads** – Specifies the number of concurrent threads that AWS DMS uses during a CDC load to push data records to a Kinesis target endpoint. The default value is zero (0) and the maximum value is 32.
- **ParallelApplyBufferSize** – Specifies the maximum number of records to store in each buffer queue for concurrent threads to push to a Kinesis target endpoint during a CDC load. The default value is 100 and the maximum value is 1,000. Use this option when **ParallelApplyThreads** specifies more than one thread.
- **ParallelApplyQueuesPerThread** – Specifies the number of queues that each thread accesses to take data records out of queues and generate a batch load for a Kinesis endpoint during CDC.

When using **ParallelApply\*** task settings, the `partition-key-type` default is the `primary-key` of the table, not `schema-name.table-name`.

## Using a before image to view original values of CDC rows for a Kinesis data stream as a target

When writing CDC updates to a data-streaming target like Kinesis, you can view a source database row's original values before change by an update. To make this possible, AWS DMS populates a *before image* of update events based on data supplied by the source database engine.

Different source database engines provide different amounts of information for a before image:

- Oracle provides updates to columns only if they change.
- PostgreSQL provides only data for columns that are part of the primary key (changed or not). To provide data for all columns (changed or not), you need to set `REPLICA_IDENTITY` to `FULL` instead of `DEFAULT`. Note that you should choose the `REPLICA_IDENTITY` setting carefully for each table. If you set `REPLICA_IDENTITY` to `FULL`, all of the column values are written to write-ahead logging (WAL) continuously. This may cause performance or resource issues with tables that are updated frequently.
- MySQL generally provides data for all columns except for `BLOB` and `CLOB` data types (changed or not).

To enable before imaging to add original values from the source database to the AWS DMS output, use either the `BeforeImageSettings` task setting or the `add-before-image-columns` parameter. This parameter applies a column transformation rule.

BeforeImageSettings adds a new JSON attribute to every update operation with values collected from the source database system, as shown following.

```
"BeforeImageSettings": {  
    "EnableBeforeImage": boolean,  
    "FieldName": string,  
    "ColumnFilter": pk-only (default) / non-lob / all (but only one)  
}
```

**Note**

Only apply BeforeImageSettings to AWS DMS tasks that contain a CDC component, such as full load plus CDC tasks (which migrate existing data and replicate ongoing changes), or to CDC only tasks (which replicate data changes only). Don't apply BeforeImageSettings to tasks that are full load only.

For BeforeImageSettings options, the following applies:

- Set the EnableBeforeImage option to true to enable before imaging. The default is false.
- Use the FieldName option to assign a name to the new JSON attribute. When EnableBeforeImage is true, FieldName is required and can't be empty.
- The ColumnFilter option specifies a column to add by using before imaging. To add only columns that are part of the table's primary keys, use the default value, pk-only. To add any column that has a before image value, use all. Note that the before image does not contain columns with LOB data types, such as CLOB or BLOB.

```
"BeforeImageSettings": {  
    "EnableBeforeImage": true,  
    "FieldName": "before-image",  
    "ColumnFilter": "pk-only"  
}
```

**Note**

Amazon S3 targets don't support BeforeImageSettings. For S3 targets, use only the add-before-image-columns transformation rule to perform before imaging during CDC.

## Using a before image transformation rule

As an alternative to task settings, you can use the add-before-image-columns parameter, which applies a column transformation rule. With this parameter, you can enable before imaging during CDC on data streaming targets like Kinesis.

By using add-before-image-columns in a transformation rule, you can apply more fine-grained control of the before image results. Transformation rules enable you to use an object locator that gives you control over tables selected for the rule. Also, you can chain transformation rules together, which allows different rules to be applied to different tables. You can then manipulate the columns produced by using other rules.

**Note**

Don't use the add-before-image-columns parameter together with the BeforeImageSettings task setting within the same task. Instead, use either the parameter or the setting, but not both, for a single task.

A transformation rule type with the add-before-image-columns parameter for a column must provide a before-image-def section. The following shows an example.

```
{
  "rule-type": "transformation",
  ...
  "rule-target": "column",
  "rule-action": "add-before-image-columns",
  "before-image-def":{
    "column-filter": one-of (pk-only / non-lob / all),
    "column-prefix": string,
    "column-suffix": string,
  }
}
```

The value of `column-prefix` is prepended to a column name, and the default value of `column-prefix` is `BI_`. The value of `column-suffix` is appended to the column name, and the default is empty. Don't set both `column-prefix` and `column-suffix` to empty strings.

Choose one value for `column-filter`. To add only columns that are part of table primary keys, choose `pk-only`. Choose `non-lob` to only add columns that are not of LOB type. Or choose `all` to add any column that has a before-image value.

### Example for a before image transformation rule

The transformation rule in the following example adds a new column called `BI_emp_no` in the target. So a statement like `UPDATE employees SET emp_no = 3 WHERE emp_no = 1;` populates the `BI_emp_no` field with 1. When you write CDC updates to Amazon S3 targets, the `BI_emp_no` column makes it possible to tell which original row was updated.

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "%",
        "table-name": "%"
      },
      "rule-action": "include"
    },
    {
      "rule-type": "transformation",
      "rule-id": "2",
      "rule-name": "2",
      "rule-target": "column",
      "object-locator": {
        "schema-name": "%",
        "table-name": "employees"
      },
      "rule-action": "add-before-image-columns",
      "before-image-def": {
        "column-prefix": "BI_",
        "column-suffix": "",
        "column-filter": "pk-only"
      }
    }
  ]
}
```

For information on using the `add-before-image-columns` rule action, see [Transformation rules and actions \(p. 500\)](#).

## Prerequisites for using a Kinesis data stream as a target for AWS Database Migration Service

Before you set up a Kinesis data stream as a target for AWS DMS, make sure that you create an IAM role. This role must allow AWS DMS to assume and grant access to the Kinesis data streams that are being migrated into. The minimum set of access permissions is shown in the following IAM policy.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "1",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "dms.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

The role that you use for the migration to a Kinesis data stream must have the following permissions.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "kinesis:DescribeStream",  
                "kinesis:PutRecord",  
                "kinesis:PutRecords"  
            ],  
            "Resource": "arn:aws:kinesis:region:accountID:stream/streamName"  
        }  
    ]  
}
```

## Limitations when using Kinesis Data Streams as a target for AWS Database Migration Service

The following limitations apply when using Kinesis Data Streams as a target:

- AWS DMS publishes each update to a single record in the source database as one data record in a given Kinesis data stream regardless of transactions. However, you can include transaction details for each data record by using relevant parameters of the `KinesisSettings` API.
- Full LOB mode is not supported.
- The maximum supported LOB size is 1 MB.
- Kinesis Data Streams don't support deduplication. Applications that consume data from a stream need to handle duplicate records. For more information, see [Handling duplicate records](#) in the *Amazon Kinesis Data Streams Developer Guide*.
- AWS DMS supports the following two forms for partition keys:

- SchemaName . TableName: A combination of the schema and table name.
- \${AttributeName}: The value of one of the fields in the JSON, or the primary key of the table in the source database.
- For information about encrypting your data at rest within Kinesis Data Streams, see [Data protection in Kinesis Data Streams](#) in the *AWS Key Management Service Developer Guide*.
- BatchApply is not supported for a Kinesis endpoint. Using Batch Apply (for example, the BatchApplyEnabled target metadata task setting) for a Kinesis target might result in loss of data.
- Kinesis targets are only supported for a Kinesis data stream in the same AWS account.
- When migrating from a MySQL source, the BeforeImage data doesn't include CLOB and BLOB data types. For more information, see [Using a before image to view original values of CDC rows for a Kinesis data stream as a target \(p. 378\)](#).
- AWS DMS migrates values of BigInt data type with more than 16 digits in the scientific notation form.

## Using object mapping to migrate data to a Kinesis data stream

AWS DMS uses table-mapping rules to map data from the source to the target Kinesis data stream. To map data to a target stream, you use a type of table-mapping rule called object mapping. You use object mapping to define how data records in the source map to the data records published to the Kinesis data stream.

Kinesis data streams don't have a preset structure other than having a partition key. In an object mapping rule, the possible values of a `partition-key-type` for data records are `schema-table`, `transaction-id`, `primary-key`, `constant`, and `attribute-name`.

To create an object-mapping rule, you specify `rule-type` as `object-mapping`. This rule specifies what type of object mapping you want to use.

The structure for the rule is as follows.

```
{  
    "rules": [  
        {  
            "rule-type": "object-mapping",  
            "rule-id": "id",  
            "rule-name": "name",  
            "rule-action": "valid object-mapping rule action",  
            "object-locator": {  
                "schema-name": "case-sensitive schema name",  
                "table-name": ""  
            }  
        }  
    ]  
}
```

AWS DMS currently supports `map-record-to-record` and `map-record-to-document` as the only valid values for the `rule-action` parameter. The `map-record-to-record` and `map-record-to-document` values specify what AWS DMS does by default to records that aren't excluded as part of the `exclude-columns` attribute list. These values don't affect the attribute mappings in any way.

Use `map-record-to-record` when migrating from a relational database to a Kinesis data stream. This rule type uses the `taskResourceId.schemaName.tableName` value from the relational database as the partition key in the Kinesis data stream and creates an attribute for each column in the source database. When using `map-record-to-record`, for any column in the source table not listed in the `exclude-columns` attribute list, AWS DMS creates a corresponding attribute in the target stream.

This corresponding attribute is created regardless of whether that source column is used in an attribute mapping.

Use `map-record-to-document` to put source columns into a single, flat document in the appropriate target stream using the attribute name `_doc`. AWS DMS places the data into a single, flat map on the source called `_doc`. This placement applies to any column in the source table not listed in the `exclude-columns` attribute list.

One way to understand `map-record-to-record` is to see it in action. For this example, assume that you are starting with a relational database table row with the following structure and data.

FirstName	LastName	StoreId	HomeAddress	HomePhone	WorkAddress	WorkPhone	DateOfBirth
Randy	Marsh	5	221B Baker Street	123456789031	Spooner Street, Quahog	987654321002/29/1988	

To migrate this information from a schema named `Test` to a Kinesis data stream, you create rules to map the data to the target stream. The following rule illustrates the mapping.

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "rule-action": "include",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            }
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "2",
            "rule-name": "DefaultMapToKinesis",
            "rule-action": "map-record-to-record",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "Customers"
            }
        }
    ]
}
```

The following illustrates the resulting record format in the Kinesis data stream:

- StreamName: XXX
- PartitionKey: Test.Customers //schmaName.tableName
- Data: //The following JSON message

```
{
    "FirstName": "Randy",
    "LastName": "Marsh",
    "StoreId": "5",
    "HomeAddress": "221B Baker Street",
```

```

        "HomePhone": "1234567890",
        "WorkAddress": "31 Spooner Street, Quahog",
        "WorkPhone": "9876543210",
        "DateOfBirth": "02/29/1988"
    }
}

```

However, suppose that you use the same rules but change the rule-action parameter to map-record-to-document and exclude certain columns. The following rule illustrates the mapping.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "rule-action": "include",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            }
        },
        {
            "rule-type": "object-mapping",
            "rule-id": "2",
            "rule-name": "DefaultMapToKinesis",
            "rule-action": "map-record-to-document",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "Customers"
            },
            "mapping-parameters": {
                "exclude-columns": [
                    "homeaddress",
                    "homephone",
                    "workaddress",
                    "workphone"
                ]
            }
        }
    ]
}

```

In this case, the columns not listed in the exclude-columns parameter, FirstName, LastName, StoreId and DateOfBirth, are mapped to \_doc. The following illustrates the resulting record format.

```

{
    "data": {
        "_doc": {
            "FirstName": "Randy",
            "LastName": "Marsh",
            "StoreId": "5",
            "DateOfBirth": "02/29/1988"
        }
    }
}

```

## Restructuring data with attribute mapping

You can restructure the data while you are migrating it to a Kinesis data stream using an attribute map. For example, you might want to combine several fields in the source into a single field in the target. The following attribute map illustrates how to restructure the data.

```
{  
    "rules": [  
        {  
            "rule-type": "selection",  
            "rule-id": "1",  
            "rule-name": "1",  
            "rule-action": "include",  
            "object-locator": {  
                "schema-name": "Test",  
                "table-name": "%"  
            }  
        },  
        {  
            "rule-type": "object-mapping",  
            "rule-id": "2",  
            "rule-name": "TransformToKinesis",  
            "rule-action": "map-record-to-record",  
            "target-table-name": "CustomerData",  
            "object-locator": {  
                "schema-name": "Test",  
                "table-name": "Customers"  
            },  
            "mapping-parameters": {  
                "partition-key-type": "attribute-name",  
                "partition-key-name": "CustomerName",  
                "exclude-columns": [  
                    "firstname",  
                    "lastname",  
                    "homeaddress",  
                    "homephone",  
                    "workaddress",  
                    "workphone"  
                ],  
                "attribute-mappings": [  
                    {  
                        "target-attribute-name": "CustomerName",  
                        "attribute-type": "scalar",  
                        "attribute-sub-type": "string",  
                        "value": "${lastname}, ${firstname}"  
                    },  
                    {  
                        "target-attribute-name": "ContactDetails",  
                        "attribute-type": "document",  
                        "attribute-sub-type": "json",  
                        "value": {  
                            "Home": {  
                                "Address": "${homeaddress}",  
                                "Phone": "${homephone}"  
                            },  
                            "Work": {  
                                "Address": "${workaddress}",  
                                "Phone": "${workphone}"  
                            }  
                        }  
                    }  
                ]  
            }  
        }  
    ]  
}
```

```
    ]  
}
```

To set a constant value for `partition-key`, specify a `partition-key` value. For example, you might do this to force all the data to be stored in a single shard. The following mapping illustrates this approach.

```
{  
    "rules": [  
        {  
            "rule-type": "selection",  
            "rule-id": "1",  
            "rule-name": "1",  
            "object-locator": {  
                "schema-name": "Test",  
                "table-name": "%"  
            },  
            "rule-action": "include"  
        },  
        {  
            "rule-type": "object-mapping",  
            "rule-id": "1",  
            "rule-name": "TransformToKinesis",  
            "rule-action": "map-record-to-document",  
            "object-locator": {  
                "schema-name": "Test",  
                "table-name": "Customer"  
            },  
            "mapping-parameters": {  
                "partition-key": {  
                    "value": "ConstantPartitionKey"  
                },  
                "exclude-columns": [  
                    "FirstName",  
                    "LastName",  
                    "HomeAddress",  
                    "HomePhone",  
                    "WorkAddress",  
                    "WorkPhone"  
                ],  
                "attribute-mappings": [  
                    {  
                        "attribute-name": "CustomerName",  
                        "value": "${FirstName},${LastName}"  
                    },  
                    {  
                        "attribute-name": "ContactDetails",  
                        "value": {  
                            "Home": {  
                                "Address": "${HomeAddress}",  
                                "Phone": "${HomePhone}"  
                            },  
                            "Work": {  
                                "Address": "${WorkAddress}",  
                                "Phone": "${WorkPhone}"  
                            }  
                        }  
                    },  
                    {  
                        "attribute-name": "DateOfBirth",  
                        "value": "${DateOfBirth}"  
                    }  
                ]  
            }  
        }  
    ]  
}
```

```
        ]  
    }
```

#### Note

The partition-key value for a control record that is for a specific table is `TaskId.SchemaName.TableName`. The partition-key value for a control record that is for a specific task is that record's `TaskId`. Specifying a partition-key value in the object mapping has no impact on the partition-key for a control record.

## Message format for Kinesis Data Streams

The JSON output is simply a list of key-value pairs. A `JSON_UNFORMATTED` message format is a single line JSON string with new line delimiter.

AWS DMS provides the following reserved fields to make it easier to consume the data from the Kinesis Data Streams:

#### RecordType

The record type can be either data or control. *Data records* represent the actual rows in the source. *Control records* are for important events in the stream, for example a restart of the task.

#### Operation

For data records, the operation can be `load`, `insert`, `update`, or `delete`.

For control records, the operation can be `create-table`, `rename-table`, `drop-table`, `change-columns`, `add-column`, `drop-column`, `rename-column`, or `column-type-change`.

#### SchemaName

The source schema for the record. This field can be empty for a control record.

#### TableName

The source table for the record. This field can be empty for a control record.

#### Timestamp

The timestamp for when the JSON message was constructed. The field is formatted with the ISO 8601 format.

## Using Apache Kafka as a target for AWS Database Migration Service

You can use AWS DMS to migrate data to an Apache Kafka cluster. Apache Kafka is a distributed streaming platform. You can use Apache Kafka for ingesting and processing streaming data in real-time.

AWS also offers Amazon Managed Streaming for Apache Kafka (Amazon MSK) to use as an AWS DMS target. Amazon MSK is a fully managed Apache Kafka streaming service that simplifies the implementation and management of Apache Kafka instances. It works with open-source Apache Kafka versions, and you access Amazon MSK instances as AWS DMS targets exactly like any Apache Kafka instance. For more information, see [What is Amazon MSK?](#) in the *Amazon Managed Streaming for Apache Kafka Developer Guide*.

A Kafka cluster stores streams of records in categories called topics that are divided into partitions. *Partitions* are uniquely identified sequences of data records (messages) in a topic. Partitions can be distributed across multiple brokers in a cluster to enable parallel processing of a topic's records. For more

information on topics and partitions and their distribution in Apache Kafka, see [Topics and logs](#) and [Distribution](#).

Your Kafka cluster can be either an Amazon MSK instance, a cluster running on an Amazon EC2 instance, or an on-premises cluster. An Amazon MSK instance or a cluster on an Amazon EC2 instance can be in the same VPC or a different one. If your cluster is on-premises, you can use your own on-premises name server for your replication instance to resolve the cluster's host name. For information about setting up a name server for your replication instance, see [Using your own on-premises name server \(p. 90\)](#). For more information about setting up a network, see [Setting up a network for a replication instance \(p. 112\)](#).

When using an Amazon MSK cluster, make sure that it has the same security group as your replication instance. For information about changing the security group for an Amazon MSK cluster, see [Changing an Amazon MSK cluster's security group](#).

AWS Database Migration Service publishes records to a Kafka topic using JSON. During conversion, AWS DMS serializes each record from the source database into an attribute-value pair in JSON format.

To migrate your data from any supported data source to a target Kafka cluster, you use object mapping. With object mapping, you determine how to structure the data records in the target topic. You also define a partition key for each table, which Apache Kafka uses to group the data into its partitions.

Currently, AWS DMS supports a single topic per task. For a single task with multiple tables, all messages go to a single topic. Each message includes a metadata section that identifies the target schema and table. AWS DMS versions 3.4.6 and later support multtopic replication using object mapping. For more information, see [Multitopic replication using object mapping \(p. 401\)](#).

## Apache Kafka endpoint settings

You can specify connection details through endpoint settings in the AWS DMS console, or the `--kafka-settings` option in the CLI. The requirements for each setting follow:

- **Broker** – Specify the locations of one or more brokers in your Kafka cluster in the form of a comma-separated list of each *broker-hostname:port*. An example is `"ec2-12-345-678-901.compute-1.amazonaws.com:2345, ec2-10-987-654-321.compute-1.amazonaws.com:2345"`. This setting can specify the locations of any or all brokers in the cluster. The cluster brokers all communicate to handle the partitioning of data records migrated to the topic.
- **Topic** – (Optional) Specify the topic name with a maximum length of 255 letters and symbols. You can use period (.), underscore (\_), and minus (-). Topic names with a period (.) or underscore (\_) can collide in internal data structures. Use either one, but not both of these symbols in the topic name. If you don't specify a topic name, AWS DMS uses "kafka-default-topic" as the migration topic.

### Note

To have AWS DMS create either a migration topic you specify or the default topic, set `auto.create.topics.enable = true` as part of your Kafka cluster configuration. For more information, see [Limitations when using Apache Kafka as a target for AWS Database Migration Service \(p. 397\)](#)

- **MessageFormat** – The output format for the records created on the endpoint. The message format is JSON (default) or JSON\_UNFORMATTED (a single line with no tab).
- **MessageMaxBytes** – The maximum size in bytes for records created on the endpoint. The default is 1,000,000.

### Note

You can only use the AWS CLI/SDK to change MessageMaxBytes to a non-default value. For example, to modify your existing Kafka endpoint and change MessageMaxBytes, use the following command.

```
aws dms modify-endpoint --endpoint-arn your-endpoint
--kafka-settings Broker="broker1-server:broker1-port,broker2-server:broker2-port,...",
```

Topic=*topic-name*,MessageMaxBytes=*integer-of-max-message-size-in-bytes*

- **IncludeTransactionDetails** – Provides detailed transaction information from the source database. This information includes a commit timestamp, a log position, and values for `transaction_id`, `previous_transaction_id`, and `transaction_record_id` (the record offset within a transaction). The default is `false`.
- **IncludePartitionValue** – Shows the partition value within the Kafka message output, unless the partition type is `schema-table-type`. The default is `false`.
- **PartitionIncludeSchemaTable** – Prefixes schema and table names to partition values, when the partition type is `primary-key-type`. Doing this increases data distribution among Kafka partitions. For example, suppose that a SysBench schema has thousands of tables and each table has only limited range for a primary key. In this case, the same primary key is sent from thousands of tables to the same partition, which causes throttling. The default is `false`.
- **IncludeTableAlterOperations** – Includes any data definition language (DDL) operations that change the table in the control data, such as `rename-table`, `drop-table`, `add-column`, `drop-column`, and `rename-column`. The default is `false`.
- **IncludeControlDetails** – Shows detailed control information for table definition, column definition, and table and column changes in the Kafka message output. The default is `false`.
- **IncludeNullAndEmpty** – Include NULL and empty columns in the target. The default is `false`.
- **SecurityProtocol** – Sets a secure connection to a Kafka target endpoint using Transport Layer Security (TLS). Options include `ssl-authentication`, `ssl-encryption`, and `sasl-ssl`. Using `sasl-ssl` requires `SaslUsername` and `SaslPassword`.

You can use settings to help increase the speed of your transfer. To do so, AWS DMS supports a multithreaded full load to an Apache Kafka target cluster. AWS DMS supports this multithreading with task settings that include the following:

- **MaxFullLoadSubTasks** – Use this option to indicate the maximum number of source tables to load in parallel. AWS DMS loads each table into its corresponding Kafka target table using a dedicated subtask. The default is 8; the maximum value is 49.
- **ParallelLoadThreads** – Use this option to specify the number of threads that AWS DMS uses to load each table into its Kafka target table. The maximum value for an Apache Kafka target is 32. You can ask to have this maximum limit increased.
- **ParallelLoadBufferSize** – Use this option to specify the maximum number of records to store in the buffer that the parallel load threads use to load data to the Kafka target. The default value is 50. The maximum value is 1,000. Use this setting with `ParallelLoadThreads`. `ParallelLoadBufferSize` is valid only when there is more than one thread.
- **ParallelLoadQueuesPerThread** – Use this option to specify the number of queues each concurrent thread accesses to take data records out of queues and generate a batch load for the target. The default is 1. The maximum is 512.

You can improve the performance of change data capture (CDC) for Kafka endpoints by tuning task settings for parallel threads and bulk operations. To do this, you can specify the number of concurrent threads, queues per thread, and the number of records to store in a buffer using `ParallelApply*` task settings. For example, suppose you want to perform a CDC load and apply 128 threads in parallel. You also want to access 64 queues per thread, with 50 records stored per buffer.

To promote CDC performance, AWS DMS supports these task settings:

- **ParallelApplyThreads** – Specifies the number of concurrent threads that AWS DMS uses during a CDC load to push data records to a Kafka target endpoint. The default value is zero (0) and the maximum value is 32.
- **ParallelApplyBufferSize** – Specifies the maximum number of records to store in each buffer queue for concurrent threads to push to a Kafka target endpoint during a CDC load. The default value

is 100 and the maximum value is 1,000. Use this option when `ParallelApplyThreads` specifies more than one thread.

- `ParallelApplyQueuesPerThread` – Specifies the number of queues that each thread accesses to take data records out of queues and generate a batch load for a Kafka endpoint during CDC. The default is 1. The maximum is 512.

When using `ParallelApply*` task settings, the `partition-key-type` default is the `primary-key` of the table, not `schema-name.table-name`.

## Connecting to Kafka using Transport Layer Security (TLS)

A Kafka cluster accepts secure connections using Transport Layer Security (TLS). With DMS, you can use any one of the following three security protocol options to secure a Kafka endpoint connection.

### SSL encryption (`server-encryption`)

Clients validate server identity through the server's certificate. Then an encrypted connection is made between server and client.

### SSL authentication (`mutual-authentication`)

Server and client validate the identity with each other through their own certificates. Then an encrypted connection is made between server and client.

### SASL-SSL (`mutual-authentication`)

The Simple Authentication and Security Layer (SASL) method replaces the client's certificate with a user name and password to validate a client identity. Specifically, you provide a user name and password that the server has registered so that the server can validate the identity of a client. Then an encrypted connection is made between server and client.

#### Important

Apache Kafka and Amazon MSK accept resolved certificates. This is a known limitation of Kafka and Amazon MSK to be addressed. For more information, see [Apache Kafka issues, KAFKA-3700](#).

If you're using Amazon MSK, consider using access control lists (ACLs) as a workaround to this known limitation. For more information about using ACLs, see [Apache Kafka ACLs](#) section of [Amazon Managed Streaming for Apache Kafka Developer Guide](#).

If you're using a self-managed Kafka cluster, see [Comment dated 21/Oct/18](#) for information about configuring your cluster.

## Using SSL encryption with Amazon MSK or a self-managed Kafka cluster

You can use SSL encryption to secure an endpoint connection to Amazon MSK or a self-managed Kafka cluster. When you use the SSL encryption authentication method, clients validate a server's identity through the server's certificate. Then an encrypted connection is made between server and client.

### To use SSL encryption to connect to Amazon MSK

- Set the security protocol endpoint setting (`SecurityProtocol`) using the `ssl-encryption` option when you create your target Kafka endpoint.

The JSON example following sets the security protocol as SSL encryption.

```
"KafkaSettings": {  
    "SecurityProtocol": "ssl-encryption",
```

}

### To use SSL encryption for a self-managed Kafka cluster

1. If you're using a private Certification Authority (CA) in your on-premises Kafka cluster, upload your private CA cert and get an Amazon Resource Name (ARN).
2. Set the security protocol endpoint setting (`SecurityProtocol`) using the `ssl-encryption` option when you create your target Kafka endpoint. The JSON example following sets the security protocol as `ssl-encryption`.

```
"KafkaSettings": {  
    "SecurityProtocol": "ssl-encryption",  
}
```

3. If you're using a private CA, set `SslCaCertificateArn` in the ARN you got in the first step above.

## Using SSL authentication

You can use SSL authentication to secure an endpoint connection to Amazon MSK or a self-managed Kafka cluster.

To enable client authentication and encryption using SSL authentication to connect to Amazon MSK, do the following:

- Prepare a private key and public certificate for Kafka.
- Upload certificates to the DMS certificate manager.
- Create a Kafka target endpoint with corresponding certificate ARNs specified in Kafka endpoint settings.

### To prepare a private key and public certificate for Amazon MSK

1. Create an EC2 instance and set up a client to use authentication as described in steps 1 through 9 in the [Client Authentication](#) section of *Amazon Managed Streaming for Apache Kafka Developer Guide*.

After you complete those steps, you have a Certificate-ARN (the public certificate ARN saved in ACM), and a private key contained within a `kafka.client.keystore.jks` file.

2. Get the public certificate and copy the certificate to the `signed-certificate-from-acm.pem` file, using the command following:

```
aws acm-pca get-certificate --certificate-authority-arn Private_CA_ARN --certificate-arn Certificate_ARN
```

That command returns information similar to the following example:

```
{"Certificate": "123", "CertificateChain": "456"}
```

You then copy your equivalent of "123" to the `signed-certificate-from-acm.pem` file.

3. Get the private key by importing the `msk-rsa` key from `kafka.client.keystore.jks` to `keystore.p12`, as shown in the following example.

```
keytool -importkeystore \
```

```
-srckeystore kafka.client.keystore.jks \
-destkeystore keystore.p12 \
-deststoretype PKCS12 \
-srcalias msk-rsa-client \
-deststorepass test1234 \
-destkeypass test1234
```

4. Use the following command to export keystore.p12 into . pem format.

```
Openssl pkcs12 -in keystore.p12 -out encrypted-private-client-key.pem -nocerts
```

The **Enter PEM pass phrase** message appears and identifies the key that is applied to encrypt the certificate.

5. Remove bag attributes and key attributes from the . pem file to make sure that the first line starts with the following string.

```
---BEGIN ENCRYPTED PRIVATE KEY---
```

## To upload a public certificate and private key to the DMS certificate manager and test the connection to Amazon MSK

1. Upload to DMS certificate manager using the following command.

```
aws dms import-certificate --certificate-identifier signed-cert --certificate-pem
file://path to signed cert
aws dms import-certificate --certificate-identifier private-key --certificate-pem
file://path to private key
```

2. Create an Amazon MSK target endpoint and test connection to make sure that TLS authentication works.

```
aws dms create-endpoint --endpoint-identifier $endpoint-identifier --engine-name kafka
--endpoint-type target --kafka-settings
'{"Broker": "b-0.kafka260.aaaaa1.a99.kafka.us-east-1.amazonaws.com:0000",
 "SecurityProtocol": "ssl-authentication",
 "SslClientCertificateArn": "arn:aws:dms:us-east-1:012346789012:cert:",
 "SslClientKeyArn": "arn:aws:dms:us-
east-1:0123456789012:cert:", "SslClientKeyPassword": "test1234"}'
aws dms test-connection -replication-instance-arn=$rep_inst_arn -endpoint-arn=
$kafka_tar_arn_msk
```

### Important

You can use SSL authentication to secure a connection to a self-managed Kafka cluster. In some cases, you might use a private Certification Authority (CA) in your on-premises Kafka cluster. If so, upload your CA chain, public certificate, and private key to the DMS certificate manager. Then, use the corresponding Amazon Resource Name (ARN) in your endpoint settings when you create your on-premises Kafka target endpoint.

## To prepare a private key and signed certificate for a self-managed Kafka cluster

1. Generate a key pair as shown in the following example.

```
keytool -genkey -keystore kafka.server.keystore.jks -validity 300 -storepass your-
keystore-password
```

```
-keypass your-key-passphrase -dname "CN=your-cn-name"  
-alias alias-of-key-pair -storetype pkcs12 -keyalg RSA
```

2. Generate a Certificate Sign Request (CSR).

```
keytool -keystore kafka.server.keystore.jks -certreq -file server-cert-sign-request-rsa  
-alias on-premise-rsa -storepass your-key-store-password  
-keypass your-key-password
```

3. Use the CA in your cluster truststore to sign the CSR. If you don't have a CA, you can create your own private CA.

```
openssl req -new -x509 -keyout ca-key -out ca-cert -days validate-days
```

4. Import ca-cert into the server truststore and keystore. If you don't have a truststore, use the following command to create the truststore and import ca-cert into it.

```
keytool -keystore kafka.server.truststore.jks -alias CARoot -import -file ca-cert  
keytool -keystore kafka.server.keystore.jks -alias CARoot -import -file ca-cert
```

5. Sign the certificate.

```
openssl x509 -req -CA ca-cert -CAkey ca-key -in server-cert-sign-request-rsa -out  
signed-server-certificate.pem  
-days validate-days -CAcreateserial -passin pass:ca-password
```

6. Import the signed certificate to the keystore.

```
keytool -keystore kafka.server.keystore.jks -import -file signed-certificate.pem -alias  
on-premise-rsa -storepass your-keystore-password  
-keypass your-key-password
```

7. Use the following command to import the on-premise-rsa key from kafka.server.keystore.jks to keystore.p12.

```
keytool -importkeystore \  
-srckeystore kafka.server.keystore.jks \  
-destkeystore keystore.p12 \  
-deststoretype PKCS12 \  
-srcalias on-premise-rsa \  
-deststorepass your-truststore-password \  
-destkeypass your-key-password
```

8. Use the following command to export keystore.p12 into .pem format.

```
openssl pkcs12 -in keystore.p12 -out encrypted-private-server-key.pem -nocerts
```

9. Upload encrypted-private-server-key.pem, signed-certificate.pem, and ca-cert to the DMS certificate manager.

10. Create an endpoint by using the returned ARNs.

```
aws dms create-endpoint --endpoint-identifier $endpoint-identifier --engine-name kafka
--endpoint-type target --kafka-settings
'{"Broker": "b-0.kafka260.aaaaa1.a99.kafka.us-east-1.amazonaws.com:9092",
 "SecurityProtocol": "ssl-authentication",
 "SslClientCertificateArn": "your-client-cert-arn", "SslClientKeyArn": "your-client-key-
arn", "SslClientKeyPassword": "your-client-key-password",
 "SslCaCertificateArn": "your-ca-certificate-arn"}'

aws dms test-connection -replication-instance-arn=$rep_inst_arn -endpoint-arn=
$kafka_tar_arn_msk
```

## Using SASL-SSL authentication to connect to Amazon MSK

The Simple Authentication and Security Layer (SASL) method uses a user name and password to validate a client identity, and makes an encrypted connection between server and client.

To use SASL, you first create a secure user name and password when you set up your Amazon MSK cluster. For a description how to set up a secure user name and password for an Amazon MSK cluster, see [Setting up SASL/SCRAM authentication for an Amazon MSK cluster](#) in the *Amazon Managed Streaming for Apache Kafka Developer Guide*.

Then, when you create your Kafka target endpoint, set the security protocol endpoint setting (`SecurityProtocol`) using the `sasl-ssl` option. You also set `SaslUsername` and `SaslPassword` options. Make sure these are consistent with the secure user name and password that you created when you first set up your Amazon MSK cluster, as shown in the following JSON example.

```
"KafkaSettings": {
    "SecurityProtocol": "sasl-ssl",
    "SaslUsername": "Amazon MSK cluster secure user name",
    "SaslPassword": "Amazon MSK cluster secure password"
}
```

### Note

Currently, AWS DMS supports only public CA backed SASL/SSL. DMS doesn't support SASL/SSL for use with self-managed Kafka that is backed by private CA.

For SASL/SSL authentication, AWS DMS supports only the SCRAM-SHA-512 mechanism.

## Using a before image to view original values of CDC rows for Apache Kafka as a target

When writing CDC updates to a data-streaming target like Kafka you can view a source database row's original values before change by an update. To make this possible, AWS DMS populates a *before image* of update events based on data supplied by the source database engine.

Different source database engines provide different amounts of information for a before image:

- Oracle provides updates to columns only if they change.
- PostgreSQL provides only data for columns that are part of the primary key (changed or not). If logical replication is in use and REPLICAS IDENTITY FULL is set for the source table, you can get entire before and after information on the row written to the WALs and available here.
- MySQL generally provides data for all columns (changed or not).

To enable before imaging to add original values from the source database to the AWS DMS output, use either the `BeforeImageSettings` task setting or the `add-before-image-columns` parameter. This parameter applies a column transformation rule.

`BeforeImageSettings` adds a new JSON attribute to every update operation with values collected from the source database system, as shown following.

```
"BeforeImageSettings": {  
    "EnableBeforeImage": boolean,  
    "FieldName": string,  
    "ColumnFilter": pk-only (default) / non-lob / all (but only one)  
}
```

**Note**

Apply `BeforeImageSettings` to full load plus CDC tasks (which migrate existing data and replicate ongoing changes), or to CDC only tasks (which replicate data changes only). Don't apply `BeforeImageSettings` to tasks that are full load only.

For `BeforeImageSettings` options, the following applies:

- Set the `EnableBeforeImage` option to `true` to enable before imaging. The default is `false`.
- Use the `FieldName` option to assign a name to the new JSON attribute. When `EnableBeforeImage` is `true`, `FieldName` is required and can't be empty.
- The `ColumnFilter` option specifies a column to add by using before imaging. To add only columns that are part of the table's primary keys, use the default value, `pk-only`. To add only columns that are not of LOB type, use `non-lob`. To add any column that has a before image value, use `all`.

```
"BeforeImageSettings": {  
    "EnableBeforeImage": true,  
    "FieldName": "before-image",  
    "ColumnFilter": "pk-only"  
}
```

## Using a before image transformation rule

As an alternative to task settings, you can use the `add-before-image-columns` parameter, which applies a column transformation rule. With this parameter, you can enable before imaging during CDC on data streaming targets like Kafka.

By using `add-before-image-columns` in a transformation rule, you can apply more fine-grained control of the before image results. Transformation rules enable you to use an object locator that gives you control over tables selected for the rule. Also, you can chain transformation rules together, which allows different rules to be applied to different tables. You can then manipulate the columns produced by using other rules.

**Note**

Don't use the `add-before-image-columns` parameter together with the `BeforeImageSettings` task setting within the same task. Instead, use either the parameter or the setting, but not both, for a single task.

A transformation rule type with the `add-before-image-columns` parameter for a column must provide a `before-image-def` section. The following shows an example.

```
{
```

```

    "rule-type": "transformation",
    ...
    "rule-target": "column",
    "rule-action": "add-before-image-columns",
    "before-image-def":{
        "column-filter": one-of (pk-only / non-lob / all),
        "column-prefix": string,
        "column-suffix": string,
    }
}

```

The value of `column-prefix` is prepended to a column name, and the default value of `column-prefix` is `BI_`. The value of `column-suffix` is appended to the column name, and the default is empty. Don't set both `column-prefix` and `column-suffix` to empty strings.

Choose one value for `column-filter`. To add only columns that are part of table primary keys, choose `pk-only`. Choose `non-lob` to only add columns that are not of LOB type. Or choose `all` to add any column that has a before-image value.

### Example for a before image transformation rule

The transformation rule in the following example adds a new column called `BI_emp_no` in the target. So a statement like `UPDATE employees SET emp_no = 3 WHERE emp_no = 1;` populates the `BI_emp_no` field with 1. When you write CDC updates to Amazon S3 targets, the `BI_emp_no` column makes it possible to tell which original row was updated.

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-target": "column",
            "object-locator": {
                "schema-name": "%",
                "table-name": "employees"
            },
            "rule-action": "add-before-image-columns",
            "before-image-def": {
                "column-prefix": "BI_",
                "column-suffix": "",
                "column-filter": "pk-only"
            }
        }
    ]
}
```

For information on using the `add-before-image-columns` rule action, see [Transformation rules and actions \(p. 500\)](#).

## Limitations when using Apache Kafka as a target for AWS Database Migration Service

The following limitations apply when using Apache Kafka as a target:

- AWS DMS Kafka target endpoints don't support IAM access control for Amazon Managed Streaming for Apache Kafka (Amazon MSK).
- Full LOB mode is not supported.
- Specify a Kafka configuration file for your cluster with properties that allow AWS DMS to automatically create new topics. Include the setting, `auto.create.topics.enable = true`. If you are using Amazon MSK, you can specify the default configuration when you create your Kafka cluster, then change the `auto.create.topics.enable` setting to `true`. For more information about the default configuration settings, see [The default Amazon MSK configuration in the Amazon Managed Streaming for Apache Kafka Developer Guide](#). If you need to modify an existing Kafka cluster created using Amazon MSK, run the AWS CLI command `aws kafka create-configuration` to update your Kafka configuration, as in the following example:

```
14:38:41 $ aws kafka create-configuration --name "kafka-configuration" --kafka-versions "2.2.1" --server-properties file://~/kafka_configuration
{
    "LatestRevision": {
        "Revision": 1,
        "CreationTime": "2019-09-06T14:39:37.708Z"
    },
    "CreationTime": "2019-09-06T14:39:37.708Z",
    "Name": "kafka-configuration",
    "Arn": "arn:aws:kafka:us-east-1:111122223333:configuration/kafka-configuration/7e008070-6a08-445f-9fe5-36ccf630ecfd-3"
}
```

Here, `//~/kafka_configuration` is the configuration file you have created with the required property settings.

If you are using your own Kafka instance installed on Amazon EC2, modify the Kafka cluster configuration with the `auto.create.topics.enable = true` setting to allow AWS DMS to automatically create new topics, using the options provided with your instance.

- AWS DMS publishes each update to a single record in the source database as one data record (message) in a given Kafka topic regardless of transactions.
- AWS DMS supports the following two forms for partition keys:
  - `SchemaName.TableName`: A combination of the schema and table name.
  - `#{AttributeName}`: The value of one of the fields in the JSON, or the primary key of the table in the source database.
- `BatchApply` is not supported for a Kafka endpoint. Using Batch Apply (for example, the `BatchApplyEnabled` target metadata task setting) for a Kafka target might result in loss of data.
- AWS DMS migrates values of `BigInt` data type with more than 16 digits in the scientific notation form.

## Using object mapping to migrate data to a Kafka topic

AWS DMS uses table-mapping rules to map data from the source to the target Kafka topic. To map data to a target topic, you use a type of table-mapping rule called object mapping. You use object mapping to define how data records in the source map to the data records published to a Kafka topic.

Kafka topics don't have a preset structure other than having a partition key.

### Note

You don't have to use object mapping. You can use regular table mapping for various transformations. However, the partition key type will follow these default behaviors:

- Primary Key is used as a partition key for Full Load.
- If no parallel-apply task settings are used, schema.table is used as a partition key for CDC.
- If parallel-apply task settings are used, Primary key is used as a partition key for CDC.

To create an object-mapping rule, specify rule-type as object-mapping. This rule specifies what type of object mapping you want to use.

The structure for the rule is as follows.

```
{
  "rules": [
    {
      "rule-type": "object-mapping",
      "rule-id": "id",
      "rule-name": "name",
      "rule-action": "valid object-mapping rule action",
      "object-locator": {
        "schema-name": "case-sensitive schema name",
        "table-name": ""
      }
    }
  ]
}
```

AWS DMS currently supports map-record-to-record and map-record-to-document as the only valid values for the rule-action parameter. The map-record-to-record and map-record-to-document values specify what AWS DMS does by default to records that aren't excluded as part of the exclude-columns attribute list. These values don't affect the attribute mappings in any way.

Use map-record-to-record when migrating from a relational database to a Kafka topic. This rule type uses the taskResourceId.schemaName.tableName value from the relational database as the partition key in the Kafka topic and creates an attribute for each column in the source database. When using map-record-to-record, for any column in the source table not listed in the exclude-columns attribute list, AWS DMS creates a corresponding attribute in the target topic. This corresponding attribute is created regardless of whether that source column is used in an attribute mapping.

One way to understand map-record-to-record is to see it in action. For this example, assume that you are starting with a relational database table row with the following structure and data.

FirstName	LastName	StoreId	HomeAddress	HomePhone	WorkAddress	WorkPhone	DateOfBirth
Randy	Marsh	5	221B Baker Street	123456789031	Spooner Street, Quahog	987654321002/29/1988	

To migrate this information from a schema named Test to a Kafka topic, you create rules to map the data to the target topic. The following rule illustrates the mapping.

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "id",
      "rule-name": "name",
      "rule-action": "valid object-mapping rule action",
      "object-locator": {
        "schema-name": "case-sensitive schema name",
        "table-name": ""
      }
    }
  ]
}
```

```

    "rule-id": "1",
    "rule-name": "1",
    "rule-action": "include",
    "object-locator": {
        "schema-name": "Test",
        "table-name": "%"
    }
},
{
    "rule-type": "object-mapping",
    "rule-id": "2",
    "rule-name": "DefaultMapToKafka",
    "rule-action": "map-record-to-record",
    "object-locator": {
        "schema-name": "Test",
        "table-name": "Customers"
    }
}
]
}

```

Given a Kafka topic and a partition key (in this case, `taskResourceId.schemaName.tableName`), the following illustrates the resulting record format using our sample data in the Kafka target topic:

```
{
    "FirstName": "Randy",
    "LastName": "Marsh",
    "StoreId": "5",
    "HomeAddress": "221B Baker Street",
    "HomePhone": "1234567890",
    "WorkAddress": "31 Spooner Street, Quahog",
    "WorkPhone": "9876543210",
    "DateOfBirth": "02/29/1988"
}
```

## Topics

- [Restructuring data with attribute mapping \(p. 399\)](#)
- [Multitopic replication using object mapping \(p. 401\)](#)
- [Message format for Apache Kafka \(p. 402\)](#)

## Restructuring data with attribute mapping

You can restructure the data while you are migrating it to a Kafka topic using an attribute map. For example, you might want to combine several fields in the source into a single field in the target. The following attribute map illustrates how to restructure the data.

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "rule-action": "include",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            }
        },
        {

```

To set a constant value for partition-key, specify a partition-key value. For example, you might do this to force all the data to be stored in a single partition. The following mapping illustrates this approach.

```
{  
  "rules": [  
    {  
      "rule-type": "selection",  
      "rule-id": "1",  
      "rule-name": "1",  
      "object-locator": {  
        "schema-name": "Test",  
        "table-name": "%"  
      },  
      "rule-action": "include"  
    },
```

```
{
    "rule-type": "object-mapping",
    "rule-id": "1",
    "rule-name": "TransformToKafka",
    "rule-action": "map-record-to-document",
    "object-locator": {
        "schema-name": "Test",
        "table-name": "Customer"
    },
    "mapping-parameters": {
        "partition-key": {
            "value": "ConstantPartitionKey"
        },
        "exclude-columns": [
            "FirstName",
            "LastName",
            "HomeAddress",
            "HomePhone",
            "WorkAddress",
            "WorkPhone"
        ],
        "attribute-mappings": [
            {
                "attribute-name": "CustomerName",
                "value": "${FirstName},${LastName}"
            },
            {
                "attribute-name": "ContactDetails",
                "value": {
                    "Home": {
                        "Address": "${HomeAddress}",
                        "Phone": "${HomePhone}"
                    },
                    "Work": {
                        "Address": "${WorkAddress}",
                        "Phone": "${WorkPhone}"
                    }
                }
            },
            {
                "attribute-name": "DateOfBirth",
                "value": "${DateOfBirth}"
            }
        ]
    }
}
```

#### Note

The partition-key value for a control record that is for a specific table is `TaskId.SchemaName.TableName`. The partition-key value for a control record that is for a specific task is that record's `TaskId`. Specifying a partition-key value in the object mapping has no impact on the partition-key for a control record.

## Multitopic replication using object mapping

By default, AWS DMS tasks migrate all source data to one of the Kafka topics following:

- As specified in the **Topic** field of the AWS DMS target endpoint.
- As specified by `kafka-default-topic` if the **Topic** field of the target endpoint isn't populated and the `Kafka auto.create.topics.enable` setting is set to `true`.

With AWS DMS engine versions 3.4.6 and later, you can use the `kafka-target-topic` attribute to map each migrated source table to a separate topic. For example, the object mapping rules following migrate the source tables `Customer` and `Address` to the Kafka topics `customer_topic` and `address_topic`, respectively. At the same time, AWS DMS migrates all other source tables, including the `Bills` table in the `Test` schema, to the topic specified in the target endpoint.

```
{  
    "rules": [  
        {  
            "rule-type": "selection",  
            "rule-id": "1",  
            "rule-name": "1",  
            "rule-action": "include",  
            "object-locator": {  
                "schema-name": "Test",  
                "table-name": "%"  
            }  
        },  
        {  
            "rule-type": "object-mapping",  
            "rule-id": "2",  
            "rule-name": "MapToKafka1",  
            "rule-action": "map-record-to-record",  
            "kafka-target-topic": "customer_topic",  
            "object-locator": {  
                "schema-name": "Test",  
                "table-name": "Customer"  
            },  
            "partition-key": {"value": "ConstantPartitionKey"}  
        },  
        {  
            "rule-type": "object-mapping",  
            "rule-id": "3",  
            "rule-name": "MapToKafka2",  
            "rule-action": "map-record-to-record",  
            "kafka-target-topic": "address_topic",  
            "object-locator": {  
                "schema-name": "Test",  
                "table-name": "Address"  
            },  
            "partition-key": {"value": "HomeAddress"}  
        },  
        {  
            "rule-type": "object-mapping",  
            "rule-id": "4",  
            "rule-name": "DefaultMapToKafka",  
            "rule-action": "map-record-to-record",  
            "object-locator": {  
                "schema-name": "Test",  
                "table-name": "Bills"  
            }  
        }  
    ]  
}
```

By using Kafka multtopic replication, you can group and migrate source tables to separate Kafka topics using a single replication task.

## Message format for Apache Kafka

The JSON output is simply a list of key-value pairs.

### RecordType

The record type can be either data or control. *Data records* represent the actual rows in the source. *Control records* are for important events in the stream, for example a restart of the task.

### Operation

For data records, the operation can be `load`, `insert`, `update`, or `delete`.

For control records, the operation can be `create-table`, `rename-table`, `drop-table`, `change-columns`, `add-column`, `drop-column`, `rename-column`, or `column-type-change`.

### SchemaName

The source schema for the record. This field can be empty for a control record.

### TableName

The source table for the record. This field can be empty for a control record.

### Timestamp

The timestamp for when the JSON message was constructed. The field is formatted with the ISO 8601 format.

The following JSON message example illustrates a data type message with all additional metadata.

```
{  
    "data": {  
        "id": 100000161,  
        "fname": "val61s",  
        "lname": "val61s",  
        "REGION": "val61s"  
    },  
    "metadata": {  
        "timestamp": "2019-10-31T22:53:59.721Z",  
        "record-type": "data",  
        "operation": "insert",  
        "partition-key-type": "primary-key",  
        "partition-key-value": "sbtest.sbtest_x.100000161",  
        "schema-name": "sbtest",  
        "table-name": "sbtest_x",  
        "transaction-id": 9324410911751,  
        "transaction-record-id": 1,  
        "prev-transaction-id": 9324410910341,  
        "prev-transaction-record-id": 10,  
        "commit-timestamp": "2019-10-31T22:53:55.00000Z",  
        "stream-position": "mysql-bin-changelog.002171:36912271:0:36912333:9324410911751:mysql-bin-changelog.002171:36912209"  
    }  
}
```

The following JSON message example illustrates a control type message.

```
{  
    "control": {  
        "table-def": {  
            "columns": {  
                "id": {  
                    "type": "WSTRING",  
                    "length": 512,  
                    "position": 1  
                }  
            }  
        }  
    }  
}
```

```
        "nullable":false
    },
    "fname":{
        "type":"WSTRING",
        "length":255,
        "nullable":true
    },
    "lname":{
        "type":"WSTRING",
        "length":255,
        "nullable":true
    },
    "REGION":{
        "type":"WSTRING",
        "length":1000,
        "nullable":true
    }
},
"primary-key":[
    "id"
],
"collation-name":"latin1_swedish_ci"
},
"metadata":{
    "timestamp":"2019-11-21T19:14:22.223792Z",
    "record-type":"control",
    "operation":"create-table",
    "partition-key-type":"task-id",
    "schema-name":"sbtest",
    "table-name":"sbtest_t1"
}
}
```

## Using an Amazon OpenSearch Service cluster as a target for AWS Database Migration Service

You can use AWS DMS to migrate data to Amazon OpenSearch Service (OpenSearch Service). OpenSearch Service is a managed service that makes it easy to deploy, operate, and scale an OpenSearch Service cluster.

In OpenSearch Service, you work with indexes and documents. An *index* is a collection of documents, and a *document* is a JSON object containing scalar values, arrays, and other objects. OpenSearch provides a JSON-based query language, so that you can query data in an index and retrieve the corresponding documents.

When AWS DMS creates indexes for a target endpoint for OpenSearch Service, it creates one index for each table from the source endpoint. The cost for creating an OpenSearch Service index depends on several factors. These are the number of indexes created, the total amount of data in these indexes, and the small amount of metadata that OpenSearch stores for each document.

Configure your OpenSearch Service cluster with compute and storage resources that are appropriate for the scope of your migration. We recommend that you consider the following factors, depending on the replication task you want to use:

- For a full data load, consider the total amount of data that you want to migrate, and also the speed of the transfer.
- For replicating ongoing changes, consider the frequency of updates, and your end-to-end latency requirements.

Also, configure the index settings on your OpenSearch cluster, paying close attention to the document count.

### Multithreaded full load task settings

To help increase the speed of the transfer, AWS DMS supports a multithreaded full load to an OpenSearch Service target cluster. AWS DMS supports this multithreading with task settings that include the following:

- **MaxFullLoadSubTasks** – Use this option to indicate the maximum number of source tables to load in parallel. DMS loads each table into its corresponding OpenSearch Service target index using a dedicated subtask. The default is 8; the maximum value is 49.
- **ParallelLoadThreads** – Use this option to specify the number of threads that AWS DMS uses to load each table into its OpenSearch Service target index. The maximum value for an OpenSearch Service target is 32. You can ask to have this maximum limit increased.

#### Note

If you don't change **ParallelLoadThreads** from its default (0), AWS DMS transfers a single record at a time. This approach puts undue load on your OpenSearch Service cluster. Make sure that you set this option to 1 or more.

- **ParallelLoadBufferSize** – Use this option to specify the maximum number of records to store in the buffer that the parallel load threads use to load data to the OpenSearch Service target. The default value is 50. The maximum value is 1,000. Use this setting with **ParallelLoadThreads**. **ParallelLoadBufferSize** is valid only when there is more than one thread.

For more information on how DMS loads an OpenSearch Service cluster using multithreading, see the AWS blog post [Scale Amazon OpenSearch Service for AWS Database Migration Service migrations](#).

### Multithreaded CDC load task settings

You can improve the performance of change data capture (CDC) for an OpenSearch Service target cluster using task settings to modify the behavior of the PutRecords API call. To do this, you can specify the number of concurrent threads, queues per thread, and the number of records to store in a buffer using **ParallelApply\*** task settings. For example, suppose you want to perform a CDC load and apply 32 threads in parallel. You also want to access 64 queues per thread, with 50 records stored per buffer.

#### Note

Support for the use of **ParallelApply\*** task settings during CDC to Amazon OpenSearch Service target endpoints is available in AWS DMS versions 3.4.0 and later.

To promote CDC performance, AWS DMS supports these task settings:

- **ParallelApplyThreads** – Specifies the number of concurrent threads that AWS DMS uses during a CDC load to push data records to a OpenSearch Service target endpoint. The default value is zero (0) and the maximum value is 32.
- **ParallelApplyBufferSize** – Specifies the maximum number of records to store in each buffer queue for concurrent threads to push to a OpenSearch Service target endpoint during a CDC load. The default value is 100 and the maximum value is 1,000. Use this option when **ParallelApplyThreads** specifies more than one thread.
- **ParallelApplyQueuesPerThread** – Specifies the number of queues that each thread accesses to take data records out of queues and generate a batch load for a OpenSearch Service endpoint during CDC.

When using **ParallelApply\*** task settings, the partition-key-type default is the primary-key of the table, not schema-name.table-name.

## Migrating from a relational database table to an OpenSearch Service index

AWS DMS supports migrating data to OpenSearch Service's scalar data types. When migrating from a relational database like Oracle or MySQL to OpenSearch Service, you might want to restructure how you store this data.

AWS DMS supports the following OpenSearch Service scalar data types:

- Boolean
- Date
- Float
- Int
- String

AWS DMS converts data of type Date into type String. You can specify custom mapping to interpret these dates.

AWS DMS doesn't support migration of LOB data types.

## Prerequisites for using Amazon OpenSearch Service as a target for AWS Database Migration Service

Before you begin work with an OpenSearch Service database as a target for AWS DMS, make sure that you create an AWS Identity and Access Management (IAM) role. This role should let AWS DMS access the OpenSearch Service indexes at the target endpoint. The minimum set of access permissions is shown in the following IAM policy.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "1",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "dms.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

The role that you use for the migration to OpenSearch Service must have the following permissions.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": [  
                "es:ESHttpDelete",  
                "es:ESHttpGet",  
                "es:ESHttpHead",  
                "es:ESHttpPost",  
                "es:ESHttpPut"  
            ]  
        }  
    ]  
}
```

```

        "es:ESHttpPost",
        "es:ESHttpPut"
    ],
    "Resource": "arn:aws:es:region:account-id:domain/domain-name/*"
}
]
}
}

```

In the preceding example, replace *region* with the AWS Region identifier, *account-id* with your AWS account ID, and *domain-name* with the name of your Amazon OpenSearch Service domain. An example is `arn:aws:es:us-west-2:123456789012:domain/my-es-domain`

## Extra connection attributes when using OpenSearch Service as a target for AWS DMS

When you set up your OpenSearch Service target endpoint, you can specify extra connection attributes. Extra connection attributes are specified by key-value pairs and separated by semicolons.

The following table describes the extra connection attributes available when using an OpenSearch Service instance as an AWS DMS source.

Attribute name	Valid values	Default value and description
<code>fullLoadErrorPercentage</code>	A positive integer greater than 0 but no larger than 100.	10 – For a full load task, this attribute determines the threshold of errors allowed before the task fails. For example, suppose that there are 1,500 rows at the source endpoint and this parameter is set to 10. Then the task fails if AWS DMS encounters more than 150 errors (10 percent of the row count) when writing to the target endpoint.
<code>errorRetryDuration</code>	A positive integer greater than 0.	300 – If an error occurs at the target endpoint, AWS DMS retries for this many seconds. Otherwise, the task fails.

## Limitations when using Amazon OpenSearch Service as a target for AWS Database Migration Service

The following limitations apply when using Amazon OpenSearch Service as a target:

- OpenSearch Service uses dynamic mapping (auto guess) to determine the data types to use for migrated data.
- OpenSearch Service stores each document with a unique ID. The following is an example ID.

```

"_id": "D359F8B537F1888BC71FE20B3D79EAE6674BE7ACA9B645B0279C7015F6FF19FD"

```

Each document ID is 64 bytes long, so anticipate this as a storage requirement. For example, if you migrate 100,000 rows from an AWS DMS source, the resulting OpenSearch Service index requires storage for an additional 6,400,000 bytes.

- With OpenSearch Service, you can't make updates to the primary key attributes. This restriction is important when using ongoing replication with change data capture (CDC) because it can result in unwanted data in the target. In CDC mode, primary keys are mapped to SHA256 values, which are 32 bytes long. These are converted to human-readable 64-byte strings, and are used as OpenSearch Service document IDs.

- If AWS DMS encounters any items that can't be migrated, it writes error messages to Amazon CloudWatch Logs. This behavior differs from that of other AWS DMS target endpoints, which write errors to an exceptions table.
- AWS DMS doesn't support connection to an Amazon ES cluster that has Fine-grained Access Control enabled with master user and password.

## Target data types for Amazon OpenSearch Service

When AWS DMS migrates data from heterogeneous databases, the service maps data types from the source database to intermediate data types called AWS DMS data types. The service then maps the intermediate data types to the target data types. The following table shows each AWS DMS data type and the data type it maps to in OpenSearch Service.

AWS DMS data type	OpenSearch Service data type
Boolean	boolean
Date	string
Time	date
Timestamp	date
INT4	integer
Real4	float
UINT4	integer

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

## Using Amazon DocumentDB as a target for AWS Database Migration Service

You can use AWS DMS to migrate data to Amazon DocumentDB (with MongoDB compatibility) from any of the source data engines that AWS DMS supports. The source engine can be on an AWS-managed service such as Amazon RDS, Aurora, or Amazon S3. Or the engine can be on a self-managed database, such as MongoDB running on Amazon EC2 or on-premises.

You can use AWS DMS to replicate source data to Amazon DocumentDB databases, collections, or documents.

**Note**

If your source endpoint is MongoDB or Amazon DocumentDB, run the migration in **Document mode**.

MongoDB stores data in a binary JSON format (BSON). AWS DMS supports all of the BSON data types that are supported by Amazon DocumentDB. For a list of these data types, see [Supported MongoDB APIs, operations, and data types in the Amazon DocumentDB Developer Guide](#).

If the source endpoint is a relational database, AWS DMS maps database objects to Amazon DocumentDB as follows:

- A relational database, or database schema, maps to an Amazon DocumentDB *database*.
- Tables within a relational database map to *collections* in Amazon DocumentDB.

- Records in a relational table map to *documents* in Amazon DocumentDB. Each document is constructed from data in the source record.

If the source endpoint is Amazon S3, then the resulting Amazon DocumentDB objects correspond to AWS DMS mapping rules for Amazon S3. For example, consider the following URI.

```
s3://mybucket/hr/employee
```

In this case, AWS DMS maps the objects in mybucket to Amazon DocumentDB as follows:

- The top-level URI part (hr) maps to an Amazon DocumentDB database.
- The next URI part (employee) maps to an Amazon DocumentDB collection.
- Each object in employee maps to a document in Amazon DocumentDB.

For more information on mapping rules for Amazon S3, see [Using Amazon S3 as a source for AWS DMS \(p. 254\)](#).

For additional details on working with Amazon DocumentDB as a target for AWS DMS, see the following sections:

#### Topics

- [Mapping data from a source to an Amazon DocumentDB target \(p. 409\)](#)
- [Ongoing replication with Amazon DocumentDB as a target \(p. 413\)](#)
- [Limitations to using Amazon DocumentDB as a target \(p. 414\)](#)
- [Target data types for Amazon DocumentDB \(p. 414\)](#)

#### Note

For a step-by-step walkthrough of the migration process, see [Migrating from MongoDB to Amazon DocumentDB](#) in the AWS Database Migration Service Step-by-Step Migration Guide.

## Mapping data from a source to an Amazon DocumentDB target

AWS DMS reads records from the source endpoint, and constructs JSON documents based on the data it reads. For each JSON document, AWS DMS must determine an `_id` field to act as a unique identifier. It then writes the JSON document to an Amazon DocumentDB collection, using the `_id` field as a primary key.

### Source data that is a single column

If the source data consists of a single column, the data must be of a string type. (Depending on the source engine, the actual data type might be VARCHAR, NVARCHAR, TEXT, LOB, CLOB, or similar.) AWS DMS assumes that the data is a valid JSON document, and replicates the data to Amazon DocumentDB as is.

If the resulting JSON document contains a field named `_id`, then that field is used as the unique `_id` in Amazon DocumentDB.

If the JSON doesn't contain an `_id` field, then Amazon DocumentDB generates an `_id` value automatically.

### Source data that is multiple columns

If the source data consists of multiple columns, then AWS DMS constructs a JSON document from all of these columns. To determine the `_id` field for the document, AWS DMS proceeds as follows:

- If one of the columns is named `_id`, then the data in that column is used as the `target_id`.
- If there is no `_id` column, but the source data has a primary key or a unique index, then AWS DMS uses that key or index value as the `_id` value. The data from the primary key or unique index also appears as explicit fields in the JSON document.
- If there is no `_id` column, and no primary key or a unique index, then Amazon DocumentDB generates an `_id` value automatically.

## Coercing a data type at the target endpoint

AWS DMS can modify data structures when it writes to an Amazon DocumentDB target endpoint. You can request these changes by renaming columns and tables at the source endpoint, or by providing transformation rules that are applied when a task is running.

### Using a nested JSON document (`json_` prefix)

To coerce a data type, you can prefix the source column name with `json_` (that is, `json_columnName`) either manually or using a transformation. In this case, the column is created as a nested JSON document within the target document, rather than as a string field.

For example, suppose that you want to migrate the following document from a MongoDB source endpoint.

```
{
  "_id": "1",
  "FirstName": "John",
  "LastName": "Doe",
  "ContactDetails": "{\"Home\": {\"Address\": \"Boston\", \"Phone\": \"1111111\"}, \"Work\":
    { \"Address\": \"Boston\", \"Phone\": \"2222222222\"}}"
}
```

If you don't coerce any of the source data types, the embedded `ContactDetails` document is migrated as a string.

```
{
  "_id": "1",
  "FirstName": "John",
  "LastName": "Doe",
  "ContactDetails": "{\"Home\": {\"Address\": \"Boston\", \"Phone\": \"1111111\"}, \"Work\"
    : { \"Address\": \"Boston\", \"Phone\": \"2222222222\"}}"
}
```

However, you can add a transformation rule to coerce `ContactDetails` to a JSON object. For example, suppose that the original source column name is `ContactDetails`. To coerce the data type as Nested JSON, the column at source endpoint needs to be renamed as `json_ContractDetails` either by adding `**json_**` prefix on the source manually or through transformation rules. For example, you can use the below transformation rule:

```
{
  "rules": [
    {
      "rule-type": "transformation",
      "rule-id": "1",
      "rule-name": "1",
      "rule-target": "column",
      "object-locator": {
        "schema-name": "%",
        "table-name": "%"
      }
    }
  ]
}
```

```

    "table-name": "%",
    "column-name": "ContactDetails"
    },
    "rule-action": "rename",
    "value": "json_ContactDetails",
    "old-value": null
}
]
}
}

```

AWS DMS replicates the ContactDetails field as nested JSON, as follows.

```
{
    "_id": "1",
    "FirstName": "John",
    "LastName": "Doe",
    "ContactDetails": {
        "Home": {
            "Address": "Boston",
            "Phone": "1111111111"
        },
        "Work": {
            "Address": "Boston",
            "Phone": "2222222222"
        }
    }
}
```

### Using a JSON array (array\_ prefix)

To coerce a data type, you can prefix a column name with `array_` (that is, `array_columnName`), either manually or using a transformation. In this case, AWS DMS considers the column as a JSON array, and creates it as such in the target document.

Suppose that you want to migrate the following document from a MongoDB source endpoint.

```
{
    "_id" : "1",
    "FirstName": "John",
    "LastName": "Doe",
    "ContactAddresses": ["Boston", "New York"],
    "ContactPhoneNumbers": ["1111111111", "2222222222"]
}
```

If you don't coerce any of the source data types, the embedded ContactDetails document is migrated as a string.

```
{
    "_id": "1",
    "FirstName": "John",
    "LastName": "Doe",
    "ContactAddresses": "[\"Boston\", \"New York\"]",
    "ContactPhoneNumbers": ["\"1111111111\", \"2222222222\"]"
}
```

However, you can add transformation rules to coerce ContactAddress and ContactPhoneNumbers to JSON arrays, as shown in the following table.

Original source column name	Renamed source column
ContactAddress	array_ContactAddress
ContactPhoneNumbers	array_ContactPhoneNumbers

AWS DMS replicates ContactAddress and ContactPhoneNumbers as follows.

```
{
    "_id": "1",
    "FirstName": "John",
    "LastName": "Doe",
    "ContactAddresses": [
        "Boston",
        "New York"
    ],
    "ContactPhoneNumbers": [
        "1111111111",
        "2222222222"
    ]
}
```

## Connecting to Amazon DocumentDB using TLS

By default, a newly created Amazon DocumentDB cluster accepts secure connections only using Transport Layer Security (TLS). When TLS is enabled, every connection to Amazon DocumentDB requires a public key.

You can retrieve the public key for Amazon DocumentDB by downloading the file, `rds-combined-ca-bundle.pem`, from an AWS-hosted Amazon S3 bucket. For more information on downloading this file, see [Encrypting connections using TLS](#) in the *Amazon DocumentDB Developer Guide*.

After you download this .pem file, you can import the public key that it contains into AWS DMS as described following.

### AWS Management Console

#### To import the public key (.pem) file

1. Open the AWS DMS console at <https://console.aws.amazon.com/dms>.
2. In the navigation pane, choose **Certificates**.
3. Choose **Import certificate** and do the following:
  - For **Certificate identifier**, enter a unique name for the certificate, for example docdb-cert.
  - For **Import file**, navigate to the location where you saved the .pem file.

When the settings are as you want them, choose **Add new CA certificate**.

### AWS CLI

Use the `aws dms import-certificate` command, as shown in the following example.

```
aws dms import-certificate \
```

```
--certificate-identifier docdb-cert \
--certificate-pem file://./rds-combined-ca-bundle.pem
```

When you create an AWS DMS target endpoint, provide the certificate identifier (for example, docdb-cert). Also, set the SSL mode parameter to verify-full.

## Ongoing replication with Amazon DocumentDB as a target

If ongoing replication is enabled, AWS DMS ensures that documents in Amazon DocumentDB stay in sync with the source. When a source record is created or updated, AWS DMS must first determine which Amazon DocumentDB record is affected by doing the following:

- If the source record has a column named `_id`, the value of that column determines the corresponding `_id` in the Amazon DocumentDB collection.
- If there is no `_id` column, but the source data has a primary key or unique index, then AWS DMS uses that key or index value as the `_id` for the Amazon DocumentDB collection.
- If the source record doesn't have an `_id` column, a primary key, or a unique index, then AWS DMS matches all of the source columns to the corresponding fields in the Amazon DocumentDB collection.

When a new source record is created, AWS DMS writes a corresponding document to Amazon DocumentDB. If an existing source record is updated, AWS DMS updates the corresponding fields in the target document in Amazon DocumentDB. Any fields that exist in the target document but not in the source record remain untouched.

When a source record is deleted, AWS DMS deletes the corresponding document from Amazon DocumentDB.

### Structural changes (DDL) at the source

With ongoing replication, any changes to source data structures (such as tables, columns, and so on) are propagated to their counterparts in Amazon DocumentDB. In relational databases, these changes are initiated using data definition language (DDL) statements. You can see how AWS DMS propagates these changes to Amazon DocumentDB in the following table.

DDL at source	Effect at Amazon DocumentDB target
CREATE TABLE	Creates an empty collection.
Statement that renames a table (RENAME TABLE, ALTER TABLE...RENAME, and similar)	Renames the collection.
TRUNCATE TABLE	Removes all the documents from the collection, but only if HandleSourceTableTruncated is true. For more information, see <a href="#">Task settings for change processing DDL handling (p. 468)</a> .
DROP TABLE	Deletes the collection, but only if HandleSourceTableDropped is true. For more information, see <a href="#">Task settings for change processing DDL handling (p. 468)</a> .
Statement that adds a column to a table (ALTER TABLE...ADD and similar)	The DDL statement is ignored, and a warning is issued. When the first INSERT is performed at the source, the new field is added to the target document.

DDL at source	Effect at Amazon DocumentDB target
ALTER TABLE...RENAME COLUMN	The DDL statement is ignored, and a warning is issued. When the first INSERT is performed at the source, the newly named field is added to the target document.
ALTER TABLE...DROP COLUMN	The DDL statement is ignored, and a warning is issued.
Statement that changes the column data type (ALTER COLUMN...MODIFY and similar)	The DDL statement is ignored, and a warning is issued. When the first INSERT is performed at the source with the new data type, the target document is created with a field of that new data type.

## Limitations to using Amazon DocumentDB as a target

The following limitations apply when using Amazon DocumentDB as a target for AWS DMS:

- In Amazon DocumentDB, collection names can't contain the dollar symbol (\$). In addition, database names can't contain any Unicode characters.
- AWS DMS doesn't support merging of multiple source tables into a single Amazon DocumentDB collection.
- When AWS DMS processes changes from a source table that doesn't have a primary key, any LOB columns in that table are ignored.
- If the **Change table** option is enabled and AWS DMS encounters a source column named "`_id`", then that column appears as "`__id`" (two underscores) in the change table.
- If you choose Oracle as a source endpoint, then the Oracle source must have full supplemental logging enabled. Otherwise, if there are columns at the source that weren't changed, then the data is loaded into Amazon DocumentDB as null values.
- The replication task setting, TargetTablePrepMode : TRUNCATE\_BEFORE\_LOAD isn't supported for use with a DocumentDB target endpoint.

## Target data types for Amazon DocumentDB

In the following table, you can find the Amazon DocumentDB target data types that are supported when using AWS DMS, and the default mapping from AWS DMS data types. For more information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

AWS DMS data type	Amazon DocumentDB data type
BOOLEAN	Boolean
BYTES	Binary data
DATE	Date
TIME	String (UTF8)
DATETIME	Date
INT1	32-bit integer
INT2	32-bit integer

AWS DMS data type	Amazon DocumentDB data type
INT4	32-bit integer
INT8	64-bit integer
NUMERIC	String (UTF8)
REAL4	Double
REAL8	Double
STRING	If the data is recognized as JSON, then AWS DMS migrates it to Amazon DocumentDB as a document. Otherwise, the data is mapped to String (UTF8).
UINT1	32-bit integer
UINT2	32-bit integer
UINT4	64-bit integer
UINT8	String (UTF8)
WSTRING	If the data is recognized as JSON, then AWS DMS migrates it to Amazon DocumentDB as a document. Otherwise, the data is mapped to String (UTF8).
BLOB	Binary
CLOB	If the data is recognized as JSON, then AWS DMS migrates it to Amazon DocumentDB as a document. Otherwise, the data is mapped to String (UTF8).
NCLOB	If the data is recognized as JSON, then AWS DMS migrates it to Amazon DocumentDB as a document. Otherwise, the data is mapped to String (UTF8).

## Using Amazon Neptune as a target for AWS Database Migration Service

Amazon Neptune is a fast, reliable, fully managed graph database service that makes it easy to build and run applications that work with highly connected datasets. The core of Neptune is a purpose-built, high-performance graph database engine. This engine is optimized for storing billions of relationships and querying the graph with milliseconds latency. Neptune supports the popular graph query languages Apache TinkerPop Gremlin and W3C's SPARQL. For more information on Amazon Neptune, see [What is Amazon Neptune?](#) in the *Amazon Neptune User Guide*.

Without a graph database such as Neptune, you probably model highly connected data in a relational database. Because the data has potentially dynamic connections, applications that use such data sources have to model connected data queries in SQL. This approach requires you to write an extra layer to convert graph queries into SQL. Also, relational databases come with schema rigidity. Any changes in the schema to model changing connections require downtime and additional maintenance of the query conversion to support the new schema. The query performance is also another big constraint to consider while designing your applications.

Graph databases can greatly simplify such situations. Free from a schema, a rich graph query layer (Gremlin or SPARQL) and indexes optimized for graph queries increase flexibility and performance.

The Amazon Neptune graph database also has enterprise features such as encryption at rest, a secure authorization layer, default backups, Multi-AZ support, read replica support, and others.

Using AWS DMS, you can migrate relational data that models a highly connected graph to a Neptune target endpoint from a DMS source endpoint for any supported SQL database.

For more details, see the following.

#### Topics

- [Overview of migrating to Amazon Neptune as a target \(p. 416\)](#)
- [Specifying endpoint settings for Amazon Neptune as a target \(p. 417\)](#)
- [Creating an IAM service role for accessing Amazon Neptune as a target \(p. 418\)](#)
- [Specifying graph-mapping rules using Gremlin and R2RML for Amazon Neptune as a target \(p. 420\)](#)
- [Data types for Gremlin and R2RML migration to Amazon Neptune as a target \(p. 423\)](#)
- [Limitations of using Amazon Neptune as a target \(p. 425\)](#)

## Overview of migrating to Amazon Neptune as a target

Before starting a migration to a Neptune target, create the following resources in your AWS account:

- A Neptune cluster for the target endpoint.
- A SQL relational database supported by AWS DMS for the source endpoint.
- An Amazon S3 bucket for the target endpoint. Create this S3 bucket in the same AWS Region as your Neptune cluster. AWS DMS uses this S3 bucket as intermediate file storage for the target data that it bulk loads to the Neptune database. For more information on creating an S3 bucket, see [Creating a bucket](#) in the *Amazon Simple Storage Service User Guide*.
- A virtual private cloud (VPC) endpoint for S3 in the same VPC as the Neptune cluster.
- An AWS Identity and Access Management (IAM) role that includes an IAM policy. This policy should specify the GetObject, PutObject, DeleteObject and ListObject permissions to the S3 bucket for your target endpoint. This role is assumed by both AWS DMS and Neptune with IAM access to both the target S3 bucket and the Neptune database. For more information, see [Creating an IAM service role for accessing Amazon Neptune as a target \(p. 418\)](#).

After you have these resources, setting up and starting a migration to a Neptune target is similar to any full load migration using the console or DMS API. However, a migration to a Neptune target requires some unique steps.

### To migrate an AWS DMS relational database to Neptune

1. Create a replication instance as described in [Creating a replication instance \(p. 119\)](#).
2. Create and test a SQL relational database supported by AWS DMS for the source endpoint.
3. Create and test the target endpoint for your Neptune database.

To connect the target endpoint to the Neptune database, specify the server name for either the Neptune cluster endpoint or the Neptune writer instance endpoint. Also, specify the S3 bucket folder for AWS DMS to store its intermediate files for bulk load to the Neptune database.

During migration, AWS DMS stores all migrated target data in this S3 bucket folder up to a maximum file size that you specify. When this file storage reaches this maximum size, AWS DMS bulk loads the stored S3 data into the target database. It clears the folder to enable storage of any additional target data for subsequent loading to the target database. For more information on specifying these settings, see [Specifying endpoint settings for Amazon Neptune as a target \(p. 417\)](#).

4. Create a full-load replication task with the resources created in steps 1–3 and do the following:
  - a. Use task table mapping as usual to identify specific source schemas, tables, and views to migrate from your relational database using appropriate selection and transformation rules. For more information, see [Using table mapping to specify task settings \(p. 491\)](#).
  - b. Specify target mappings by choosing one of the following to specify mapping rules from source tables and views to your Neptune target database graph:
    - Gremlin JSON – For information on using Gremlin JSON to load a Neptune database, see [Gremlin load data format](#) in the *Amazon Neptune User Guide*.
    - SPARQL RDB to Resource Description Framework Mapping Language (R2RML) – For information on using SPARQL R2RML, see the W3C specification [R2RML: RDB to RDF mapping language](#).
  - c. Do one of the following:
    - Using the AWS DMS console, specify graph-mapping options using **Graph mapping rules** on the [Create database migration task](#) page.
    - Using the AWS DMS API, specify these options using the TaskData request parameter of the `CreateReplicationTask` API call.

For more information and examples using Gremlin JSON and SPARQL R2RML to specify graph-mapping rules, see [Specifying graph-mapping rules using Gremlin and R2RML for Amazon Neptune as a target \(p. 420\)](#).

5. Start the replication for your migration task.

## Specifying endpoint settings for Amazon Neptune as a target

To create or modify a target endpoint, you can use the console or the `CreateEndpoint` or `ModifyEndpoint` API operations.

For a Neptune target in the AWS DMS console, specify **Endpoint-specific settings** on the [Create endpoint](#) or [Modify endpoint](#) console page. For `CreateEndpoint` and `ModifyEndpoint`, specify request parameters for the `NeptuneSettings` option. The following example shows how to do this using the CLI.

```
dms create-endpoint --endpoint-identifier my-neptune-target-endpoint
--endpoint-type target --engine-name neptune
--server-name my-neptune-db.cluster-cspckvk1bvgf.us-east-1.neptune.amazonaws.com
--port 8192
--neptune-settings
'{"ServiceAccessRoleArn":"arn:aws:iam::123456789012:role/myNeptuneRole",
 "S3BucketName":"my-bucket",
 "S3BucketFolder":"my-bucket-folder",
 "ErrorRetryDuration":57,
 "MaxFileSize":100,
 "MaxRetryCount": 10,
 "IAMAuthEnabled":false}'
```

Here, the CLI `--server-name` option specifies the server name for the Neptune cluster writer endpoint. Or you can specify the server name for a Neptune writer instance endpoint.

The `--neptune-settings` option request parameters follow:

- `ServiceAccessRoleArn` – (Required) The Amazon Resource Name (ARN) of the service role that you created for the Neptune target endpoint. For more information, see [Creating an IAM service role for accessing Amazon Neptune as a target \(p. 418\)](#).

- **S3BucketName** – (Required) The name of the S3 bucket where DMS can temporarily store migrated graph data in .csv files before bulk loading it to the Neptune target database. DMS maps the SQL source data to graph data before storing it in these .csv files.
- **S3BucketFolder** – (Required) A folder path where you want DMS to store migrated graph data in the S3 bucket specified by **S3BucketName**.
- **ErrorRetryDuration** – (Optional) The number of milliseconds for DMS to wait to retry a bulk load of migrated graph data to the Neptune target database before raising an error. The default is 250.
- **MaxFileSize** – (Optional) The maximum size in KB of migrated graph data stored in a .csv file before DMS bulk loads the data to the Neptune target database. The default is 1,048,576 KB (1 GB). If successful, DMS clears the bucket, ready to store the next batch of migrated graph data.
- **MaxRetryCount** – (Optional) The number of times for DMS to retry a bulk load of migrated graph data to the Neptune target database before raising an error. The default is 5.
- **IAMAuthEnabled** – (Optional) If you want IAM authorization enabled for this endpoint, set this parameter to true and attach the appropriate IAM policy document to your service role specified by **ServiceAccessRoleArn**. The default is false.

## Creating an IAM service role for accessing Amazon Neptune as a target

To access Neptune as a target, create a service role using IAM. Depending on your Neptune endpoint configuration, attach to this role some or all of the following IAM policy and trust documents. When you create the Neptune endpoint, you provide the ARN of this service role. Doing so enables AWS DMS and Amazon Neptune to assume permissions to access both Neptune and its associated Amazon S3 bucket.

If you set the **IAMAuthEnabled** parameter in **NeptuneSettings** to **true** in your Neptune endpoint configuration, attach an IAM policy like the following to your service role. If you set **IAMAuthEnabled** to **false**, you can ignore this policy.

```
// Policy to access Neptune

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "neptune-db:*",
            "Resource": "arn:aws:neptune-db:us-east-1:123456789012:cluster-CLG7H7FHK54AZGHEH6MNS55JKM/*"
        }
    ]
}
```

The preceding IAM policy allows full access to the Neptune target cluster specified by **Resource**.

Attach an IAM policy like the following to your service role. This policy allows DMS to temporarily store migrated graph data in the S3 bucket that you created for bulk loading to the Neptune target database.

```
//Policy to access S3 bucket

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListObjectsInBucket0",
```

```
        "Effect": "Allow",
        "Action": "s3>ListBucket",
        "Resource": [
            "arn:aws:s3:::my-bucket"
        ]
    },
    {
        "Sid": "AllObjectActions",
        "Effect": "Allow",
        "Action":
            "s3:GetObject",
            "s3:PutObject",
            "s3>DeleteObject",

        "Resource": [
            "arn:aws:s3:::my-bucket/"
        ],
        {
            "Sid": "ListObjectsInBucket1",
            "Effect": "Allow",
            "Action": "s3>ListBucket",
            "Resource": [
                "arn:aws:s3:::my-bucket",
                "arn:aws:s3:::my-bucket/"
            ]
        }
    ]
}
```

The preceding IAM policy allows your account to query the contents of the S3 bucket (`arn:aws:s3:::my-bucket`) created for your Neptune target. It also allows your account to fully operate on the contents of all bucket files and folders (`arn:aws:s3:::my-bucket/`).

Edit the trust relationship and attach the following IAM role to your service role to allow both AWS DMS and Amazon Neptune database service to assume the role.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "dms.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Sid": "neptune",
            "Effect": "Allow",
            "Principal": {
                "Service": "rds.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

For information about specifying this service role for your Neptune target endpoint, see [Specifying endpoint settings for Amazon Neptune as a target \(p. 417\)](#).

## Specifying graph-mapping rules using Gremlin and R2RML for Amazon Neptune as a target

The graph-mapping rules that you create specify how data extracted from an SQL relational database source is loaded into a Neptune database cluster target. The format of these mapping rules differs depending on whether the rules are for loading property-graph data using Apache TinkerPop Gremlin or Resource Description Framework (RDF) data using R2RML. Following, you can find information about these formats and where to learn more.

You can specify these mapping rules when you create the migration task using either the console or DMS API.

Using the console, specify these mapping rules using **Graph mapping rules** on the **Create database migration task** page. In **Graph mapping rules**, you can enter and edit the mapping rules directly using the editor provided. Or you can browse for a file that contains the mapping rules in the appropriate graph-mapping format.

Using the API, specify these options using the `TaskData` request parameter of the `CreateReplicationTask` API call. Set `TaskData` to the path of a file containing the mapping rules in the appropriate graph-mapping format.

### Graph-mapping rules for generating property-graph data using Gremlin

Using Gremlin to generate the property-graph data, specify a JSON object with a mapping rule for each graph entity to be generated from the source data. The format of this JSON is defined specifically for bulk loading Amazon Neptune. The following template shows what each rule in this object looks like.

```
{  
    "rules": [  
        {  
            "rule_id": "(an identifier for this rule)",  
            "rule_name": "(a name for this rule)",  
            "table_name": "(the name of the table or view being loaded)",  
            "vertex_definitions": [  
                {  
                    "vertex_id_template": "{col1}",  
                    "vertex_label": "(the vertex to create)",  
                    "vertex_definition_id": "(an identifier for this vertex)",  
                    "vertex_properties": [  
                        {  
                            "property_name": "(name of the property)",  
                            "property_value_template": "{col2} or text",  
                            "property_value_type": "(data type of the property)"  
                        }  
                    ]  
                }  
            ]  
        },  
        {  
            "rule_id": "(an identifier for this rule)",  
            "rule_name": "(a name for this rule)",  
            "table_name": "(the name of the table or view being loaded)",  
            "edge_definitions": [  
                {  
                    "from_vertex": {  
                        "vertex_id_template": "{col1}",  
                        "vertex_definition_id": "(an identifier for the vertex referenced  
above)"  
                    },  
                    "to_vertex": {  
                        "vertex_id_template": "{col2}",  
                        "vertex_definition_id": "(an identifier for the vertex referenced  
above)"  
                    },  
                    "edge_label": "(the edge label)",  
                    "edge_definition_id": "(an identifier for this edge)",  
                    "edge_properties": [  
                        {  
                            "property_name": "(name of the property)",  
                            "property_value_template": "{col3} or text",  
                            "property_value_type": "(data type of the property)"  
                        }  
                    ]  
                }  
            ]  
        }  
    ]  
}
```

```

        "to_vertex": {
            "vertex_id_template": "{col3}",
            "vertex_definition_id": "(an identifier for the vertex referenced
above)"
        },
        "edge_id_template": {
            "label": "(the edge label to add)",
            "template": "{col1}_{col3}"
        },
        "edge_properties": [
            {
                "property_name": "(the property to add)",
                "property_value_template": "{col4} or text",
                "property_value_type": "(data type like String, int, double)"
            }
        ]
    }
}

```

The presence of a vertex label implies that the vertex is being created here. Its absence implies that the vertex is created by a different source, and this definition is only adding vertex properties. Specify as many vertex and edge definitions as required to specify the mappings for your entire relational database source.

A sample rule for an employee table follows.

```

{
    "rules": [
        {
            "rule_id": "1",
            "rule_name": "vertex_mapping_rule_from_nodes",
            "table_name": "nodes",
            "vertex_definitions": [
                {
                    "vertex_id_template": "{emp_id}",
                    "vertex_label": "employee",
                    "vertex_definition_id": "1",
                    "vertex_properties": [
                        {
                            "property_name": "name",
                            "property_value_template": "{emp_name}",
                            "property_value_type": "String"
                        }
                    ]
                }
            ],
            "rule_id": "2",
            "rule_name": "edge_mapping_rule_from_emp",
            "table_name": "nodes",
            "edge_definitions": [
                {
                    "from_vertex": {
                        "vertex_id_template": "{emp_id}",
                        "vertex_definition_id": "1"
                    },
                    "to_vertex": {
                        "vertex_id_template": "{mgr_id}",

```

```
        "vertex_definition_id": "1"
    },
    "edge_id_template": {
        "label": "reportsTo",
        "template": "{emp_id}_{mgr_id}"
    },
    "edge_properties": [
        {
            "property_name": "team",
            "property_value_template": "{team}",
            "property_value_type": "String"
        }
    ]
}
]
```

Here, the vertex and edge definitions map a reporting relationship from an employee node with employee ID (EmpID) and an employee node with a manager ID (managerId).

For more information about creating graph-mapping rules using Gremlin JSON, see [Gremlin load data format](#) in the *Amazon Neptune User Guide*.

Graph-mapping rules for generating RDF/SPARQL data

If you are loading RDF data to be queried using SPARQL, write the graph-mapping rules in R2RML. R2RML is a standard W3C language for mapping relational data to RDF. In an R2RML file, a *triples map* (for example, <#TriplesMap1> following) specifies a rule for translating each row of a logical table to zero or more RDF triples. A *subject map* (for example, any rr:subjectMap following) specifies a rule for generating the subjects of the RDF triples generated by a triples map. A *predicate-object map* (for example, any rr:predicateObjectMap following) is a function that creates one or more predicate-object pairs for each logical table row of a logical table.

A simple example for a nodes table follows.

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>
  rr:logicalTable [ rr:tableName "nodes" ];
  rr:subjectMap [
    rr:template "http://data.example.com/employee/{id}";
    rr:class ex:Employee;
  ];
  rr:predicateObjectMap [
    rr:predicate ex:name;
    rr:objectMap [ rr:column "label" ];
  ]
]
```

In the previous example, the mapping defines graph nodes mapped from a table of employees.

Another simple example for a Student table follows.

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.  
@prefix ex: <http://example.com/#>.  
@prefix foaf: <http://xmlns.com/foaf/0.1/>.  
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
```

```
<#TriplesMap2>
  rr:logicalTable [ rr:tableName "Student" ];
  rr:subjectMap [ rr:template "http://example.com/{ID}{Name}";
                  rr:class foaf:Person ];
  rr:predicateObjectMap [
    rr:predicate ex:id ;
    rr:objectMap [ rr:column "ID";
                   rr:datatype xsd:integer ]
  ];
  rr:predicateObjectMap [
    rr:predicate foaf:name ;
    rr:objectMap [ rr:column "Name" ]
  ].
```

In the previous example, the mapping defines graph nodes mapping friend-of-a-friend relationships between persons in a Student table.

For more information about creating graph-mapping rules using SPARQL R2RML, see the W3C specification [R2RML: RDB to RDF mapping language](#).

## Data types for Gremlin and R2RML migration to Amazon Neptune as a target

AWS DMS performs data type mapping from your SQL source endpoint to your Neptune target in one of two ways. Which way you use depends on the graph mapping format that you're using to load the Neptune database:

- Apache TinkerPop Gremlin, using a JSON representation of the migration data.
- W3C's SPARQL, using an R2RML representation of the migration data.

For more information on these two graph mapping formats, see [Specifying graph-mapping rules using Gremlin and R2RML for Amazon Neptune as a target \(p. 420\)](#).

Following, you can find descriptions of the data type mappings for each format.

### SQL source to Gremlin target data type mappings

The following table shows the data type mappings from a SQL source to a Gremlin formatted target.

AWS DMS maps any unlisted SQL source data type to a Gremlin `String`.

SQL source data types	Gremlin target data types
NUMERIC (and variants)	Double
DECIMAL	
TINYINT	Byte
SMALLINT	Short
INT, INTEGER	Int
BIGINT	Long
FLOAT	Float

SQL source data types	Gremlin target data types
DOUBLE PRECISION	
REAL	Double
BIT	Boolean
BOOLEAN	
DATE	Date
TIME	
TIMESTAMP	
CHARACTER (and variants)	String

For more information on the Gremlin data types for loading Neptune, see [Gremlin data types in the Neptune User Guide](#).

## SQL source to R2RML (RDF) target data type mappings

The following table shows the data type mappings from a SQL source to an R2RML formatted target.

All listed RDF data types are case-sensitive, except RDF literal. AWS DMS maps any unlisted SQL source data type to an RDF literal.

An *RDF literal* is one of a variety of literal lexical forms and data types. For more information, see [RDF literals](#) in the W3C specification *Resource Description Framework (RDF): Concepts and Abstract Syntax*.

SQL source data types	R2RML (RDF) target data types
BINARY (and variants)	xsd:hexBinary
NUMERIC (and variants)	xsd:decimal
DECIMAL	
TINYINT	xsd:integer
SMALLINT	
INT, INTEGER	
BIGINT	
FLOAT	xsd:double
DOUBLE PRECISION	
REAL	
BIT	xsd:boolean
BOOLEAN	
DATE	xsd:date
TIME	xsd:time

SQL source data types	R2RML (RDF) target data types
TIMESTAMP	xsd:dateTime
CHARACTER (and variants)	RDF literal

For more information on the RDF data types for loading Neptune and their mappings to SQL source data types, see [Datatype conversions in the W3C specification R2RML: RDB to RDF Mapping Language](#).

## Limitations of using Amazon Neptune as a target

The following limitations apply when using Neptune as a target:

- AWS DMS currently supports full load tasks only for migration to a Neptune target. Change data capture (CDC) migration to a Neptune target isn't supported.
- Make sure that your target Neptune database is manually cleared of all data before starting the migration task, as in the following examples.

To drop all data (vertices and edges) within the graph, run the following Gremlin command.

```
gremlin> g.V().drop().iterate()
```

To drop vertices that have the label 'customer', run the following Gremlin command.

```
gremlin> g.V().hasLabel('customer').drop()
```

### Note

It can take some time to drop a large dataset. You might want to iterate drop( ) with a limit, for example, limit(1000).

To drop edges that have the label 'rated', run the following Gremlin command.

```
gremlin> g.E().hasLabel('rated').drop()
```

### Note

It can take some time to drop a large dataset. You might want to iterate drop( ) with a limit, for example limit(1000).

- The DMS API operation `DescribeTableStatistics` can return inaccurate results about a given table because of the nature of Neptune graph data structures.

During migration, AWS DMS scans each source table and uses graph mapping to convert the source data into a Neptune graph. The converted data is first stored in the S3 bucket folder specified for the target endpoint. If the source is scanned and this intermediate S3 data is generated successfully, `DescribeTableStatistics` assumes that the data was successfully loaded into the Neptune target database. But this isn't always true. To verify that the data was loaded correctly for a given table, compare `count()` return values at both ends of the migration for that table.

In the following example, AWS DMS has loaded a `customer` table from the source database, which is assigned the label 'customer' in the target Neptune database graph. You can make sure that this label is written to the target database. To do this, compare the number of `customer` rows available from the source database with the number of 'customer' labeled rows loaded in the Neptune target database after the task completes.

To get the number of customer rows available from the source database using SQL, run the following.

```
select count(*) from customer;
```

To get the number of 'customer' labeled rows loaded into the target database graph using Gremlin, run the following.

```
gremlin> g.V().hasLabel('customer').count()
```

- Currently, if any single table fails to load, the whole task fails. Unlike in a relational database target, data in Neptune is highly connected, which makes it impossible in many cases to resume a task. If a task can't be resumed successfully because of this type of data load failure, create a new task to load the table that failed to load. Before running this new task, manually clear the partially loaded table from the Neptune target.

**Note**

You can resume a task that fails migration to a Neptune target if the failure is recoverable (for example, a network transit error).

- AWS DMS supports most standards for R2RML. However, AWS DMS doesn't support certain R2RML standards, including inverse expressions, joins, and views. A work-around for an R2RML view is to create a corresponding custom SQL view in the source database. In the migration task, use table mapping to choose the view as input. Then map the view to a table that is then consumed by R2RML to generate graph data.
- When you migrate source data with unsupported SQL data types, the resulting target data can have a loss of precision. For more information, see [Data types for Gremlin and R2RML migration to Amazon Neptune as a target \(p. 423\)](#).
- AWS DMS doesn't support migrating LOB data into a Neptune target.

## Using Redis as a target for AWS Database Migration Service

Redis is an open-source in-memory data structure store used as a database, cache, and message broker. Managing data in-memory can result in read or write operations taking less than a millisecond, and hundreds of millions of operations performed each second. As an in-memory data store, Redis powers the most demanding applications requiring sub-millisecond response times.

Using AWS DMS, you can migrate data from any supported source database to a target Redis data store with minimal downtime. For additional information about Redis see, [Redis Documentation](#).

In addition to on-premises Redis, AWS Database Migration Service supports the following:

- [Amazon ElastiCache for Redis](#) as a target data store. ElastiCache for Redis works with your Redis clients and uses the open Redis data format to store your data.
- [Amazon MemoryDB for Redis](#) as a target data store. MemoryDB is compatible with Redis and enables you to build applications using all the Redis data structures, APIs, and commands in use today.

For additional information about working with Redis as a target for AWS DMS, see the following sections:

### Topics

- [Prerequisites for using a Redis cluster as a target for AWS DMS \(p. 427\)](#)
- [Limitations when using Redis as a target for AWS Database Migration Service \(p. 427\)](#)
- [Migrating data from a relational or non-relational database to a Redis target \(p. 427\)](#)
- [Specifying endpoint settings for Redis as a target \(p. 430\)](#)

## Prerequisites for using a Redis cluster as a target for AWS DMS

DMS supports an on-premises Redis target in a standalone configuration, or as a Redis cluster where data is automatically *sharded* across multiple nodes. Sharding is the process of separating data into smaller chunks called shards that are spread across multiple servers or nodes. In effect, a shard is a data partition that contains a subset of the total data set, and serves a slice of the overall workload.

Since Redis is a key-value NoSQL data store, the Redis key naming convention to use when your source is a relational database, is **schema-name.table-name.primary-key**. In Redis, the key and value must not contain the special character %. Otherwise, DMS skips the record.

### Note

If you are using ElastiCache for Redis as a target, DMS supports *cluster mode enabled* configurations only. For more information about using ElastiCache for Redis version 6.x or later to create a cluster mode enabled target data store, see [Getting started](#) in the *Amazon ElastiCache for Redis User Guide*.

Before you begin a database migration, launch your Redis cluster with the following criteria.

- Your cluster has one or more shards.
- If you're using an ElastiCache for Redis target, ensure that your cluster doesn't use IAM role-based access control. Instead, use Redis Auth to authenticate users.
- Enable Multi-AZ (Availability Zones).
- Ensure the cluster has sufficient memory available to fit the data to be migrated from your database.
- Make sure that your target Redis cluster is clear of all data before starting the initial migration task.

You should determine your security requirements for the data migration prior to creating your cluster configuration. DMS supports migration to target replication groups regardless of their encryption configuration. But you can enable or disable encryption only when you create your cluster configuration.

## Limitations when using Redis as a target for AWS Database Migration Service

The following limitations apply when using Redis as a target:

- Since Redis is a key-value no-sql data store, the Redis key naming convention to use when your source is a relational database, is schema-name.table-name.primary-key.
- In Redis, the key-value can't contain the special character %. Otherwise, DMS skips the record.
- DMS won't migrate rows that contain special characters.
- DMS won't migrate fields that contain special characters in the field name.
- Full LOB mode is not supported.
- A private Certificate Authority (CA) isn't supported when using ElastiCache for Redis as a target.

## Migrating data from a relational or non-relational database to a Redis target

You can migrate data from any source SQL or NoSQL data store directly to a Redis target. Setting up and starting a migration to a Redis target is similar to any full load and change data capture migration using the DMS console or API. To perform a database migration to a Redis target, you do the following.

- Create a replication instance to perform all the processes for the migration. For more information, see [Creating a replication instance \(p. 119\)](#).

- Specify a source endpoint. For more information, see [Creating source and target endpoints \(p. 131\)](#).
- Locate the DNS name and port number of your cluster.
- Download a certificate bundle that you can use to verify SSL connections.
- Specify a target endpoint, as described below.
- Create a task or set of tasks to define what tables and replication processes you want to use. For more information, see [Creating a task \(p. 442\)](#).
- Migrate data from your source database to your target cluster.

You begin a database migration in one of two ways:

1. You can choose the AWS DMS console and perform each step there.
2. You can use the AWS Command Line Interface (AWS CLI). For more information about using the CLI with AWS DMS, see [AWS CLI for AWS DMS](#).

### To locate the DNS name and port number of your cluster

- Use the following AWS CLI command to provide the `replication-group-id` with the name of your replication group.

```
aws elasticache describe-replication-groups --replication-group-id myreplgroup
```

Here, the output shows the DNS name in the `Address` attribute and the port number in the `Port` attribute of the primary node in the cluster.

```
...
"ReadEndpoint": {
"Port": 6379,
"Address": "myreplgroup-
111.1abc1d.1111.uuu1.cache.example.com"
}
...
```

If you are using MemoryDB for Redis as your target, use the following AWS CLI command to provide an endpoint address to your Redis cluster.

```
aws memorydb describe-clusters --clusterid clusterid
```

### Download a certificate bundle for use to verify SSL connections

- Enter the following `wget` command at the command line. `Wget` is a free GNU command-line utility tool used to download files from the internet.

```
wget https://s3.amazonaws.com/rds-downloads/rds-combined-ca-bundle.pem
```

Here, `aws-api-domain` completes the Amazon S3 domain in your AWS Region required to access the specified S3 bucket and the `rds-combined-ca-bundle.pem` file that it provides.

## To create a target endpoint using the AWS DMS console

This endpoint is for your Redis target that is already running.

- On the console, choose **Endpoints** from the navigation pane and then choose **Create Endpoint**. The following table describes the settings.

For this option	Do this
<b>Endpoint type</b>	Choose the <b>Target</b> endpoint type.
<b>Endpoint identifier</b>	Enter the name of your endpoint. For example, include the type of endpoint in the name, such as <b>my-redis-target</b> .
<b>Target engine</b>	Choose <b>Redis</b> as the type of database engine that you want this endpoint to connect.
<b>Cluster name</b>	Enter the DNS name of your Redis cluster.
<b>Port</b>	Enter the port number of your Redis cluster.
<b>SSL security protocol</b>	<p>Choose either <b>Plain text</b> or <b>SSL encryption</b>.</p> <p><b>Plain text</b>—This option doesn't provide Transport Layer Security (TLS) encryption for traffic between endpoint and database.</p> <p><b>SSL encryption</b>—If you choose this option, enter an SSL Certificate Authority (CA) certificate ARN to verify the server's certificate and make an encrypted connection.</p> <p>For on-premises Redis, DMS supports both public and private Certificate Authority (CA). For ElastiCache for Redis, DMS supports only a public CA.</p>
<b>Authentication type</b>	<p>Choose the type of authentication to perform while connecting to Redis. Options include, <b>None</b>, <b>Authentication role</b>, and <b>Authentication token</b>.</p> <p>If you choose Authentication role, provide an <b>Authentication username</b> and an <b>Authentication password</b>.</p> <p>If you choose Authentication token, provide an <b>Authentication password</b> only.</p>
<b>Replication instance</b>	[Optional] Only if you intend to test your connection, choose the name of the replication instance you previously entered on the <b>Create replication instance</b> page.

When you're finished providing all information for your endpoint, AWS DMS creates your Redis target endpoint for use during database migration.

For information about creating a migration task and starting your database migration, see [Creating a task \(p. 442\)](#).

## Specifying endpoint settings for Redis as a target

To create or modify a target endpoint, you can use the console or the `CreateEndpoint` or `ModifyEndpoint` API operations.

For a Redis target in the AWS DMS console, specify **Endpoint-specific settings** on the **Create endpoint** or **Modify endpoint** console page.

When using `CreateEndpoint` and `ModifyEndpoint` API operations, specify request parameters for the `RedisSettings` option. The example following shows how to do this using the AWS CLI.

```
aws dms create-endpoint --endpoint-identifier my-redis-target
--endpoint-type target --engine-name redis --redis-settings
'{"ServerName":"sample-test-sample.zz01zz.cluster.eee1.cache.bbbxxx.com","Port":6379,"AuthType":"auth-token",
"SslSecurityProtocol":"ssl-encryption", "AuthPassword":"notanactualpassword"}'

{
    "Endpoint": {
        "EndpointIdentifier": "my-redis-target",
        "EndpointType": "TARGET",
        "EngineName": "redis",
        "EngineDisplayName": "Redis",
        "TransferFiles": false,
        "ReceiveTransferredFiles": false,
        "Status": "active",
        "KmsKeyId": "arn:aws:kms:us-east-1:999999999999:key/x-b188188x",
        "EndpointArn": "arn:aws:dms:us-
east-1:555555555555:endpoint:ABCDEFHGIJKLMNOPQRSTUVWXYZ",
        "SslMode": "none",
        "RedisSettings": {
            "ServerName": "sample-test-sample.zz01zz.cluster.eee1.cache.bbbxxx.com",
            "Port": 6379,
            "SslSecurityProtocol": "ssl-encryption",
            "AuthType": "auth-token"
        }
    }
}
```

The `--redis-settings` parameters follow:

- `ServerName`—(Required) Of type `string`, specifies the Redis cluster that data will be migrated to, and is in your same VPC.
- `Port`—(Required) Of type `number`, the port value used to access the endpoint.
- `SslSecurityProtocol`—(Optional) Valid values include `plaintext` and `ssl-encryption`. The default is `ssl-encryption`.

The `plaintext` option doesn't provide Transport Layer Security (TLS) encryption for traffic between endpoint and database.

Use `ssl-encryption` to make an encrypted connection. `ssl-encryption` doesn't require an SSL Certificate Authority (CA) ARN to verify a server's certificate, but one can be identified optionally using the `SslCaCertificateArn` setting. If a certificate authority ARN isn't given, DMS uses the Amazon root CA.

When using an on-premises Redis target, you can use `SslCaCertificateArn` to import public or private Certificate Authority (CA) into DMS, and provide that ARN for server authentication. A private CA isn't supported when using ElastiCache for Redis as a target.

- AuthType—(Required) Indicates the type of authentication to perform when connecting to Redis. Valid values include none, auth-token, and auth-role.

The auth-token option requires an "*AuthPassword*" be provided, while the auth-role option requires "*AuthUserName*" and "*AuthPassword*" be provided.

## Using Babelfish as a target for AWS Database Migration Service

You can migrate data from a Microsoft SQL Server source database to a Babelfish target using AWS Database Migration Service.

Babelfish for Aurora PostgreSQL extends your Amazon Aurora PostgreSQL-Compatible Edition database with the ability to accept database connections from Microsoft SQL Server clients. Doing this allows applications originally built for SQL Server to work directly with Aurora PostgreSQL with few code changes compared to a traditional migration, and without changing database drivers.

The AWS DMS Babelfish endpoint supports Aurora PostgreSQL with Babelfish compatibility version 13.6 (Babelfish version 1.2.0) or later. Earlier versions of Babelfish on Aurora PostgreSQL version 13.4 and 13.5 require a minor version upgrade before using the Babelfish endpoint.

For information about using Babelfish as a database endpoint, see [Babelfish for Aurora PostgreSQL](#) in the *Amazon Aurora User Guide for Aurora*

## Prerequisites to using Babelfish as a target for AWS DMS

Perform the following preparatory steps prior to migrating data from your source to your target Babelfish database.

- Prepare and create your tables.
- Manage tables that have an IDENTITY column.
- Set minimum Aurora capacity units (ACUs) for your Aurora PostgreSQL Serverless V2 target database.

Procedures for performing those preparatory steps follow.

### To prepare and create your tables prior to migration

1. Run your create table DDL statements that include any unique constraints, primary keys, or default constraints.

Do not include foreign key constraints, or any DDL statements for objects like views, stored procedures, functions, or triggers. You can apply them after migrating your source database.

2. Identify any identity columns, computed columns, or columns containing rowversion or timestamp data types for your tables. Then, create the necessary transformation rules to handle known issues when running the migration task. For more information see, [Transformation rules and actions \(p. 500\)](#).
3. Identify columns with data types that Babelfish doesn't support. Then, change the affected columns in the target table to use supported data types, or create a transformation rule that removes them during the migration task. For more information see, [Transformation rules and actions \(p. 500\)](#).

The following table lists source data types not supported by Babelfish, and the corresponding recommended target data type to use.

Source data type	Recommended Babelfish data type
IMAGE	VARBINARY(MAX)
DATETIMEOFFSET(7)	DATETIMEOFFSET(6)  Babelfish supports values less than 7.
XML	When using AWS DMS to migrate XML data from a SQL Server source to a Babelfish target, the size of XML data can nearly double. So, we recommend you double the size typically set for the LobMaxSize option in the target metadata task settings. For more information, see <a href="#">Target metadata task settings (p. 452)</a> .
HEIRARCHYID	NVARCHAR(250)
GEOMETRY	VARCHAR(MAX)
GEOGRAPHY	VARCHAR(MAX)

### To set Aurora capacity units (ACUs) level for your Aurora PostgreSQL Serverless V2 source database

You can improve performance of your AWS DMS migration task prior to running it by setting the minimum ACU value.

- From the **Serverless v2 capacity settings** window, set **Minimum ACUs** to **2**, or a reasonable level for your Aurora DB cluster.

For additional information about setting Aurora capacity units, see [Choosing the Aurora Serverless v2 capacity range for an Aurora cluster](#) in the *Amazon Aurora User Guide*

After running your AWS DMS migration task, you can reset the minimum value of your ACUs to a reasonable level for your Aurora PostgreSQL Serverless V2 source database.

## Security requirements when using Babelfish as a target for AWS Database Migration Service

The following describes the security requirements for using AWS DMS with a Babelfish target:

- The administrator user name (the Admin user) used to create the database.
- T-SQL login and user with the sufficient SELECT, INSERT, UPDATE, DELETE, and REFERENCES permissions.

## User permissions for using Babelfish as a target for AWS DMS

### Important

For security purposes, the user account used for the data migration must be a registered user in any Babelfish database that you use as a target.

Your Babelfish target endpoint requires minimum user permissions to run an AWS DMS migration.

### To create a login and a low-privileged Transact-SQL (T-SQL) user

1. Create a login and password to use when connecting to the server.

```
CREATE LOGIN dms_user WITH PASSWORD = 'password';  
GO
```

2. Create the virtual database for your Babelfish cluster.

```
CREATE DATABASE my_database;  
GO
```

3. Create the T-SQL user for your target database.

```
USE my_database  
GO  
CREATE USER dms_user FOR LOGIN dms_user;  
GO
```

4. For each table in your Babelfish database, GRANT permissions to the tables.

```
GRANT SELECT, DELETE, INSERT, REFERENCES, UPDATE ON [dbo].[Categories] TO dms_user;
```

## Limitations on using Babelfish as a target for AWS Database Migration Service

The following limitations apply when using a Babelfish database as a target for AWS DMS:

- Enter TDS port number 1433 (Default) to use Babelfish as a target.
- Only Full Load Replication is supported. AWS DMS doesn't support CDC with Babelfish as a target.
- Only table preparation mode “**Do Nothing**” is supported.
- The ROWVERSION data type requires a table mapping rule that removes the column name from the table during the migration task.
- The sql\_variant data type isn't supported.
- Full LOB mode isn't supported.
- Replication task settings have the following limitations:

```
{  
    "FullLoadSettings": {  
        "TargetTablePrepMode": "DO_NOTHING",  
        "CreatePkAfterFullLoad": false,  
    },  
    "ValidationSettings": {  
        "EnableValidation": false,  
    }  
}
```

- TIME(7), DATETIME2(7), and DATETIMEOFFSET(7) data types in Babelfish limit the precision value for the seconds portion of the time to 6 digits. Consider using a precision value of 6 for your target table when using these data types. For Babelfish versions 2.2.0 and later, when using TIME(7) and DATETIME2(7), the seventh digit of precision is always zero.
- In DO\_NOTHING mode, DMS checks to see if the table already exists. If the table doesn't exist in the target schema, DMS creates the table based on the source table definition, and maps any user defined data types to their base data type.

- An AWS DMS migration task to a Babelfish target doesn't support tables that have columns using ROWVERSION or TIMESTAMP data types. You can use a table mapping rule that removes the column name from the table during the transfer process. In the following transformation rule example, a table named Actor in your source is transformed to remove all columns starting with the characters col from the Actor table in your target.

```
{  
  "rules": [ {  
    "rule-type": "selection",  
    "rule-id": "1",  
    "rule-name": "1",  
    "object-locator": {  
      "schema-name": "test",  
      "table-name": "%"  
    },  
    "rule-action": "include"  
  }, {  
    "rule-type": "transformation",  
    "rule-id": "2",  
    "rule-name": "2",  
    "rule-action": "remove-column",  
    "rule-target": "column",  
    "object-locator": {  
      "schema-name": "test",  
      "table-name": "Actor",  
      "column-name": "col%"  
    }  
  }]  
}
```

- Babelfish doesn't copy IMAGE data type columns.
- For tables with identity or computed columns, where the target tables use mixed case names like Categories, you must create a transformation rule action that converts the table names to lowercase for your DMS task. The following example shows how to create the transformation rule action, **Make lowercase** using the AWS DMS console. For more information, see [Transformation rules and actions \(p. 500\)](#).

The screenshot shows the 'Transformation rules' section of the AWS Database Migration Service. It includes a summary message, a rule definition, and configuration fields for a specific transformation rule.

**Transformation rules**

You can use transformation rules to change or transform schema, table or column names of some or all of the selected objects. [Info](#)

**Add transformation rule**

**where schema name is like 'dbo' and table name is like '%', convert-lowercase**

**Rule target**: Table

**Source name**: Enter a schema

**Source name**: Use the % character as a wildcard  
dbo

**Table name**: Use the % character as a wildcard  
%

**Action**: Make lowercase

- Prior to Babelfish version 2.2.0, DMS limits the number of columns that you can replicate to a Babelfish target endpoint to twenty (20) columns. With Babelfish version 2.2.0 and later, the limit increases to 100 columns. For tables with more than 100 columns, you can use the [PostgreSQL endpoint \(p. 304\)](#) as a workaround.

## Target data types for Babelfish

The following table shows the Babelfish target data types that are supported when using AWS DMS and the default mapping from AWS DMS data types.

For additional information about AWS DMS data types, see [Data types for AWS Database Migration Service \(p. 710\)](#).

AWS DMS data type	Babelfish data type
BOOLEAN	TINYINT
BYTES	VARBINARY(length)
DATE	DATE
TIME	TIME
INT1	SMALLINT

AWS DMS data type	Babelfish data type
INT2	SMALLINT
INT4	INT
INT8	BIGINT
NUMERIC	NUMERIC(p,s)
REAL4	REAL
REAL8	FLOAT
STRING	<p>If the column is a date or time column, then do the following:</p> <ul style="list-style-type: none"> <li>For SQL Server 2008 and later, use DATETIME2.</li> <li>For earlier versions, if the scale is 3 or less use DATETIME. In all other cases, use VARCHAR (37).</li> </ul> <p>If the column is not a date or time column, use VARCHAR (length).</p>
UINT1	TINYINT
UINT2	SMALLINT
UINT4	INT
UINT8	BIGINT
WSTRING	NVARCHAR(length)
BLOB	<p>VARBINARY(max)</p> <p>To use this data type with DMS, you must enable the use of BLOBs for a specific task. DMS supports BLOB data types only in tables that include a primary key.</p>
CLOB	<p>VARCHAR(max)</p> <p>To use this data type with DMS, you must enable the use of CLOBs for a specific task.</p>
NCLOB	<p>NVARCHAR(max)</p> <p>To use this data type with DMS, you must enable the use of NCLOBs for a specific task. During CDC, DMS supports NCLOB data types only in tables that include a primary key.</p>

# Configuring VPC endpoints as AWS DMS source and target endpoints

AWS DMS supports Amazon virtual private cloud (VPC) endpoints as sources and targets. AWS DMS can connect to any AWS source or target database with Amazon VPC endpoints as long as explicitly defined routes to these source and target databases are defined in their AWS DMS VPC.

By supporting Amazon VPC endpoints, AWS DMS makes it easier to maintain end-to-end network security for all replication tasks without additional networking configuration and setup. Using VPC endpoints for all source and target endpoints ensures that all your traffic remains within your VPC and under your control. Upgrades to AWS DMS versions 3.4.7 and higher require that you configure AWS DMS to use VPC endpoints or to use public routes to all your source and target endpoints that interact with the Amazon Web Services following:

- Amazon S3
- Amazon Kinesis
- AWS Secrets Manager
- Amazon DynamoDB
- Amazon Redshift
- Amazon OpenSearch Service

You might need VPC endpoints to support AWS DMS starting with version 3.4.7 as described following.

## Who is impacted when migrating to AWS DMS versions 3.4.7 and higher?

You are impacted if you are using one or more of the previously listed AWS DMS endpoints, and these endpoints are not publicly routable or they don't have VPC endpoints already associated with them.

## Who is not impacted when migrating to AWS DMS versions 3.4.7 and higher?

You are not impacted if:

- You aren't using one or more of the previously listed AWS DMS endpoints.
- You are using any of the previously listed endpoints and they are publically routable.
- You are using any of the previously listed endpoints and they have VPC endpoints associated with them.

## Preparing a migration to AWS DMS versions 3.4.7 and higher

To prevent AWS DMS task failures when you are using any of the endpoints described previously, take one of the steps following prior to upgrading AWS DMS to version 3.4.7 or higher:

- Make the impacted AWS DMS endpoints publicly routable. For example, add an Internet Gateway (IGW) route to any VPC already used by your AWS DMS replication instance to make all its source and target endpoints publicly routable.

- Create VPC endpoints to access all source and target endpoints used by AWS DMS as described following.

For any existing VPC endpoints that you use for your AWS DMS source and target endpoints, ensure that they use a trust policy that conforms with the XML policy document, `dms-vpc-role`. For more information on this XML policy document, see [Creating the IAM roles to use with the AWS CLI and AWS DMS API \(p. 632\)](#).

**Otherwise, to ensure that all source and target endpoints used by your AWS DMS replication instances are configured as VPC endpoints:**

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. On the VPC console menu bar, choose the same AWS Region as your AWS DMS replication instance.
3. In the VPC navigation pane, choose **Endpoints**.
4. In **Endpoints**, choose **Create endpoint**.
5. You can optionally specify a name tag. For example, **my-endpoint-DynamoDB-01**.
6. Under **Services** for S3 or DynamoDB only, choose a **Service Name** with its **Type** set to **Gateway**.
7. Under **VPC**, choose the same VPC as our AWS DMS replication instance to create the endpoint.
8. Under **Route Tables**, choose all available **Route Table ID** values.
9. To specify access control, under **Policy**, choose **Full access**. If you want to use a policy creation tool to specify your own access control, choose **Custom**. In any case, use a trust policy that conforms with the XML policy document, `dms-vpc-role`. For more information on this XML policy document, see [Creating the IAM roles to use with the AWS CLI and AWS DMS API \(p. 632\)](#).
10. Under **Endpoints**, verify that your newly created VPC endpoint **Status** is **Available**.

For more information on configuring VPC endpoints for an AWS DMS replication instance, see [Network configurations for database migration \(p. 112\)](#). For more information on creating interface VPC endpoints for accessing AWS services generally, see [Access an AWS service using an interface VPC endpoint](#) in the *AWS PrivateLink Guide*. For information on AWS DMS regional availability for VPC endpoints, see the [AWS Region Table](#).

## DDL statements supported by AWS DMS

You can execute data definition language (DDL) statements on the source database during the data migration process. These statements are replicated to the target database by the replication server.

Supported DDL statements include the following:

- Create table
- Drop table
- Rename table
- Truncate table
- Add column
- Drop column
- Rename column
- Change column data type

DMS doesn't capture all supported DDL statements for some source engine types. And DMS handles DDL statements differently when applying them to specific target engines. For information about which

DDL statements are supported for a specific source, and how they're applied to a target, see the specific documentation topic for that source and target endpoint.

You can use task settings to configure the way DMS handles DDL behavior during change data capture (CDC). For more information, see [Task settings for change processing DDL handling \(p. 468\)](#).

# Working with AWS DMS tasks

An AWS Database Migration Service (AWS DMS) task is where all the work happens. You specify what tables (or views) and schemas to use for your migration and any special processing, such as logging requirements, control table data, and error handling.

A task can consist of three major phases:

- Migration of existing data (Full load)
- The application of cached changes
- Ongoing replication (Change Data Capture)

For more information and an overview of how AWS DMS migration tasks migrate data, see [High-level view of AWS DMS \(p. 3\)](#)

When creating a migration task, you need to know several things:

- Before you can create a task, make sure that you create a source endpoint, a target endpoint, and a replication instance.
- You can specify many task settings to tailor your migration task. You can set these by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS DMS API. These settings include specifying how migration errors are handled, error logging, and control table information. For information about how to use a task configuration file to set task settings, see [Task settings example \(p. 450\)](#).
- After you create a task, you can run it immediately. The target tables with the necessary metadata definitions are automatically created and loaded, and you can specify ongoing replication.
- By default, AWS DMS starts your task as soon as you create it. However, in some situations, you might want to postpone the start of the task. For example, when using the AWS CLI, you might have a process that creates a task and a different process that starts the task based on some triggering event. As needed, you can postpone your task's start.
- You can monitor, stop, or restart tasks using the console, AWS CLI, or AWS DMS API. For information about stopping a task using the AWS DMS API, see [StopReplicationTask](#) in the [AWS DMS API Reference](#).

The following are actions that you can do when working with an AWS DMS task.

Task	Relevant documentation
<b>Creating a task</b>  When you create a task, you specify the source, target, and replication instance, along with any migration settings.	<a href="#">Creating a task (p. 442)</a>
<b>Creating an ongoing replication task</b>  You can set up a task to provide continuous replication between the source and target.	<a href="#">Creating tasks for ongoing replication using AWS DMS (p. 481)</a>
<b>Applying task settings</b>	<a href="#">Specifying task settings for AWS Database Migration Service tasks (p. 448)</a>

Task	Relevant documentation
<p>Each task has settings that you can configure according to the needs of your database migration. You create these settings in a JSON file or, with some settings, you can specify the settings using the AWS DMS console. For information about how to use a task configuration file to set task settings, see <a href="#">Task settings example (p. 450)</a>.</p>	
<p><b>Using table mapping</b></p> <p>Table mapping specifies additional task settings for tables using several types of rules. These rules allows you to specify the data source, source schema, tables and views, data, any table and data transformations that are to occur during the task, and settings for how these tables and columns are migrated from the source to the target.</p>	<p>Selection rules <a href="#">Selection rules and actions (p. 495)</a></p> <p>Transformation rules <a href="#">Transformation rules and actions (p. 500)</a></p> <p>Table-settings rules <a href="#">Table and collection settings rules and operations (p. 523)</a></p>
<p><b>Running premigration task assessments</b></p> <p>You can enable and run premigration task assessments showing issues with a supported source and target database that can cause problems during a migration. This can include issues such as unsupported data types, mismatched indexes and primary keys, and other conflicting task settings. These premigration assessments run before you run the task to identify potential issues before they occur during a migration.</p>	<p>Enabling and working with premigration assessments for a task (p. 548)</p>
<p><b>Data validation</b></p> <p>Data validation is a task setting you can use to have AWS DMS compare the data on your target data store with the data from your source data store.</p>	<p><a href="#">AWS DMS data validation (p. 584).</a></p>
<p><b>Modifying a task</b></p> <p>When a task is stopped, you can modify the settings for the task.</p>	<p><a href="#">Modifying a task (p. 487)</a></p>
<p><b>Moving a task</b></p> <p>When a task is stopped, you can move the task to a different replication instance.</p>	<p><a href="#">Moving a task (p. 487)</a></p>

Task	Relevant documentation
<b>Reloading tables during a task</b>  You can reload a table during a task if an error occurs during the task.	<a href="#">Reloading tables during a task (p. 488)</a>
<b>Applying filters</b>  You can use source filters to limit the number and type of records transferred from your source to your target. For example, you can specify that only employees with a location of headquarters are moved to the target database. You apply filters on a column of data.	<a href="#">Using source filters (p. 543)</a>
<b>Monitoring a task</b>  There are several ways to get information on the performance of a task and the tables used by the task.	<a href="#">Monitoring AWS DMS tasks (p. 558)</a>
<b>Managing task logs</b>  You can view and delete task logs using the AWS DMS API or AWS CLI.	<a href="#">Viewing and managing AWS DMS task logs (p. 567)</a>

### Topics

- [Creating a task \(p. 442\)](#)
- [Creating tasks for ongoing replication using AWS DMS \(p. 481\)](#)
- [Modifying a task \(p. 487\)](#)
- [Moving a task \(p. 487\)](#)
- [Reloading tables during a task \(p. 488\)](#)
- [Using table mapping to specify task settings \(p. 491\)](#)
- [Using source filters \(p. 543\)](#)
- [Enabling and working with premigration assessments for a task \(p. 548\)](#)
- [Specifying supplemental data for task settings \(p. 557\)](#)

## Creating a task

To create an AWS DMS migration task, you do the following:

- Create a source endpoint, a target endpoint, and a replication instance before you create a migration task.
- Choose a migration method:
  - **Migrating data to the target database** – This process creates files or tables in the target database and automatically defines the metadata that is required at the target. It also populates the tables with data from the source. The data from the tables is loaded in parallel for improved efficiency. This process is the **Migrate existing data** option in the AWS Management Console and is called Full Load in the API.

- **Capturing changes during migration** – This process captures changes to the source database that occur while the data is being migrated from the source to the target. When the migration of the originally requested data has completed, the change data capture (CDC) process then applies the captured changes to the target database. Changes are captured and applied as units of single committed transactions, and you can update several different target tables as a single source commit. This approach guarantees transactional integrity in the target database. This process is the **Migrate existing data and replicate ongoing changes** option in the console and is called full-load-and-cdc in the API.
- **Replicating only data changes on the source database** – This process reads the recovery log file of the source database management system (DBMS) and groups together the entries for each transaction. In some cases, AWS DMS can't apply changes to the target within a reasonable time (for example, if the target isn't accessible). In these cases, AWS DMS buffers the changes on the replication server for as long as necessary. It doesn't reread the source DBMS logs, which can take a large amount of time. This process is the **Replicate data changes only** option in the AWS DMS console.
- Determine how the task should handle large binary objects (LOBs) on the source. For more information, see [Setting LOB support for source databases in an AWS DMS task \(p. 479\)](#).
- Specify migration task settings. These include setting up logging, specifying what data is written to the migration control table, how errors are handled, and other settings. For more information about task settings, see [Specifying task settings for AWS Database Migration Service tasks \(p. 448\)](#).
- Set up table mapping to define rules to select and filter data that you are migrating. For more information about table mapping, see [Using table mapping to specify task settings \(p. 491\)](#). Before you specify your mapping, make sure that you review the documentation section on data type mapping for your source and your target database.
- Enable and run premigration task assessments before you run the task. For more information about premigration assessments, see [Enabling and working with premigration assessments for a task \(p. 548\)](#).
- Specify any required supplemental data for the task to migrate your data. For more information, see [Specifying supplemental data for task settings \(p. 557\)](#).

You can choose to start a task as soon as you finish specifying information for that task on the **Create task** page. Alternatively, you can start the task from the Dashboard page later as well.

The following procedure assumes that you have already specified replication instance information and endpoints. For more information about setting up endpoints, see [Creating source and target endpoints \(p. 131\)](#).

### To create a migration task

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.

If you are signed in as an AWS Identity and Access Management (IAM) user, make sure that you have the appropriate permissions to access AWS DMS. For more information about the permissions required, see [IAM permissions needed to use AWS DMS \(p. 628\)](#).
2. On the navigation pane, choose **Database migration tasks**, and then choose **Create task**.
3. On the **Create database migration task** page, in the **Task configuration** section, specify the task options. The following table describes the settings.

# Create database migration task

## Task configuration

### Task identifier

*Type a unique identifier for the task*

### Descriptive Amazon Resource Name (ARN) - optional

A friendly name to override the default DMS ARN. You cannot modify this ARN after you create the task.

*Friendly-ARN-name*

### Replication instance

*Choose a replication instance*

### Source database endpoint

*Choose a source database endpoint*

### Target database endpoint

*Choose a target database endpoint*

### Migration type [Info](#)

[Migrate existing data](#)

For this option	Do this
<b>Task identifier</b>	Enter a name for the task.
<b>Descriptive Amazon Resource Name (ARN) - optional</b>	A friendly name to override the default AWS DMS ARN. You can't change this name after you create the task.
<b>Replication instance</b>	Shows the replication instance to be used.
<b>Source database endpoint</b>	Shows the source endpoint to be used.
<b>Target database endpoint</b>	Shows the target endpoint to be used.
<b>Migration type</b>	Choose the migration method you want to use. You can choose to have just the existing data migrated to the target database or have ongoing changes sent to the target database in addition to the migrated data.

4. In the **Task Settings** section, specify values for editing your task, target table preparation mode, stop task, LOB settings, validation, and logging.

For this option	Do this
<b>Editing mode</b>	Choose whether to use the Wizard or the JSON editor to specify your task settings. If you choose Wizard, the following options will be displayed.
<b>CDC start mode for source transactions</b>	<p>This setting is only visible if you choose <b>Replicate data changes only</b> for <b>Migration type</b> in the preceding section.</p> <p><b>Disable custom CDC start mode</b> – If you choose this option, you can start your task either automatically by using the <b>Automatically on create</b> option following, or manually by using the console.</p> <p><b>Enable custom CDC start mode</b> – If you choose this option, you can specify a custom UTC start time to start processing changes.</p>
<b>Target table preparation mode</b>	<p>This setting is only visible if you choose <b>Migrate existing data</b> or <b>Migrate existing data and replicate ongoing changes</b> for <b>Migration type</b> in the preceding section.</p> <p><b>Do nothing</b> – In <b>Do nothing</b> mode, AWS DMS assumes that the target tables have been precreated on the target. If the migration is a full load or full load plus CDC, make sure that the target tables are empty before starting the migration. If the target table doesn't exist, DMS creates the table for you. Your table structure remains as is and any existing data is left in the table. <b>Do nothing</b> mode is appropriate for CDC-only tasks when the target tables have been backfilled from the source and ongoing replication is applied to keep the source and target in sync. To precreate tables, you can use the AWS Schema Conversion Tool (AWS SCT). For more information, see <a href="#">Installing AWS SCT</a>.</p>

For this option	Do this
	<p><b>Drop tables on target</b> – In <b>Drop tables on target</b> mode, AWS DMS drops the target tables and recreates them before starting the migration. This approach ensures that the target tables are empty when the migration starts. AWS DMS creates only the objects required to efficiently migrate the data: tables, primary keys, and in some cases, unique indexes. AWS DMS doesn't create secondary indexes, nonprimary key constraints, or column data defaults. If you are performing a full load plus CDC or CDC-only task, we recommend that you pause the migration at this point. Then, create secondary indexes that support filtering for update and delete statements.</p> <p>You might need to perform some configuration on the target database when you use <b>Drop tables on target</b> mode. For example, for an Oracle target, AWS DMS can't create a schema (database user) for security reasons. In this case, you precreate the schema user so AWS DMS can create the tables when the migration starts. For most other target types, AWS DMS creates the schema and all associated tables with the proper configuration parameters.</p> <p><b>Truncate</b> – In <b>Truncate</b> mode, AWS DMS truncates all target tables before the migration starts. If the target table doesn't exist, DMS creates the table for you. Your table structure remains as is but tables are truncated at the target. <b>Truncate</b> mode is appropriate for full load or full load plus CDC migrations where the target schema has been precreated before the migration starts. To precreate tables, you can use AWS SCT. For more information, see <a href="#">Installing AWS SCT</a>.</p> <p><b>Note</b>  If your source is MongoDB, <b>Truncate</b> mode doesn't truncate tables at the target. Instead, it drops the collection and loses all the indices. Avoid <b>Truncate</b> mode when your source is MongoDB.</p>

For this option	Do this
<b>Stop task after full load completes</b>	<p>This setting is only visible if you choose <b>Migrate existing data and replicate ongoing changes</b> for <b>Migration type</b> in the preceding section.</p> <p><b>Don't stop</b> – Don't stop the task but immediately apply cached changes and continue on.</p> <p><b>Stop before applying cached changes</b> – Stop the task before the application of cached changes. Using this approach, you can add secondary indexes that might speed the application of changes.</p> <p><b>Stop after applying cached changes</b> – Stop the task after cached changes have been applied. Using this approach, you can add foreign keys if you are using transactional apply.</p>
<b>Include LOB columns in replication</b>	<p><b>Don't include LOB columns</b> – LOB columns are excluded from the migration.</p> <p><b>Full LOB mode</b> – Migrate complete LOBs regardless of size. AWS DMS migrates LOBs piecewise in chunks controlled by the <b>Max LOB size</b> parameter. This mode is slower than using Limited LOB mode.</p> <p><b>Limited LOB mode</b> – Truncate LOBs to the value of the <b>Max LOB size</b> parameter. This mode is faster than using Full LOB mode.</p>
<b>Maximum LOB size (kb)</b>	In <b>Limited LOB Mode</b> , LOB columns that exceed the setting of <b>Max LOB size</b> are truncated to the specified <b>Max LOB Size</b> value.
<b>Enable validation</b>	Enables data validation, to verify that the data is migrated accurately from the source to the target. For more information, see <a href="#">AWS DMS data validation (p. 584)</a> .
<b>Enable CloudWatch logs</b>	Enables logging by Amazon CloudWatch.

5. In the **Premigration assessment** section, choose whether to run a premigration assessment. A premigration assessment warns you of potential migration issues before starting your database migration task. For more information, see [Enabling and working with premigration assessments \(p. 548\)](#).
6. In the **Migration task startup configuration** section, specify whether to start the task automatically after creation.
7. In the **Tags** section, specify any tags you need to organize your task. You can use tags to manage your IAM roles and policies, and track your DMS costs. For more information, see [Tagging resources \(p. 591\)](#).
8. After you have finished with the task settings, choose **Create task**.

# Specifying task settings for AWS Database Migration Service tasks

Each task has settings that you can configure according to the needs of your database migration. You create these settings in a JSON file or, with some settings, you can specify the settings using the AWS DMS console. For information about how to use a task configuration file to set task settings, see [Task settings example \(p. 450\)](#).

There are several main types of task settings, as listed following.

## Topics

- [Task settings example \(p. 450\)](#)
- [Target metadata task settings \(p. 452\)](#)
- [Full-load task settings \(p. 454\)](#)
- [Time Travel task settings \(p. 455\)](#)
- [Logging task settings \(p. 460\)](#)
- [Control table task settings \(p. 461\)](#)
- [Stream buffer task settings \(p. 465\)](#)
- [Change processing tuning settings \(p. 465\)](#)
- [Data validation task settings \(p. 467\)](#)
- [Task settings for change processing DDL handling \(p. 468\)](#)
- [Character substitution task settings \(p. 469\)](#)
- [Before image task settings \(p. 473\)](#)
- [Error handling task settings \(p. 474\)](#)
- [Saving task settings \(p. 476\)](#)

Task settings	Relevant documentation
<b>Creating a task assessment report</b>  You can create a task assessment report that shows any unsupported data types that could cause problems during migration. You can run this report on your task before running the task to find out potential issues.	<a href="#">Enabling and working with premigration assessments for a task (p. 548)</a>
<b>Creating a task</b>  When you create a task, you specify the source, target, and replication instance, along with any migration settings.	<a href="#">Creating a task (p. 442)</a>
<b>Creating an ongoing replication task</b>  You can set up a task to provide continuous replication between the source and target.	<a href="#">Creating tasks for ongoing replication using AWS DMS (p. 481)</a>
<b>Applying task settings</b>	<a href="#">Specifying task settings for AWS Database Migration Service tasks (p. 448)</a>

Task settings	Relevant documentation
Each task has settings that you can configure according to the needs of your database migration. You create these settings in a JSON file or, with some settings, you can specify the settings using the AWS DMS console.	
<b>Data validation</b>  Use data validation to have AWS DMS compare the data on your target data store with the data from your source data store.	<a href="#">AWS DMS data validation (p. 584)</a>
<b>Modifying a task</b>  When a task is stopped, you can modify the settings for the task.	<a href="#">Modifying a task (p. 487)</a>
<b>Reloading tables during a task</b>  You can reload a table during a task if an error occurs during the task.	<a href="#">Reloading tables during a task (p. 488)</a>
<b>Using table mapping</b>  Table mapping uses several types of rules to specify task settings for the data source, source schema, data, and any transformations that should occur during the task.	Selection Rules <a href="#">Selection rules and actions (p. 495)</a>  Transformation Rules <a href="#">Transformation rules and actions (p. 500)</a>
<b>Applying filters</b>  You can use source filters to limit the number and type of records transferred from your source to your target. For example, you can specify that only employees with a location of headquarters are moved to the target database. You apply filters on a column of data.	<a href="#">Using source filters (p. 543)</a>
<b>Monitoring a task</b>  There are several ways to get information on the performance of a task and the tables used by the task.	<a href="#">Monitoring AWS DMS tasks (p. 558)</a>
<b>Managing task logs</b>  You can view and delete task logs using the AWS DMS API or AWS CLI.	<a href="#">Viewing and managing AWS DMS task logs (p. 567)</a>

## Task settings example

You can use either the AWS Management Console or the AWS CLI to create a replication task. If you use the AWS CLI, you set task settings by creating a JSON file and by providing the file as the [ReplicationTaskSettings](#) parameter of the [CreateReplicationTask](#) operation.

The following example shows how to use the AWS CLI to call the [CreateReplicationTask](#) operation:

```
aws dms create-replication-task \
--replication-task-identifier MyTask \
--source-endpoint-arn arn:aws:dms:us-
west-2:123456789012:endpoint:ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890ABC \
--target-endpoint-arn arn:aws:dms:us-
west-2:123456789012:endpoint:ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890ABC \
--replication-instance-arn arn:aws:dms:us-
west-2:123456789012:rep:ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890ABC \
--migration-type cdc \
--table-mappings file://tablemappings.json \
--replication-task-settings file://settings.json
```

The preceding example uses a table mapping file called `tablemappings.json`. For table mapping examples, see [Using table mapping to specify task settings \(p. 491\)](#).

A task settings JSON file can look like the following.

```
{
    "TargetMetadata": {
        "TargetSchema": "",
        "SupportLobs": true,
        "FullLobMode": false,
        "LobChunkSize": 64,
        "LimitedSizeLobMode": true,
        "LobMaxSize": 32,
        "InlineLobMaxSize": 0,
        "LoadMaxFileSize": 0,
        "ParallelLoadThreads": 0,
        "ParallelLoadBufferSize": 0,
        "ParallelLoadQueuesPerThread": 1,
        "ParallelApplyThreads": 0,
        "ParallelApplyBufferSize": 100,
        "ParallelApplyQueuesPerThread": 1,
        "BatchApplyEnabled": false,
        "TaskRecoveryTableEnabled": false
    },
    "FullLoadSettings": {
        "TargetTablePrepMode": "DO_NOTHING",
        "CreatePkAfterFullLoad": false,
        "StopTaskCachedChangesApplied": false,
        "StopTaskCachedChangesNotApplied": false,
        "MaxFullLoadSubTasks": 8,
        "TransactionConsistencyTimeout": 600,
        "CommitRate": 10000
    },
    "TTSettings" : {
        "EnableTT" : true,
        "TTS3Settings": {
            "EncryptionMode": "SSE_KMS",
            "ServerSideEncryptionKmsKeyId": "arn:aws:kms:us-west-2:112233445566:key/myKMSKey",
            "ServiceAccessRoleArn": "arn:aws:iam::112233445566:role/dms-tt-s3-access-role",
            "BucketName": "myttbucket",
            "BucketFolder": "myttfolder",
        }
    }
}
```

```

        "EnableDeletingFromS3OnTaskDelete": false
    },
    "TTRRecordSettings": {
        "EnableRawData" : true,
        "OperationsToLog": "DELETE,UPDATE",
        "MaxRecordSize": 64
    }
},
"Logging": {
    "EnableLogging": false
},
"ControlTablesSettings": {
    "ControlSchema":"",
    "HistoryTimeslotInMinutes":5,
    "HistoryTableEnabled": false,
    "SuspendedTablesTableEnabled": false,
    "StatusTableEnabled": false
},
"StreamBufferSettings": {
    "StreamBufferCount": 3,
    "StreamBufferSizeInMB": 8
},
"ChangeProcessingTuning": {
    "BatchApplyPreserveTransaction": true,
    "BatchApplyTimeoutMin": 1,
    "BatchApplyTimeoutMax": 30,
    "BatchApplyMemoryLimit": 500,
    "BatchSplitSize": 0,
    "MinTransactionSize": 1000,
    "CommitTimeout": 1,
    "MemoryLimitTotal": 1024,
    "MemoryKeepTime": 60,
    "StatementCacheSize": 50
},
"ChangeProcessingDdlHandlingPolicy": {
    "HandleSourceTableDropped": true,
    "HandleSourceTableTruncated": true,
    "HandleSourceTableAltered": true
},
"LoopbackPreventionSettings": {
    "EnableLoopbackPrevention": true,
    "SourceSchema": "LOOP-DATA",
    "TargetSchema": "loop-data"
},
"CharacterSetSettings": {
    "CharacterReplacements": [ {
        "SourceCharacterCodePoint": 35,
        "TargetCharacterCodePoint": 52
    }, {
        "SourceCharacterCodePoint": 37,
        "TargetCharacterCodePoint": 103
    }
],
    "CharacterSetSupport": {
        "CharacterSet": "UTF16_PlatformEndian",
        "ReplaceWithCharacterCodePoint": 0
    }
},
"BeforeImageSettings": {
    "EnableBeforeImage": false,
    "FieldName": "",
    "ColumnFilter": "pk-only"
},
"ErrorBehavior": {
    "DataErrorPolicy": "LOG_ERROR",

```

```

    "DataTruncationErrorPolicy": "LOG_ERROR",
    "DataErrorEscalationPolicy": "SUSPEND_TABLE",
    "DataErrorEscalationCount": 50,
    "TableErrorPolicy": "SUSPEND_TABLE",
    "TableErrorEscalationPolicy": "STOP_TASK",
    "TableErrorEscalationCount": 50,
    "RecoverableErrorCount": 0,
    "RecoverableErrorInterval": 5,
    "RecoverableErrorThrottling": true,
    "RecoverableErrorThrottlingMax": 1800,
    "ApplyErrorDeletePolicy": "IGNORE_RECORD",
    "ApplyErrorInsertPolicy": "LOG_ERROR",
    "ApplyErrorUpdatePolicy": "LOG_ERROR",
    "ApplyErrorEscalationPolicy": "LOG_ERROR",
    "ApplyErrorEscalationCount": 0,
    "FullLoadIgnoreConflicts": true
},
"ValidationSettings": {
    "EnableValidation": false,
    "ValidationMode": "ROW_LEVEL",
    "ThreadCount": 5,
    "PartitionSize": 10000,
    "FailureMaxCount": 1000,
    "RecordFailureDelayInMinutes": 5,
    "RecordSuspendDelayInMinutes": 30,
    "MaxKeyColumnSize": 8096,
    "TableFailureMaxCount": 10000,
    "ValidationOnly": false,
    "HandleCollationDiff": false,
    "RecordFailureDelayLimitInMinutes": 1,
    "SkipLobColumns": false,
    "ValidationPartialLobSize": 0,
    "ValidationQueryCdcDelaySeconds": 0
}
}
}

```

## Target metadata task settings

Target metadata settings include the following. For information about how to use a task configuration file to set task settings, see [Task settings example \(p. 450\)](#).

- **TargetSchema** – The target table schema name. If this metadata option is empty, the schema from the source table is used. AWS DMS automatically adds the owner prefix for the target database to all tables if no source schema is defined. This option should be left empty for MySQL-type target endpoints.
- **LOB settings** – Settings that determine how large objects (LOBs) are managed. If you set `SupportLobs=true`, you must set one of the following to `true`:
  - **FullLobMode** – If you set this option to `true`, then you must enter a value for the `LobChunkSize` option. Enter the size, in kilobytes, of the LOB chunks to use when replicating the data to the target. The `FullLobMode` option works best for very large LOB sizes but tends to cause slower loading. The recommended value for `LobChunkSize` is 64 kilobytes. Increasing the value for `LobChunkSize` above 64 kilobytes can cause task failures.
  - **InlineLobMaxSize** – This value determines which LOBs AWS DMS transfers inline during a full load. Transferring small LOBs is more efficient than looking them up from a source table. During a full load, AWS DMS checks all LOBs and performs an inline transfer for the LOBs that are smaller than `InlineLobMaxSize`. AWS DMS transfers all LOBs larger than the `InlineLobMaxSize` in `FullLobMode`. The default value for `InlineLobMaxSize` is 0 and the range is 1–102400 kilobytes (100 MB). Set a value for `InlineLobMaxSize` only if you know that most of the LOBs are smaller than the value specified in `InlineLobMaxSize`.

- **LimitedSizeLobMode** – If you set this option to `true`, then you must enter a value for the `LobMaxSize` option. Enter the maximum size, in kilobytes, for an individual LOB. The maximum recommended value for `LobMaxSize` is 102400 kilobytes (100 MB).

For more information about the criteria for using these task LOB settings, see [Setting LOB support for source databases in an AWS DMS task \(p. 479\)](#). You can also control the management of LOBs for individual tables. For more information, see [Table and collection settings rules and operations \(p. 523\)](#).

- **LoadMaxFileSize** – An option for CSV-based target endpoints like MySQL, PostgreSQL, and Amazon Redshift that support use of comma-separated value (.csv) files for loading data. `LoadMaxFileSize` defines the maximum size on disk of stored, unloaded data, such as .csv files. This option overrides the target endpoint connection attribute, `maxFileSize`. You can provide values from 0, which indicates that this option doesn't override the connection attribute, to 100,000 KB.
- **BatchApplyEnabled** – Determines if each transaction is applied individually or if changes are committed in batches. The default value is `false`.

When `BatchApplyEnabled` is set to `true`, DMS requires a Primary Key (PK) or Unique Key (UK) on the **source** table(s). Without a PK or UK on source tables, only batch inserts are applied but not batch updates and deletes.

When `BatchApplyEnabled` is set to `true`, AWS DMS generates an error message if a **target** table has a unique constraint and a primary key. Target tables with both a unique constraint and primary key aren't supported when `BatchApplyEnabled` is set to `true`.

When `BatchApplyEnabled` is set to `true` and AWS DMS encounters a data error from a table with the default error-handling policy, the AWS DMS task switches from batch mode to one-by-one mode for the rest of the tables. To alter this behavior, you can set the "SUSPEND\_TABLE" action on the following policies in the "ErrorBehavior" group property of the task settings JSON file:

- `DataErrorPolicy`
- `ApplyErrorDeletePolicy`
- `ApplyErrorInsertPolicy`
- `ApplyErrorUpdatePolicy`

For more information on this "ErrorBehavior" group property, see the example task settings JSON file in [Specifying task settings for AWS Database Migration Service tasks \(p. 448\)](#). After setting these policies to "SUSPEND\_TABLE", the AWS DMS task then suspends data errors on any tables that raise them and continues in batch mode for all tables.

You can use the `BatchApplyEnabled` parameter with the `BatchApplyPreserveTransaction` parameter. If `BatchApplyEnabled` is set to `true`, then the `BatchApplyPreserveTransaction` parameter determines the transactional integrity.

If `BatchApplyPreserveTransaction` is set to `true`, then transactional integrity is preserved and a batch is guaranteed to contain all the changes within a transaction from the source.

If `BatchApplyPreserveTransaction` is set to `false`, then there can be temporary lapses in transactional integrity to improve performance.

The `BatchApplyPreserveTransaction` parameter applies only to Oracle target endpoints, and is only relevant when the `BatchApplyEnabled` parameter is set to `true`.

When LOB columns are included in the replication, you can use `BatchApplyEnabled` only in limited LOB mode.

For more information about using these settings for a change data capture (CDC) load, see [Change processing tuning settings \(p. 465\)](#).

- **ParallelLoadThreads** – Specifies the number of threads that AWS DMS uses to load each table into the target database. This parameter has maximum values for non-RDBMS targets. The maximum value for a DynamoDB target is 200. The maximum value for an Amazon Kinesis Data Streams, Apache Kafka, or Amazon OpenSearch Service target is 32. You can ask to have this maximum limit increased. For information on the settings for parallel load of individual tables, see [Table and collection settings rules and operations \(p. 523\)](#).

This setting applies to the following endpoint engine types:

- DynamoDB
- Amazon Kinesis Data Streams
- Amazon MSK
- Amazon OpenSearch Service
- Amazon Redshift
- **ParallelLoadBufferSize** – Specifies the maximum number of records to store in the buffer that the parallel load threads use to load data to the target. The default value is 50. The maximum value is 1,000. This setting is currently only valid when DynamoDB, Kinesis, Apache Kafka, or OpenSearch is the target. Use this parameter with ParallelLoadThreads. ParallelLoadBufferSize is valid only when there is more than one thread. For information on the settings for parallel load of individual tables, see [Table and collection settings rules and operations \(p. 523\)](#).
- **ParallelLoadQueuesPerThread** – Specifies the number of queues that each concurrent thread accesses to take data records out of queues and generate a batch load for the target. The default is 1. This setting is currently only valid when Kinesis or Apache Kafka is the target.
- **ParallelApplyThreads** – Specifies the number of concurrent threads that AWS DMS uses during a CDC load to push data records to a Kinesis, Apache Kafka, OpenSearch, or Amazon Redshift target endpoint. The default is zero (0).

This setting applies to the following endpoint engine types:

- Amazon Kinesis Data Streams
- Amazon MSK
- Amazon OpenSearch Service
- Amazon Redshift
- **ParallelApplyBufferSize** – Specifies the maximum number of records to store in each buffer queue for concurrent threads to push to a Kinesis, Apache Kafka, OpenSearch, or Amazon Redshift target endpoint during a CDC load. The default value is 100. Use this option when ParallelApplyThreads specifies more than one thread.
- **ParallelApplyQueuesPerThread** – Specifies the number of queues that each thread accesses to take data records out of queues and generate a batch load for a Kinesis, Apache Kafka, or OpenSearch endpoint during CDC. The default value is 1.

## Full-load task settings

Full-load settings include the following. For information about how to use a task configuration file to set task settings, see [Task settings example \(p. 450\)](#).

- To indicate how to handle loading the target at full-load startup, specify one of the following values for the TargetTablePrepMode option:
  - DO\_NOTHING – Data and metadata of the existing target table aren't affected.
  - DROP\_AND\_CREATE – The existing table is dropped and a new table is created in its place.
  - TRUNCATE\_BEFORE\_LOAD – Data is truncated without affecting the table metadata.
- To delay primary key or unique index creation until after a full load completes, set the CreatePkAfterFullLoad option to true.

- For full-load and CDC-enabled tasks, you can set the following options for Stop task after full load completes:
  - StopTaskCachedChangesApplied – Set this option to true to stop a task after a full load completes and cached changes are applied.
  - StopTaskCachedChangesNotApplied – Set this option to true to stop a task before cached changes are applied.
- To indicate the maximum number of tables to load in parallel, set the MaxFullLoadSubTasks option. The default is 8; the maximum value is 49.
- You can set the number of seconds that AWS DMS waits for transactions to close before beginning a full-load operation. To do so, if transactions are open when the task starts set the TransactionConsistencyTimeout option. The default value is 600 (10 minutes). AWS DMS begins the full load after the timeout value is reached, even if there are open transactions. A full-load-only task doesn't wait for 10 minutes but instead starts immediately.
- To indicate the maximum number of records that can be transferred together, set the CommitRate option. The default value is 10000, and the maximum value is 50000.

## Time Travel task settings

To log and debug replication tasks, you can use AWS DMS Time Travel. In this approach, you use Amazon S3 to store logs and encrypt them using your encryption keys. Only with access to your Time Travel S3 bucket, can you retrieve your S3 logs using date-time filters, then view, download, and obfuscate logs as needed. By doing this, you can securely "travel back in time" to investigate database activities. Time Travel works independently from the CloudWatch logging. For more information on CloudWatch logging, see [Logging task settings \(p. 460\)](#).

You can use Time Travel in all AWS Regions with DMS-supported PostgreSQL source endpoints and DMS-supported PostgreSQL and MySQL target endpoints. You can turn on Time Travel only for full-load and CDC tasks and for CDC only tasks. To turn on Time Travel or to modify any existing Time Travel settings, ensure that your replication task is stopped.

The Time Travel settings include the TTSettings properties following:

- EnableTT – If this option is set to true, Time Travel logging is turned on for the task. The default value is false.

Type: Boolean

Required: No

- EncryptionMode – The type of server-side encryption being used on your S3 bucket to store your data and logs. You can specify either "SSE\_S3" (the default) or "SSE\_KMS".

You can change EncryptionMode from "SSE\_KMS" to "SSE\_S3", but not the reverse.

Type: String

Required: No

- ServerSideEncryptionKmsKeyId – If you specify "SSE\_KMS" for EncryptionMode, provide the ID for your custom managed AWS KMS key. Make sure that the key that you use has an attached policy that turns on AWS Identity and Access Management (IAM) user permissions and allows use of the key.

Only your own custom-managed symmetric KMS key is supported with the "SSE\_KMS" option.

Type: String

Required: Only if you set EncryptionMode to "SSE\_KMS"

- **ServiceAccessRoleArn** – The Amazon Resource Name (ARN) used by the service to access the IAM role. Set the role name to `dms-tt-s3-access-role`. This is a required setting that allows AWS DMS to write and read objects from an S3 bucket.

Type: String

Required: If Time Travel is turned on

Following is an example policy for this role.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "kms:GenerateDataKey",
                "kms:Decrypt",
                "s3>ListBucket",
                "s3>DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::S3bucketName*",
                "arn:aws:kms:us-east-1:112233445566:key/1234a1a1-1m2m-1z2z-
d1d2-12dmstt1234"
            ]
        }
    ]
}
```

Following is an example trust policy for this role.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "dms.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

- **BucketName** – The name of the S3 bucket to store Time Travel logs. Make sure to create this S3 bucket before turning on Time Travel logs.

Type: String

Required: If Time Travel is turned on

- **BucketFolder** – An optional parameter to set a folder name in the S3 bucket. If you specify this parameter, DMS creates the Time Travel logs in the path `"/BucketName/BucketFolder/taskARN/YYYY/MM/DD/hh"`. If you don't specify this parameter, AWS DMS creates the default path as `"/BucketName/dms-time-travel-logs/taskARN/YYYY/MM/DD/hh"`.

Type: String

Required: No

- **EnableDeletingFromS3OnTaskDelete** – When this option is set to `true`, AWS DMS deletes the Time Travel logs from S3 if the task is deleted. The default value is `false`.

Type: String

Required: No

- **EnableRawData** – When this option is set to `true`, the data manipulation language (DML) raw data for Time Travel logs appears under the `raw_data` column of the Time Travel logs. For the details, see [Using the Time Travel logs \(p. 459\)](#). The default value is `false`. When this option is set to `false`, only the type of DML is captured.

Type: String

Required: No

- **OperationsToLog** – Specifies the type of DML operations to log in Time Travel logs. You can specify one of the following:

- "INSERT"
- "UPDATE"
- "DELETE"
- "COMMIT"
- "ROLLBACK"
- "ALL"

The default is "ALL".

Type: String

Required: No

- **MaxRecordSize** – Specifies the maximum size of Time Travel log records that are logged for each row. Use this parameter to control the growth of Time Travel logs for especially busy tables. The default is 64 KB.

Type: Integer

Required: No

For more information on turning on and using Time Travel logs, see the following topics.

#### Topics

- [Turning on the Time Travel logs for a task \(p. 457\)](#)
- [Using the Time Travel logs \(p. 459\)](#)
- [How often AWS DMS uploads Time Travel logs to S3 \(p. 459\)](#)

### Turning on the Time Travel logs for a task

You can turn on Time Travel for an AWS DMS task using the task settings described previously. Make sure that your replication task is stopped before you turn on Time Travel.

## To turn on Time Travel using the AWS CLI

1. Create a DMS task configuration JSON file and add a TTSettings section such as the following. For information about how to use a task configuration file to set task settings, see [Task settings example \(p. 450\)](#).

```

.
.
.

    },
"TTSettings" : {
    "EnableTT" : true,
    "TTS3Settings": {
        "EncryptionMode": "SSE_KMS",
        "ServerSideEncryptionKmsKeyId": "arn:aws:kms:us-west-2:112233445566:key/
myKMSKey",
        "ServiceAccessRoleArn": "arn:aws:iam::112233445566:role/dms-tt-s3-access-role",
        "BucketName": "myttbucket",
        "BucketFolder": "myttfolder",
        "EnableDeletingFromS3OnTaskDelete": false
    },
    "TTRecordSettings": {
        "EnableRawData" : true,
        "OperationsToLog": "DELETE,UPDATE",
        "MaxRecordSize": 64
    },
.
.
.
```

2. In an appropriate task action, specify this JSON file using the `--replication-task-settings` option. For example, the CLI code fragment following specifies this Time Travel settings file as part of `create-replication-task`.

```

aws dms create-replication-task
--target-endpoint-arn arn:aws:dms:us-
east-1:112233445566:endpoint:ELS507YTYV452CAZR2EYBNQGILFHQIFVPWFRQAY \
--source-endpoint-arn arn:aws:dms:us-
east-1:112233445566:endpoint:HNX2BWIIN5ZYFF7F6UFFZVWTDFFSMTNOV2FTXZA \
--replication-instance-arn arn:aws:dms:us-
east-1:112233445566:rep:ERLHG2UA52EEJJKFYNWRPCG6T7EPUAB5AWBUJQ \
--migration-type full-load-and-cdc --table-mappings 'file:///FilePath/mappings.json' \
--replication-task-settings 'file:///FilePath/task-settings-tt-enabled.json' \
--replication-task-identifier test-task
.
.
```

Here, the name of this Time Travel settings file is `task-settings-tt-enabled.json`.

Similarly, you can specify this file as part of the `modify-replication-task` action.

Note the special handling of Time Travel logs for the task actions following:

- `start-replication-task` – When you run a replication task, if an S3 bucket used for Time Travel isn't accessible, the task is marked as FAILED.
- `stop-replication-task` – When the task stops, AWS DMS immediately pushes all Time Travel logs that are currently available for the replication instance to the S3 bucket used for Time Travel.

While a replication task runs, you can change the EncryptionMode value from "SSE\_KMS" to "SSE\_S3" but not the reverse.

If the size of Time Travel logs for an ongoing task exceeds 1 GB, DMS pushes the logs to S3 within five minutes of reaching that size. After a task is running, if the S3 bucket or KMS key becomes inaccessible, DMS stops pushing logs to this bucket. If you find your logs aren't being pushed to your S3 bucket, check your S3 and AWS KMS permissions. For more details on how often DMS pushes these logs to S3, see [How often AWS DMS uploads Time Travel logs to S3 \(p. 459\)](#).

To turn on Time Travel for an existing task from the console, use the JSON editor option under **Task Settings** to add a TTSettings section.

## Using the Time Travel logs

*Time Travel log files* are comma-separated value (CSV) files with the fields following.

```
log_timestamp
component
dms_source_code_location
transaction_id
event_id
event_timestamp
lsn/scn
primary_key
record_type
event_type
schema_name
table_name
statement
action
result
raw_data
```

After your Time Travel logs are available in S3, you can directly access and query them with tools such as Amazon Athena. Or you can download the logs as you can any file from S3.

The example following shows a Time Travel log where transactions for a table called `mytable` are logged. The line endings for the following log are added for readability.

```
"log_timestamp ","tt_record_type","dms_source_code_location ","transaction_id",
"event_id","event_timestamp","scn_lsn","primary_key","record_type","event_type",
"schema_name","table_name","statement","action","result","raw_data"
"2021-09-23T01:03:00:778230","SOURCE_CAPTURE","postgres_endpoint_wal_engine.c:00819",
"609284109","565612992","2021-09-23 01:03:00.765321+00","00000E9C/D53AB518","","","DML",
"UPDATE (3)","dmstest","mytable","","Migrate","","table dmstest.mytable:
UPDATE: id[bigint]:2244937 phone_number[character varying]:'phone-number-482'
age[integer]:82 gender[character]:'f' isactive[character]:'true '
date_of_travel[timestamp without time zone]:'2021-09-23 01:03:00.76593'
description[text]:'TEST DATA TEST DATA TEST DATA TEST DATA'"
```

## How often AWS DMS uploads Time Travel logs to S3

To minimize the storage usage of your replication instance, AWS DMS offloads Time Travel logs from it periodically.

The Time travel logs get pushed to your Amazon S3 bucket in the cases following:

- If the current size of logs exceeds 1 GB, AWS DMS uploads the logs to S3 within five minutes. Thus, AWS DMS can make up to 12 calls an hour to S3 and AWS KMS for each running task.
- AWS DMS uploads the logs to S3 every hour, regardless of the size of the logs.
- When a task is stopped, AWS DMS immediately uploads the time travel logs to S3.

## Logging task settings

Logging uses Amazon CloudWatch to log information during the migration process. Using logging task settings, you can specify which component activities are logged and what amount of information is written to the log. Logging task settings are written to a JSON file. For information about how to use a task configuration file to set task settings, see [Task settings example \(p. 450\)](#).

You can turn on CloudWatch logging in several ways. You can select the `EnableLogging` option on the AWS Management Console when you create a migration task. Or, you can set the `EnableLogging` option to `true` when creating a task using the AWS DMS API. You can also specify "`EnableLogging": true`" in the JSON of the logging section of task settings.

CloudWatch integrates with AWS Identity and Access Management (IAM), and you can specify which CloudWatch actions a user in your AWS account can perform. For more information about working with IAM in CloudWatch, see [Identity and access management for amazon CloudWatch](#) and [Logging Amazon CloudWatch API calls in the Amazon CloudWatch User Guide](#).

To delete the task logs, you can set `DeleteTaskLogs` to `true` in the JSON of the logging section of the task settings.

You can specify logging for the following types of events:

- `FILE_FACTORY` – The file factory manages files used for batch apply and batch load, and manages Amazon S3 endpoints.
- `METADATA_MANAGER` – The metadata manager manages source and target metadata, partitioning, and table state during replication.
- `SORTER` – The SORTER receives incoming events from the `SOURCE_CAPTURE` process. The events are batched in transactions, and passed to the `TARGET_APPLY` service component. If the `SOURCE_CAPTURE` process produces events faster than the `TARGET_APPLY` component can consume them, the SORTER component caches the backlogged events to disk or to a swap file. Cached events are a common cause for running out of storage in replication instances.

The SORTER service component manages cached events, gathers CDC statistics, and reports task latency.

- `SOURCE_CAPTURE` – Ongoing replication (CDC) data is captured from the source database or service, and passed to the SORTER service component.
- `SOURCE_UNLOAD` – Data is unloaded from the source database or service during Full Load.
- `TABLES_MANAGER` – The table manager tracks captured tables, manages the order of table migration, and collects table statistics.
- `TARGET_APPLY` – Data and data definition language (DDL) statements are applied to the target database.
- `TARGET_LOAD` – Data is loaded into the target database.
- `TASK_MANAGER` – The task manager manages running tasks, and breaks tasks down into sub-tasks for parallel data processing.
- `TRANSFORMATION` – Table-mapping transformation events. For more information, see [Using table mapping to specify task settings \(p. 491\)](#).
- `VALIDATOR/ VALIDATOR_EXT` – The VALIDATOR service component verifies that data was migrated accurately from the source to the target. For more information, see [Data validation \(p. 584\)](#).

After you specify one of the preceding, you can then specify the amount of information that is logged, as shown in the following list.

The levels of severity are in order from lowest to highest level of information. The higher levels always include information from the lower levels.

- **LOGGER\_SEVERITY\_ERROR** – Error messages are written to the log.
- **LOGGER\_SEVERITY\_WARNING** – Warnings and error messages are written to the log.
- **LOGGER\_SEVERITY\_INFO** – Informational messages, warnings, and error messages are written to the log.
- **LOGGER\_SEVERITY\_DEFAULT** – Informational messages, warnings, and error messages are written to the log.
- **LOGGER\_SEVERITY\_DEBUG** – Debug messages, informational messages, warnings, and error messages are written to the log.
- **LOGGER\_SEVERITY\_DETAILED\_DEBUG** – All information is written to the log.

The following JSON example shows task settings for logging all actions and levels of severity.

```

...
    "Logging": {
        "EnableLogging": true,
        "LogComponents": [
            {
                "Id": "FILE_FACTORY",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "METADATA_MANAGER",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "SORTER",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "SOURCE_CAPTURE",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "SOURCE_UNLOAD",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "TABLES_MANAGER",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "TARGET_APPLY",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            },
            {
                "Id": "TARGET_LOAD",
                "Severity": "LOGGER_SEVERITY_INFO"
            },
            {
                "Id": "TASK_MANAGER",
                "Severity": "LOGGER_SEVERITY_DEBUG"
            },
            {
                "Id": "TRANSFORMATION",
                "Severity": "LOGGER_SEVERITY_DEBUG"
            },
            {
                "Id": "VALIDATOR",
                "Severity": "LOGGER_SEVERITY_DEFAULT"
            }
        ],
        "CloudWatchLogGroup": null,
        "CloudWatchLogStream": null
    },
...

```

## Control table task settings

Control tables provide information about an AWS DMS task. They also provide useful statistics that you can use to plan and manage both the current migration task and future tasks. You can apply these

task settings in a JSON file or by choosing **Advanced Settings** on the [Create task](#) page in the AWS DMS console. The Apply Exceptions table (`dmslogs.awsdms_apply_exceptions`) is always created on database targets. For information about how to use a task configuration file to set task settings, see [Task settings example \(p. 450\)](#).

For full load and CDC (Migrate existing data and replicate ongoing changes) and CDC only (Replicate data changes only) tasks, you can also create additional tables, including the following:

- **Replication Status (`dmslogs.awsdms_status`)** – This table provides details about the current task. These include task status, amount of memory consumed by the task, and the number of changes not yet applied to the target. This table also gives the position in the source database where AWS DMS is currently reading. Also, it indicates if the task is in the full load phase or change data capture (CDC).
- **Suspended Tables (`dmslogs.awsdms_suspended_tables`)** – This table provides a list of suspended tables as well as the reason they were suspended.
- **Replication History (`dmslogs.awsdms_history`)** – This table provides information about replication history. This information includes the number and volume of records processed during the task, latency at the end of a CDC task, and other statistics.

The Apply Exceptions table (`dmslogs.awsdms_apply_exceptions`) contains the following parameters.

Column	Type	Description
TASK_NAME	nvchar	The Resource ID of the AWS DMS task. Resource ID can be found in task ARN.
TABLE_OWNER	nvchar	The table owner.
TABLE_NAME	nvchar	The table name.
ERROR_TIME	timestamp	The time the exception (error) occurred.
STATEMENT	nvchar	The statement that was being run when the error occurred.
ERROR	nvchar	The error name and description.

The Replication Status table (`dmslogs.awsdms_status`) contains the current status of the task and the target database. It has the following settings.

Column	Type	Description
SERVER_NAME	nvchar	The name of the machine where the replication task is running.
TASK_NAME	nvchar	The Resource ID of the AWS DMS task. Resource ID can be found in task ARN.
TASK_STATUS	varchar	One of the following values: <ul style="list-style-type: none"> <li>• FULL LOAD</li> <li>• CHANGE PROCESSING (CDC)</li> </ul>

Column	Type	Description
		Task status is set to FULL LOAD as long as there is at least one table in full load. After all tables have been loaded, the task status changes to CHANGE PROCESSING if CDC is enabled.
STATUS_TIME	timestamp	The timestamp of the task status.
PENDING_CHANGES	int	The number of change records that weren't applied to the target.
DISK_SWAP_SIZE	int	The amount of disk space used by old or offloaded transactions.
TASK_MEMORY	int	Current memory used, in MB.
SOURCE_CURRENT_POSITION	varchar	The position in the source database that AWS DMS is currently reading from.
SOURCE_CURRENT_TIMESTAMP	timestamp	The timestamp in the source database that AWS DMS is currently reading from.
SOURCE_TAIL_POSITION	varchar	The position of the oldest start transaction that isn't committed. This value is the newest position that you can revert to without losing any changes.
SOURCE_TAIL_TIMESTAMP	timestamp	The timestamp of the oldest start transaction that isn't committed. This value is the newest timestamp that you can revert to without losing any changes.
SOURCE_TIMESTAMP_APPLIED	timestamp	The timestamp of the last transaction commit. In a bulk apply process, this value is the timestamp for the commit of the last transaction in the batch.

The Suspended table (`dmslogs.awsdms_suspended_tables`) contains the following parameters.

Column	Type	Description
SERVER_NAME	nvchar	The name of the machine where the replication task is running.
TASK_NAME	nvchar	The name of the AWS DMS task
TABLE_OWNER	nvchar	The table owner.
TABLE_NAME	nvchar	The table name.

Column	Type	Description
SUSPEND_REASON	nvchar	Reason for suspension.
SUSPEND_TIMESTAMP	timestamp	The time the suspension occurred.

The Replication History table (`dmslogs.awsdms_history`) contains the following parameters.

Column	Type	Description
SERVER_NAME	nvchar	The name of the machine where the replication task is running.
TASK_NAME	nvchar	The Resource ID of the AWS DMS task. Resource ID can be found in task ARN.
TIMESLOT_TYPE	varchar	<p>One of the following values:</p> <ul style="list-style-type: none"> <li>• FULL LOAD</li> <li>• CHANGE PROCESSING (CDC)</li> </ul> <p>If the task is running both full load and CDC, two history records are written to the time slot.</p>
TIMESLOT	timestamp	The ending timestamp of the time slot.
TIMESLOT_DURATION	int	The duration of the time slot, in minutes.
TIMESLOT_LATENCY	int	The target latency at the end of the time slot, in seconds. This value only applies to CDC time slots.
RECORDS	int	The number of records processed during the time slot.
TIMESLOT_VOLUME	int	The volume of data processed in MB.

The Validation Failure table (`awsdms_validation_failures_v1`) contains all the data validation failures for a task. For more information see, [Data Validation Troubleshooting \(p. 589\)](#).

Additional control table settings include the following:

- `HistoryTimeslotInMinutes` – Use this option to indicate the length of each time slot in the Replication History table. The default is 5 minutes.
- `ControlSchema` – Use this option to indicate the database schema name for the control tables for the AWS DMS target. If you don't enter any information for this option, then the tables are copied to the default location in the database as listed following:
  - PostgreSQL, Public
  - Oracle, the target schema
  - Microsoft SQL Server, dbo in the target database
  - MySQL, awsdms\_control

- MariaDB, awsdms\_control
- Amazon Redshift, Public
- DynamoDB, created as individual tables in the database

## Stream buffer task settings

You can set stream buffer settings using the AWS CLI, including the following. For information about how to use a task configuration file to set task settings, see [Task settings example \(p. 450\)](#).

- StreamBufferCount – Use this option to specify the number of data stream buffers for the migration task. The default stream buffer number is 3. Increasing the value of this setting might increase the speed of data extraction. However, this performance increase is highly dependent on the migration environment, including the source system and instance class of the replication server. The default is sufficient for most situations.
- StreamBufferSizeInMB – Use this option to indicate the maximum size of each data stream buffer. The default size is 8 MB. You might need to increase the value for this option when you work with very large LOBs. You also might need to increase the value if you receive a message in the log files that the stream buffer size is insufficient. When calculating the size of this option, you can use the following equation: [Max LOB size (or LOB chunk size)]\*[number of LOB columns]\*[number of stream buffers]\*[number of tables loading in parallel per task(MaxFullLoadSubTasks)]\*3
- CtrlStreamBufferSizeInMB – Use this option to set the size of the control stream buffer. The value is in megabytes, and can be 1–8. The default value is 5. You might need to increase this when working with a very large number of tables, such as tens of thousands of tables.

## Change processing tuning settings

The following settings determine how AWS DMS handles changes for target tables during change data capture (CDC). Several of these settings depend on the value of the target metadata parameter BatchApplyEnabled. For more information on the BatchApplyEnabled parameter, see [Target metadata task settings \(p. 452\)](#). For information about how to use a task configuration file to set task settings, see [Task settings example \(p. 450\)](#).

Change processing tuning settings include the following:

The following settings apply only when the target metadata parameter BatchApplyEnabled is set to true.

- BatchApplyPreserveTransaction – If set to true, transactional integrity is preserved and a batch is guaranteed to contain all the changes within a transaction from the source. The default value is true. This setting applies only to Oracle target endpoints.

If set to false, there can be temporary lapses in transactional integrity to improve performance. There is no guarantee that all the changes within a transaction from the source are applied to the target in a single batch.

By default, AWS DMS processes changes in a transactional mode, which preserves transactional integrity. If you can afford temporary lapses in transactional integrity, you can use the batch optimized apply option instead. This option efficiently groups transactions and applies them in batches for efficiency purposes. Using the batch optimized apply option almost always violates referential integrity constraints. So we recommend that you turn these constraints off during the migration process and turn them on again as part of the cutover process.

- BatchApplyTimeoutMin – Sets the minimum amount of time in seconds that AWS DMS waits between each application of batch changes. The default value is 1.

- **BatchApplyTimeoutMax** – Sets the maximum amount of time in seconds that AWS DMS waits between each application of batch changes before timing out. The default value is 30.
- **BatchApplyMemoryLimit** – Sets the maximum amount of memory in (MB) to use for pre-processing in **Batch optimized apply mode**. The default value is 500.
- **BatchSplitSize** – Sets the maximum number of changes applied in a single batch. The default value 0, meaning there is no limit applied.

The following settings apply only when the target metadata parameter **BatchApplyEnabled** is set to **false**.

- **MinTransactionSize** – Sets the minimum number of changes to include in each transaction. The default value is 1000.
- **CommitTimeout** – Sets the maximum time in seconds for AWS DMS to collect transactions in batches before declaring a timeout. The default value is 1.

For bidirectional replication, the following setting applies only when the target metadata parameter **BatchApplyEnabled** is set to **false**.

- **LoopbackPreventionSettings** – These settings provide loopback prevention for each ongoing replication task in any pair of tasks involved in bidirectional replication. *Loopback prevention* prevents identical changes from being applied in both directions of the bidirectional replication, which can corrupt data. For more information about bidirectional replication, see [Performing bidirectional replication \(p. 485\)](#).

AWS DMS attempts to keep transaction data in memory until the transaction is fully committed to the source, the target, or both. However, transactions that are larger than the allocated memory or that aren't committed within the specified time limit are written to disk.

The following settings apply to change processing tuning regardless of the change processing mode.

- **MemoryLimitTotal** – Sets the maximum size (in MB) that all transactions can occupy in memory before being written to disk. The default value is 1024.
- **MemoryKeepTime** – Sets the maximum time in seconds that each transaction can stay in memory before being written to disk. The duration is calculated from the time that AWS DMS started capturing the transaction. The default value is 60.
- **StatementCacheSize** – Sets the maximum number of prepared statements to store on the server for later execution when applying changes to the target. The default value is 50. The maximum value is 200.

Following is an example of how task settings that handle Change Processing Tuning appear in a task setting JSON file:

```
"ChangeProcessingTuning": {  
    "BatchApplyPreserveTransaction": true,  
    "BatchApplyTimeoutMin": 1,  
    "BatchApplyTimeoutMax": 30,  
    "BatchApplyMemoryLimit": 500,  
    "BatchSplitSize": 0,  
    "MinTransactionSize": 1000,  
    "CommitTimeout": 1,  
    "MemoryLimitTotal": 1024,  
    "MemoryKeepTime": 60,  
    "StatementCacheSize": 50  
}
```

To control the frequency of writes to an Amazon S3 target during a data replication task, you can configure the `cdcMaxBatchInterval` and `cdcMinFileSize` extra connection attributes. This can result in better performance when analyzing the data without any additional overhead operations. For more information, see [Extra connection attributes when using Amazon S3 as a target for AWS DMS \(p. 350\)](#).

## Data validation task settings

You can ensure that your data was migrated accurately from the source to the target. If you enable validation for a task, AWS DMS begins comparing the source and target data immediately after a full load is performed for a table. For more information about task data validation, its requirements, the scope of its database support, and the metrics it reports, see [AWS DMS data validation \(p. 584\)](#). For information about how to use a task configuration file to set task settings, see [Task settings example \(p. 450\)](#).

The data validation settings and their values include the following:

- `EnableValidation` – Enables data validation when set to true. Otherwise, validation is disabled for the task. The default value is false.
- `ValidationMode` – Controls how DMS will validate the data in target table against source table. AWS DMS provides this setting for future extensibility. Currently, the default and only valid value is `ROW_LEVEL`. AWS DMS validates all rows between the source and target tables.
- `FailureMaxCount` – Specifies the maximum number of records that can fail validation before validation is suspended for the task. The default value is 10,000. If you want the validation to continue regardless of the number of records that fail validation, set this value higher than the number of records in the source.
- `HandleCollationDiff` – When this option is set to true, the validation accounts for column collation differences in PostgreSQL and SQL Server endpoints when identifying source and target records to compare. Otherwise, any such differences in column collation are ignored for validation. Column collations can dictate the order of rows, which is important for data validation. Setting `HandleCollationDiff` to true resolves those collation differences automatically and prevents false positives in data validation. The default value is false.
- `RecordFailureDelayInMinutes` – Specifies the delay time in minutes before reporting any validation failure details.
- `RecordFailureDelayLimitInMinutes` – Specifies the delay before reporting any validation failure details. Normally, AWS DMS uses the task latency to recognize actual delay for changes to make it to the target in order to prevent false positives. This setting overrides the actual delay value and enables you to set a higher delay before reporting any validation metrics. The default value is 0.
- `RecordSuspendDelayInMinutes` – Specifies the delay time in minutes before tables are suspended from validation due to error threshold set in `FailureMaxCount`.
- `SkipLobColumns` – When this option is set to true, AWS DMS skips data validation for all the LOB columns in the table's part of the task validation. The default value is false.
- `TableFailureMaxCount` – Specifies the maximum number of rows in one table that can fail validation before validation is suspended for the table. The default value is 1,000.
- `ThreadCount` – Specifies the number of execution threads that AWS DMS uses during validation. Each thread selects not-yet-validated data from the source and target to compare and validate. The default value is 5. If you set `ThreadCount` to a higher number, AWS DMS can complete the validation faster. However, AWS DMS then runs more simultaneous queries, consuming more resources on the source and the target.
- `ValidationOnly` – When this option is set to true, the task performs data validation without performing any migration or replication of data. The default value is false. You can't modify the `ValidationOnly` setting after the task is created.

You must set `TargetTablePrepMode` to `DO_NOTHING` (the default for a validation only task) and set `Migration Type` to one of the following:

- Full Load — Set the task **Migration type** to **Migrate existing data** in the AWS DMS console. Or, in the AWS DMS API set the migration type to FULL-LOAD.
- CDC — Set the task **Migration type** to **Replicate data changes only** in the AWS DMS console. Or, in the AWS DMS API set the migration type to CDC.

Regardless of the migration type chosen, data isn't actually migrated or replicated during a validation only task.

For more information, see [Validation only tasks \(p. 588\)](#).

**Important**

The ValidationOnly setting is immutable. It can't be modified for a task after that task is created.

- ValidationPartialLobSize – Specifies if you want to do partial validation for LOB columns instead of validating all of the data stored in the column. This is something you might find useful when you are migrating just part of the LOB data and not the whole LOB data set. The value is in KB units. The default value is 0, which means AWS DMS validates all the LOB column data. For example, "ValidationPartialLobSize": 32 means that AWS DMS only validates the first 32KB of the column data in both the source and target.
- PartitionSize – Specifies the batch size of records to read for comparison from both source and target. The default is 10,000.
- ValidationQueryCdcDelaySeconds – The amount of time the first validation query is delayed on both source and target for each CDC update. This might help reduce resource contention when migration latency is high. A validation only task automatically sets this option to 180 seconds. The default is 0.

For example, the following JSON enables data validation with twice the default number of threads. It also accounts for differences in record order caused by column collation differences in PostgreSQL endpoints. Also, it provides a validation reporting delay to account for additional time to process any validation failures.

```
"ValidationSettings": {  
    "EnableValidation": true,  
    "ThreadCount": 10,  
    "HandleCollationDiff": true,  
    "RecordFailureDelayLimitInMinutes": 30  
}
```

**Note**

For an Oracle endpoint, AWS DMS uses DBMS\_CRYPTO to validate BLOBS. If your Oracle endpoint uses BLOBS, grant the execute permission for DBMS\_CRYPTO to the user account that accesses the Oracle endpoint. To do this, run the following statement.

```
grant execute on sys.dbms_crypto to dms_endpoint_user;
```

## Task settings for change processing DDL handling

The following settings determine how AWS DMS handles data definition language (DDL) changes for target tables during change data capture (CDC). For information about how to use a task configuration file to set task settings, see [Task settings example \(p. 450\)](#).

Task settings to handle change processing DDL include the following:

- `HandleSourceTableDropped` – Set this option to `true` to drop the target table when the source table is dropped.
- `HandleSourceTableTruncated` – Set this option to `true` to truncate the target table when the source table is truncated.
- `HandleSourceTableAltered` – Set this option to `true` to alter the target table when the source table is altered.

Following is an example of how task settings that handle change processing DDL appear in a task setting JSON file:

```
"ChangeProcessingDdlHandlingPolicy": {  
    "HandleSourceTableDropped": true,  
    "HandleSourceTableTruncated": true,  
    "HandleSourceTableAltered": true  
},
```

**Note**

For information about which DDL statements are supported for a specific endpoint, see the topic describing that endpoint.

## Character substitution task settings

You can specify that your replication task perform character substitutions on the target database for all source database columns with the AWS DMS STRING or WSTRING data type. For information about how to use a task configuration file to set task settings, see [Task settings example \(p. 450\)](#).

You can configure character substitution for any task with endpoints from the following source and target databases:

- Source databases:
  - Oracle
  - Microsoft SQL Server
  - MySQL
  - PostgreSQL
  - SAP Adaptive Server Enterprise (ASE)
  - IBM Db2 LUW
- Target databases:
  - Oracle
  - Microsoft SQL Server
  - MySQL
  - PostgreSQL
  - SAP Adaptive Server Enterprise (ASE)
  - Amazon Redshift

You can specify character substitutions using the `CharacterSetSettings` parameter in your task settings. These character substitutions occur for characters specified using the Unicode code point value in hexadecimal notation. You can implement the substitutions in two phases, in the following order if both are specified:

1. **Individual character replacement** – AWS DMS can replace the values of selected characters on the source with specified replacement values of corresponding characters on the target. Use the

CharacterReplacements array in CharacterSetSettings to select all source characters having the Unicode code points you specify. Use this array also to specify the replacement code points for the corresponding characters on the target.

To select all characters on the source that have a given code point, set an instance of SourceCharacterCodePoint in the CharacterReplacements array to that code point. Then specify the replacement code point for all equivalent target characters by setting the corresponding instance of TargetCharacterCodePoint in this array. To delete target characters instead of replacing them, set the appropriate instances of TargetCharacterCodePoint to zero (0). You can replace or delete as many different values of target characters as you want by specifying additional pairs of SourceCharacterCodePoint and TargetCharacterCodePoint settings in the CharacterReplacements array. If you specify the same value for multiple instances of SourceCharacterCodePoint, the value of the last corresponding setting of TargetCharacterCodePoint applies on the target.

For example, suppose that you specify the following values for CharacterReplacements.

```
"CharacterSetSettings": {
    "CharacterReplacements": [ {
        "SourceCharacterCodePoint": 62,
        "TargetCharacterCodePoint": 61
    }, {
        "SourceCharacterCodePoint": 42,
        "TargetCharacterCodePoint": 41
    }
}
```

In this example, AWS DMS replaces all characters with the source code point hex value 62 on the target by characters with the code point value 61. Also, AWS DMS replaces all characters with the source code point 42 on the target by characters with the code point value 41. In other words, AWS DMS replaces all instances of the letter 'b' on the target by the letter 'a'. Similarly, AWS DMS replaces all instances of the letter 'B' on the target by the letter 'A'.

2. **Character set validation and replacement** – After any individual character replacements complete, AWS DMS can make sure that all target characters have valid Unicode code points in the single character set that you specify. You use CharacterSetSupport in CharacterSetSettings to configure this target character verification and modification. To specify the verification character set, set CharacterSet in CharacterSetSupport to the character set's string value. (The possible values for CharacterSet follow.) You can have AWS DMS modify the invalid target characters in one of the following ways:

- Specify a single replacement Unicode code point for all invalid target characters, regardless of their current code point. To configure this replacement code point, set ReplaceWithCharacterCodePoint in CharacterSetSupport to the specified value.
- Configure the deletion of all invalid target characters by setting ReplaceWithCharacterCodePoint to zero (0).

For example, suppose that you specify the following values for CharacterSetSupport.

```
"CharacterSetSettings": {
    "CharacterSetSupport": {
        "CharacterSet": "UTF16_PlatformEndian",
        "ReplaceWithCharacterCodePoint": 0
    }
}
```

In this example, AWS DMS deletes any characters found on the target that are invalid in the "UTF16\_PlatformEndian" character set. So, any characters specified with the hex value 2FB6 are

deleted. This value is invalid because this is a 4-byte Unicode code point and UTF16 character sets accept only characters with 2-byte code points.

**Note**

The replication task completes all of the specified character substitutions before starting any global or table-level transformations that you specify through table mapping. For more information about table mapping, see [Using table mapping to specify task settings \(p. 491\)](#).

The values that AWS DMS supports for CharacterSet appear in the table following.

UTF-8	ibm-860_P100-1995	ibm-280_P100-1995
UTF-16	ibm-861_P100-1995	ibm-284_P100-1995
UTF-16BE	ibm-862_P100-1995	ibm-285_P100-1995
UTF-16LE	ibm-863_P100-1995	ibm-290_P100-1995
UTF-32	ibm-864_X110-1999	ibm-297_P100-1995
UTF-32BE	ibm-865_P100-1995	ibm-420_X120-1999
UTF-32LE	ibm-866_P100-1995	ibm-424_P100-1995
UTF16_PlatformEndian	ibm-867_P100-1998	ibm-500_P100-1995
UTF16_OppositeEndian	ibm-868_P100-1995	ibm-803_P100-1999
UTF32_PlatformEndian	ibm-869_P100-1995	ibm-838_P100-1995
UTF32_OppositeEndian	ibm-878_P100-1996	ibm-870_P100-1995
UTF-16BE,version=1	ibm-901_P100-1999	ibm-871_P100-1995
UTF-16LE,version=1	ibm-902_P100-1999	ibm-875_P100-1995
UTF-16,version=1	ibm-922_P100-1999	ibm-918_P100-1995
UTF-16,version=2	ibm-1168_P100-2002	ibm-930_P120-1999
UTF-7	ibm-4909_P100-1999	ibm-933_P110-1995
IMAP-mailbox-name	ibm-5346_P100-1998	ibm-935_P110-1999
SCSU	ibm-5347_P100-1998	ibm-937_P110-1999
BOCU-1	ibm-5348_P100-1997	ibm-939_P120-1999
CESU-8	ibm-5349_P100-1998	ibm-1025_P100-1995
ISO-8859-1	ibm-5350_P100-1998	ibm-1026_P100-1995
US-ASCII	ibm-9447_P100-2002	ibm-1047_P100-1995
gb18030	ibm-9448_X100-2005	ibm-1097_P100-1995
ibm-912_P100-1995	ibm-9449_P100-2002	ibm-1112_P100-1995
ibm-913_P100-2000	ibm-5354_P100-1998	ibm-1122_P100-1999
ibm-914_P100-1995	ibm-1250_P100-1995	ibm-1123_P100-1995

ibm-915_P100-1995	ibm-1251_P100-1995	ibm-1130_P100-1997
ibm-1089_P100-1995	ibm-1252_P100-2000	ibm-1132_P100-1998
ibm-9005_X110-2007	ibm-1253_P100-1995	ibm-1137_P100-1999
ibm-813_P100-1995	ibm-1254_P100-1995	ibm-4517_P100-2005
ibm-5012_P100-1999	ibm-1255_P100-1995	ibm-1140_P100-1997
ibm-916_P100-1995	ibm-5351_P100-1998	ibm-1141_P100-1997
ibm-920_P100-1995	ibm-1256_P110-1997	ibm-1142_P100-1997
iso-8859_10-1998	ibm-5352_P100-1998	ibm-1143_P100-1997
iso-8859_11-2001	ibm-1257_P100-1995	ibm-1144_P100-1997
ibm-921_P100-1995	ibm-5353_P100-1998	ibm-1145_P100-1997
iso-8859_14-1998	ibm-1258_P100-1997	ibm-1146_P100-1997
ibm-923_P100-1998	macos-0_2-10.2	ibm-1147_P100-1997
ibm-942_P12A-1999	macos-6_2-10.4	ibm-1148_P100-1997
ibm-943_P15A-2003	macos-7_3-10.2	ibm-1149_P100-1997
ibm-943_P130-1999	macos-29-10.2	ibm-1153_P100-1999
ibm-33722_P12A_P12A-2009_U	macos-35-10.2	ibm-1154_P100-1999
ibm-33722_P120-1999	ibm-1051_P100-1995	ibm-1155_P100-1999
ibm-954_P101-2007	ibm-1276_P100-1995	ibm-1156_P100-1999
euc-jp-2007	ibm-1006_P100-1995	ibm-1157_P100-1999
ibm-1373_P100-2002	ibm-1098_P100-1995	ibm-1158_P100-1999
windows-950-2000	ibm-1124_P100-1996	ibm-1160_P100-1999
ibm-950_P110-1999	ibm-1125_P100-1997	ibm-1164_P100-1999
ibm-1375_P100-2008	ibm-1129_P100-1997	ibm-1364_P110-2007
ibm-5471_P100-2006	ibm-1131_P100-1997	ibm-1371_P100-1999
ibm-1386_P100-2001	ibm-1133_P100-1997	ibm-1388_P103-2001
windows-936-2000	ISO_2022,locale=ja,version=10	ibm-1390_P110-2003
ibm-1383_P110-1999	ISO_2022,locale=ja,version=1ibm-1399_P110-2003	ibm-1399_P110-2003
ibm-5478_P100-1995	ISO_2022,locale=ja,version=1ibm-5123_P100-1999	ibm-5123_P100-1999
euc-tw-2014	ISO_2022,locale=ja,version=1ibm-8482_P100-1999	ibm-8482_P100-1999
ibm-964_P110-1999	ISO_2022,locale=ja,version=1ibm-16684_P110-2003	ibm-16684_P110-2003
ibm-949_P110-1999	ISO_2022,locale=ko,version=1ibm-4899_P100-1998	ibm-4899_P100-1998
ibm-949_P11A-1999	ISO_2022,locale=ko,version=1ibm-4971_P100-1999	ibm-4971_P100-1999

ibm-970_P110_P110-2006_U2	ISO_2022,locale=zh,version=10m-9067_X100-2005	
ibm-971_P100-1995	ISO_2022,locale=zh,version=ibm-12712_P100-1998	
ibm-1363_P11B-1998	ISO_2022,locale=zh,version=ibm-16804_X110-1999	
ibm-1363_P110-1997	HZ	ibm-37_P100-1995,swaplfnl
windows-949-2000	x11-compound-text	ibm-1047_P100-1995,swaplfnl
windows-874-2000	ISCII,version=0	ibm-1140_P100-1997,swaplfnl
ibm-874_P100-1995	ISCII,version=1	ibm-1141_P100-1997,swaplfnl
ibm-1162_P100-1999	ISCII,version=2	ibm-1142_P100-1997,swaplfnl
ibm-437_P100-1995	ISCII,version=3	ibm-1143_P100-1997,swaplfnl
ibm-720_P100-1997	ISCII,version=4	ibm-1144_P100-1997,swaplfnl
ibm-737_P100-1997	ISCII,version=5	ibm-1145_P100-1997,swaplfnl
ibm-775_P100-1996	ISCII,version=6	ibm-1146_P100-1997,swaplfnl
ibm-850_P100-1995	ISCII,version=7	ibm-1147_P100-1997,swaplfnl
ibm-851_P100-1995	ISCII,version=8	ibm-1148_P100-1997,swaplfnl
ibm-852_P100-1995	LMBCS-1	ibm-1149_P100-1997,swaplfnl
ibm-855_P100-1995	ibm-37_P100-1995	ibm-1153_P100-1999,swaplfnl
ibm-856_P100-1995	ibm-273_P100-1995	ibm-12712_P100-1998,swaplfnl
ibm-857_P100-1995	ibm-277_P100-1995	ibm-16804_X110-1999,swaplfnl
ibm-858_P100-1997	ibm-278_P100-1995	ebcdic-xml-us

## Before image task settings

When writing CDC updates to a data-streaming target like Kinesis or Apache Kafka, you can view a source database row's original values before change by an update. To make this possible, AWS DMS populates a *before image* of update events based on data supplied by the source database engine. For information about how to use a task configuration file to set task settings, see [Task settings example \(p. 450\)](#).

To do so, you use the `BeforeImageSettings` parameter, which adds a new JSON attribute to every update operation with values collected from the source database system.

Make sure to apply `BeforeImageSettings` only to full load plus CDC tasks or CDC only tasks. Full load plus CDC tasks migrate existing data and replicate ongoing changes. CDC only tasks replicate data changes only.

Don't apply `BeforeImageSettings` to tasks that are full load only.

Possible options for `BeforeImageSettings` are the following:

- `EnableBeforeImage` – Turns on before imaging when set to `true`. The default is `false`.
- `FieldName` – Assigns a name to the new JSON attribute. When `EnableBeforeImage` is `true`, `FieldName` is required and can't be empty.

- **ColumnFilter** – Specifies a column to add by using before imaging. To add only columns that are part of the table's primary keys, use the default value, `pk-only`. To add any column that has a before image value, use `all`. Note that the before image doesn't support large binary object (LOB) data types such as CLOB and BLOB.

The following shows an example of the use of `BeforeImageSettings`.

```
"BeforeImageSettings": {  
    "EnableBeforeImage": true,  
    "FieldName": "before-image",  
    "ColumnFilter": "pk-only"  
}
```

For information on before image settings for Kinesis, including additional table mapping settings, see [Using a before image to view original values of CDC rows for a Kinesis data stream as a target \(p. 378\)](#).

For information on before image settings for Kafka, including additional table mapping settings, see [Using a before image to view original values of CDC rows for Apache Kafka as a target \(p. 394\)](#).

## Error handling task settings

You can set the error handling behavior of your replication task during change data capture (CDC) using the following settings. For information about how to use a task configuration file to set task settings, see [Task settings example \(p. 450\)](#).

- **DataErrorPolicy** – Determines the action AWS DMS takes when there is an error related to data processing at the record level. Some examples of data processing errors include conversion errors, errors in transformation, and bad data. The default is `LOG_ERROR`.
  - `IGNORE_RECORD` – The task continues and the data for that record is ignored. The error counter for the `DataErrorEscalationCount` property is incremented. Thus, if you set a limit on errors for a table, this error counts toward that limit.
  - `LOG_ERROR` – The task continues and the error is written to the task log.
  - `SUSPEND_TABLE` – The task continues but data from the table with the error record is moved into an error state and the data isn't replicated.
  - `STOP_TASK` – The task stops and manual intervention is required.
- **DataTruncationErrorPolicy** – Determines the action AWS DMS takes when data is truncated. The default is `LOG_ERROR`.
  - `IGNORE_RECORD` – The task continues and the data for that record is ignored. The error counter for the `DataErrorEscalationCount` property is incremented. Thus, if you set a limit on errors for a table, this error counts toward that limit.
  - `LOG_ERROR` – The task continues and the error is written to the task log.
  - `SUSPEND_TABLE` – The task continues but data from the table with the error record is moved into an error state and the data isn't replicated.
  - `STOP_TASK` – The task stops and manual intervention is required.
- **DataErrorEscalationPolicy** – Determines the action AWS DMS takes when the maximum number of errors (set in the `DataErrorEscalationCount` parameter) is reached. The default is `SUSPEND_TABLE`.
  - `SUSPEND_TABLE` – The task continues but data from the table with the error record is moved into an error state and the data isn't replicated.
  - `STOP_TASK` – The task stops and manual intervention is required.
- **DataErrorEscalationCount** – Sets the maximum number of errors that can occur to the data for a specific record. When this number is reached, the data for the table that contains the error record is handled according to the policy set in the `DataErrorEscalationPolicy`. The default is 0.

- **EventErrorPolicy** – Determines the action AWS DMS takes when an error occurs while sending a task-related event. Its possible values are
  - IGNORE – The task continues and any data associated with that event is ignored.
  - STOP\_TASK – The task stops and manual intervention is required.
- **TableErrorPolicy** – Determines the action AWS DMS takes when an error occurs when processing data or metadata for a specific table. This error only applies to general table data and isn't an error that relates to a specific record. The default is SUSPEND\_TABLE.
  - SUSPEND\_TABLE – The task continues but data from the table with the error record is moved into an error state and the data isn't replicated.
  - STOP\_TASK – The task stops and manual intervention is required.
- **TableErrorEscalationPolicy** – Determines the action AWS DMS takes when the maximum number of errors (set using the TableErrorEscalationCount parameter). The default and only user setting is STOP\_TASK, where the task is stopped and manual intervention is required.
- **TableErrorEscalationCount** – The maximum number of errors that can occur to the general data or metadata for a specific table. When this number is reached, the data for the table is handled according to the policy set in the TableErrorEscalationPolicy. The default is 0.
- **RecoverableErrorCount** – The maximum number of attempts made to restart a task when an environmental error occurs. After the system attempts to restart the task the designated number of times, the task is stopped and manual intervention is required. The default value is -1, which instructs AWS DMS to attempt to restart the task indefinitely. Set this value to 0 to never attempt to restart a task. If a fatal error occurs, AWS DMS stops attempting to restart the task after six attempts.
- **RecoverableErrorInterval** – The number of seconds that AWS DMS waits between attempts to restart a task. The default is 5.
- **RecoverableErrorThrottling** – When enabled, the interval between attempts to restart a task is increased in a series based on the value of RecoverableErrorInterval. For example, if RecoverableErrorInterval is set to 5 seconds, then the next retry will happen after 10 seconds, then 20, then 40 seconds and so on. The default is true.
- **RecoverableErrorThrottlingMax** – The maximum number of seconds that AWS DMS waits between attempts to restart a task if RecoverableErrorThrottling is enabled. The default is 1800.
- **RecoverableErrorStopRetryAfterThrottlingMax** – When set to true, stops restarting the task after the maximum number of seconds that AWS DMS waits between recovery attempts is reached, per RecoverableErrorThrottlingMax.
- **ApplyErrorDeletePolicy** – Determines what action AWS DMS takes when there is a conflict with a DELETE operation. The default is IGNORE\_RECORD. Possible values are the following:
  - IGNORE\_RECORD – The task continues and the data for that record is ignored. The error counter for the ApplyErrorEscalationCount property is incremented. Thus, if you set a limit on errors for a table, this error counts toward that limit.
  - LOG\_ERROR – The task continues and the error is written to the task log.
  - SUSPEND\_TABLE – The task continues but data from the table with the error record is moved into an error state and the data isn't replicated.
  - STOP\_TASK – The task stops and manual intervention is required.
- **ApplyErrorInsertPolicy** – Determines what action AWS DMS takes when there is a conflict with an INSERT operation. The default is LOG\_ERROR. Possible values are the following:
  - IGNORE\_RECORD – The task continues and the data for that record is ignored. The error counter for the ApplyErrorEscalationCount property is incremented. Thus, if you set a limit on errors for a table, this error counts toward that limit.
  - LOG\_ERROR – The task continues and the error is written to the task log.
  - SUSPEND\_TABLE – The task continues but data from the table with the error record is moved into an error state and the data isn't replicated.
  - STOP\_TASK – The task stops and manual intervention is required.

- **INSERT\_RECORD** – If there is an existing target record with the same primary key as the inserted source record, the target record is updated.
- **ApplyErrorUpdatePolicy** – Determines what action AWS DMS takes when there is a missing data conflict with an UPDATE operation. The default is LOG\_ERROR. This error handling doesn't support LOBs. Possible values are the following:
  - **IGNORE\_RECORD** – The task continues and the data for that record is ignored. The error counter for the ApplyErrorEscalationCount property is incremented. Thus, if you set a limit on errors for a table, this error counts toward that limit.
  - **LOG\_ERROR** – The task continues and the error is written to the task log.
  - **SUSPEND\_TABLE** – The task continues but data from the table with the error record is moved into an error state and the data isn't replicated.
  - **STOP\_TASK** – The task stops and manual intervention is required.
  - **UPDATE\_RECORD** – If the target record is missing, the missing target record is inserted into the target table. Selecting this option requires full supplemental logging to be enabled for all the source table columns when Oracle is the source database.
- **ApplyErrorEscalationPolicy** – Determines what action AWS DMS takes when the maximum number of errors (set using the ApplyErrorEscalationCount parameter) is reached. The default is LOG\_ERROR:
  - **LOG\_ERROR** – The task continues and the error is written to the task log.
  - **SUSPEND\_TABLE** – The task continues but data from the table with the error record is moved into an error state and the data isn't replicated.
  - **STOP\_TASK** – The task stops and manual intervention is required.
- **ApplyErrorEscalationCount** – This option sets the maximum number of APPLY conflicts that can occur for a specific table during a change process operation. When this number is reached, the table data is handled according to the policy set in the ApplyErrorEscalationPolicy parameter. The default is 0.
- **ApplyErrorFailOnTruncationDdl** – Set this option to true to cause the task to fail when a truncation is performed on any of the tracked tables during CDC. The default is false.

This approach doesn't work with PostgreSQL version 11.x or lower, or any other source endpoint that doesn't replicate DDL table truncation.

- **FailOnNoTablesCaptured** – Set this option to true to cause a task to fail when the table mappings defined for a task find no tables when the task starts. The default is false.
- **FailOnTransactionConsistencyBreached** – This option applies to tasks using Oracle as a source with CDC. The default is false. Set it to true to cause a task to fail when a transaction is open for more time than the specified timeout and can be dropped.

When a CDC task starts with Oracle, AWS DMS waits for a limited time for the oldest open transaction to close before starting CDC. If the oldest open transaction doesn't close until the timeout is reached, then in most cases AWS DMS starts CDC, ignoring that transaction. If this option is set to true, the task fails.

- **FullLoadIgnoreConflicts** – Set this option to true to have AWS DMS ignore "zero rows affected" and "duplicates" errors when applying cached events. If set to false, AWS DMS reports all errors instead of ignoring them. The default is true.

## Saving task settings

You can save task settings as a JSON file in case you want to reuse the settings for another task. You can find tasks settings to copy to a JSON file under the **Overview details** section of a task.

### Note

While reusing task settings for other tasks, remove any CloudWatchLogGroup and CloudWatchLogStream attributes. Otherwise, the following error is given: SYSTEM ERROR

MESSAGE:Task Settings CloudWatchLogGroup or CloudWatchLogStream cannot be set on create.

For example, the following JSON file contains settings saved for a task.

```
{  
    "TargetMetadata": {  
        "TargetSchema": "",  
        "SupportLobs": true,  
        "FullLobMode": false,  
        "LobChunkSize": 0,  
        "LimitedSizeLobMode": true,  
        "LobMaxSize": 32,  
        "InlineLobMaxSize": 0,  
        "LoadMaxFileSize": 0,  
        "ParallelLoadThreads": 0,  
        "ParallelLoadBufferSize": 0,  
        "BatchApplyEnabled": false,  
        "TaskRecoveryTableEnabled": false,  
        "ParallelLoadQueuesPerThread": 0,  
        "ParallelApplyThreads": 0,  
        "ParallelApplyBufferSize": 0,  
        "ParallelApplyQueuesPerThread": 0  
    },  
    "FullLoadSettings": {  
        "TargetTablePrepMode": "DO_NOTHING",  
        "CreatePkAfterFullLoad": false,  
        "StopTaskCachedChangesApplied": false,  
        "StopTaskCachedChangesNotApplied": false,  
        "MaxFullLoadSubTasks": 8,  
        "TransactionConsistencyTimeout": 600,  
        "CommitRate": 10000  
    },  
    "Logging": {  
        "EnableLogging": true,  
        "LogComponents": [  
            {  
                "Id": "TRANSFORMATION",  
                "Severity": "LOGGER_SEVERITY_DEFAULT"  
            },  
            {  
                "Id": "SOURCE_UNLOAD",  
                "Severity": "LOGGER_SEVERITY_DEFAULT"  
            },  
            {  
                "Id": "IO",  
                "Severity": "LOGGER_SEVERITY_DEFAULT"  
            },  
            {  
                "Id": "TARGET_LOAD",  
                "Severity": "LOGGER_SEVERITY_DEFAULT"  
            },  
            {  
                "Id": "PERFORMANCE",  
                "Severity": "LOGGER_SEVERITY_DEFAULT"  
            },  
            {  
                "Id": "SOURCE_CAPTURE",  
                "Severity": "LOGGER_SEVERITY_DEFAULT"  
            },  
            {  
                "Id": "SORTER",  
                "Severity": "LOGGER_SEVERITY_DEFAULT"  
            },  
        ]  
    }  
}
```

```
{
    "Id": "REST_SERVER",
    "Severity": "LOGGER_SEVERITY_DEFAULT"
},
{
    "Id": "VALIDATOR_EXT",
    "Severity": "LOGGER_SEVERITY_DEFAULT"
},
{
    "Id": "TARGET_APPLY",
    "Severity": "LOGGER_SEVERITY_DEFAULT"
},
{
    "Id": "TASK_MANAGER",
    "Severity": "LOGGER_SEVERITY_DEFAULT"
},
{
    "Id": "TABLES_MANAGER",
    "Severity": "LOGGER_SEVERITY_DEFAULT"
},
{
    "Id": "METADATA_MANAGER",
    "Severity": "LOGGER_SEVERITY_DEFAULT"
},
{
    "Id": "FILE_FACTORY",
    "Severity": "LOGGER_SEVERITY_DEFAULT"
},
{
    "Id": "COMMON",
    "Severity": "LOGGER_SEVERITY_DEFAULT"
},
{
    "Id": "ADDONS",
    "Severity": "LOGGER_SEVERITY_DEFAULT"
},
{
    "Id": "DATA_STRUCTURE",
    "Severity": "LOGGER_SEVERITY_DEFAULT"
},
{
    "Id": "COMMUNICATION",
    "Severity": "LOGGER_SEVERITY_DEFAULT"
},
{
    "Id": "FILE_TRANSFER",
    "Severity": "LOGGER_SEVERITY_DEFAULT"
}
],
},
"ControlTablesSettings": {
    "historyTimeslotInMinutes": 5,
    "ControlSchema": "",
    "HistoryTimeslotInMinutes": 5,
    "HistoryTableEnabled": false,
    "SuspendedTablesTableEnabled": false,
    "StatusTableEnabled": false,
    "FullLoadExceptionTableEnabled": false
},
"StreamBufferSettings": {
    "StreamBufferCount": 3,
    "StreamBufferSizeInMB": 8,
    "CtrlStreamBufferSizeInMB": 5
},
"ChangeProcessingDdlHandlingPolicy": {
    "HandleSourceTableDropped": true,
}
```

```

        "HandleSourceTableTruncated": true,
        "HandleSourceTableAltered": true
    },
    "ErrorBehavior": {
        "DataErrorPolicy": "LOG_ERROR",
        "DataTruncationErrorPolicy": "LOG_ERROR",
        "DataErrorEscalationPolicy": "SUSPEND_TABLE",
        "DataErrorEscalationCount": 0,
        "TableErrorPolicy": "SUSPEND_TABLE",
        "TableErrorEscalationPolicy": "STOP_TASK",
        "TableErrorEscalationCount": 0,
        "RecoverableErrorCount": -1,
        "RecoverableErrorInterval": 5,
        "RecoverableErrorThrottling": true,
        "RecoverableErrorThrottlingMax": 1800,
        "RecoverableErrorStopRetryAfterThrottlingMax": true,
        "ApplyErrorDeletePolicy": "IGNORE_RECORD",
        "ApplyErrorInsertPolicy": "LOG_ERROR",
        "ApplyErrorUpdatePolicy": "LOG_ERROR",
        "ApplyErrorEscalationPolicy": "LOG_ERROR",
        "ApplyErrorEscalationCount": 0,
        "ApplyErrorFailOnTruncationDdl": false,
        "FullLoadIgnoreConflicts": true,
        "FailOnTransactionConsistencyBreached": false,
        "FailOnNoTablesCaptured": true
    },
    "ChangeProcessingTuning": {
        "BatchApplyPreserveTransaction": true,
        "BatchApplyTimeoutMin": 1,
        "BatchApplyTimeoutMax": 30,
        "BatchApplyMemoryLimit": 500,
        "BatchSplitSize": 0,
        "MinTransactionSize": 1000,
        "CommitTimeout": 1,
        "MemoryLimitTotal": 1024,
        "MemoryKeepTime": 60,
        "StatementCacheSize": 50
    },
    "PostProcessingRules": null,
    "CharacterSetSettings": null,
    "LoopbackPreventionSettings": null,
    "BeforeImageSettings": null,
    "FailTaskWhenCleanTaskResourceFailed": false
}

```

## Setting LOB support for source databases in an AWS DMS task

Large binary objects (LOBs) can sometimes be difficult to migrate between systems. AWS DMS offers a number of options to help with the tuning of LOB columns. To see which and when data types are considered LOBs by AWS DMS, see the AWS DMS documentation.

When you migrate data from one database to another, you might take the opportunity to rethink how your LOBs are stored, especially for heterogeneous migrations. If you want to do so, there's no need to migrate the LOB data.

If you decide to include LOBs, you can then decide the other LOB settings:

- The LOB mode determines how LOBs are handled:

- **Full LOB mode** – In full LOB mode AWS DMS migrates all LOBs from source to target regardless of size. In this configuration, AWS DMS has no information about the maximum size of LOBs to expect. Thus, LOBs are migrated one at a time, piece by piece. Full LOB mode can be quite slow.
- **Limited LOB mode** – In limited LOB mode, you set a maximum LOB size for DMS to accept. That enables DMS to pre-allocate memory and load the LOB data in bulk. LOBs that exceed the maximum LOB size are truncated, and a warning is issued to the log file. In limited LOB mode, you can gain significant performance over full LOB mode. We recommend that you use limited LOB mode whenever possible. The maximum recommended value is 102400 KB (100 MB).

**Note**

Using the Max LOB size (K) option with a value greater than 63KB impacts the performance of a full load configured to run in limited LOB mode. During a full load, DMS allocates memory by multiplying the Max LOB size (k) value by the Commit rate, and the product is multiplied by the number of LOB columns. When DMS can't pre-allocate that memory, DMS starts consuming SWAP memory, and that impacts performance of a full load. So, if you experience performance issues when using limited LOB mode, consider decreasing the commit rate until you achieve an acceptable level of performance. You can also consider using inline LOB mode for supported endpoints once you understand your LOB distribution for the table.

To validate limited LOB size, you must set ValidationPartialLobSize to the same value as LobMaxSize (K).

- **Inline LOB mode** – In inline LOB mode, you set the maximum LOB size that DMS transfers inline. LOBs smaller than the specified size are transferred inline. LOBs larger than the specified size are replicated using full LOB mode. You can select this option to replicate both small and large LOBs when most of the LOBs are small. DMS doesn't support inline LOB mode for endpoints that don't support Full LOB mode, like S3 and Redshift.

**Note**

With Oracle, LOBs are treated as VARCHAR data types whenever possible. This approach means that AWS DMS fetches them from the database in bulk, which is significantly faster than other methods. The maximum size of a VARCHAR in Oracle is 32 K. Therefore, a limited LOB size of less than 32 K is optimal when Oracle is your source database.

- When a task is configured to run in limited LOB mode, the **Max LOB size (K)** option sets the maximum size LOB that AWS DMS accepts. Any LOBs that are larger than this value are truncated to this value.
- When a task is configured to use full LOB mode, AWS DMS retrieves LOBs in pieces. The **LOB chunk size (K)** option determines the size of each piece. When setting this option, pay particular attention to the maximum packet size allowed by your network configuration. If the LOB chunk size exceeds your maximum allowed packet size, you might see disconnect errors. The recommended value for LobChunkSize is 64 kilobytes. Increasing the value for LobChunkSize above 64 kilobytes can cause task failures.
- When a task is configured to run in inline LOB mode, the **InlineLobMaxSize** setting determines which LOBs DMS transfers inline.

For information on the task settings to specify these options, see [Target metadata task settings \(p. 452\)](#)

## Creating multiple tasks

In some migration scenarios, you might have to create several migration tasks. Tasks work independently and can run concurrently. Each task has its own initial load, CDC, and log reading process. Tables that are related through data manipulation language (DML) must be part of the same task.

Some reasons to create multiple tasks for a migration include the following:

- The target tables for the tasks reside on different databases, such as when you are fanning out or breaking a system into multiple systems.

- You want to break the migration of a large table into multiple tasks by using filtering.

**Note**

Because each task has its own change capture and log reading process, changes are *not* coordinated across tasks. Therefore, when using multiple tasks to perform a migration, make sure that source transactions are wholly contained within a single task.

## Creating tasks for ongoing replication using AWS DMS

You can create an AWS DMS task that captures ongoing changes from the source data store. You can do this capture while you are migrating your data. You can also create a task that captures ongoing changes after you complete your initial (full-load) migration to a supported target data store. This process is called ongoing replication or change data capture (CDC). AWS DMS uses this process when replicating ongoing changes from a source data store. This process works by collecting changes to the database logs using the database engine's native API.

**Note**

You can migrate views using full-load tasks only. If your task is either a CDC-only task or a full-load task that starts CDC after it completes, the migration includes only tables from the source. Using a full-load-only task, you can migrate views or a combination of tables and views. For more information, see [Specifying table selection and transformations rules using JSON \(p. 494\)](#).

Each source engine has specific configuration requirements for exposing this change stream to a given user account. Most engines require some additional configuration to make it possible for the capture process to consume the change data in a meaningful way, without data loss. For example, Oracle requires the addition of supplemental logging, and MySQL requires row-level binary logging (bin logging).

To read ongoing changes from the source database, AWS DMS uses engine-specific API actions to read changes from the source engine's transaction logs. Following are some examples of how AWS DMS does that:

- For Oracle, AWS DMS uses either the Oracle LogMiner API or binary reader API (bfile API) to read ongoing changes. AWS DMS reads ongoing changes from the online or archive redo logs based on the system change number (SCN).
- For Microsoft SQL Server, AWS DMS uses MS-Replication or MS-CDC to write information to the SQL Server transaction log. It then uses the `fn_dblog()` or `fn_dump_dblog()` function in SQL Server to read the changes in the transaction log based on the log sequence number (LSN).
- For MySQL, AWS DMS reads changes from the row-based binary logs (binlogs) and migrates those changes to the target.
- For PostgreSQL, AWS DMS sets up logical replication slots and uses the `test_decoding` plugin to read changes from the source and migrate them to the target.
- For Amazon RDS as a source, we recommend ensuring that backups are enabled to set up CDC. We also recommend ensuring that the source database is configured to retain change logs for a sufficient time —24 hours is usually enough.

There are two types of ongoing replication tasks:

- Full load plus CDC – The task migrates existing data and then updates the target database based on changes to the source database.
- CDC only – The task migrates ongoing changes after you have data on your target database.

## Performing replication starting from a CDC start point

You can start an AWS DMS ongoing replication task (change data capture only) from several points. These include the following:

- **From a custom CDC start time** – You can use the AWS Management Console or AWS CLI to provide AWS DMS with a timestamp where you want the replication to start. AWS DMS then starts an ongoing replication task from this custom CDC start time. AWS DMS converts the given timestamp (in UTC) to a native start point, such as an LSN for SQL Server or an SCN for Oracle. AWS DMS uses engine-specific methods to determine where to start the migration task based on the source engine's change stream.

### Note

Only by setting the `StartFromContext` connection attribute to the required timestamp does Db2 as a source offer a customized CDC start time.

PostgreSQL as a source doesn't support a custom CDC start time. This is because the PostgreSQL database engine doesn't have a way to map a timestamp to an LSN or SCN as Oracle and SQL Server do.

- **From a CDC native start point** – You can also start from a native point in the source engine's transaction log. In some cases, you might prefer this approach because a timestamp can indicate multiple native points in the transaction log. AWS DMS supports this feature for the following source endpoints:
  - SQL Server
  - PostgreSQL
  - Oracle
  - MySQL

When the task is created, AWS DMS marks the CDC start point, and it can't be changed. To use a different CDC start point, create a new task.

## Determining a CDC native start point

A *CDC native start point* is a point in the database engine's log that defines a time where you can begin CDC. As an example, suppose that a bulk data dump has already been applied to the target. You can look up the native start point for the ongoing replication-only task. To avoid any data inconsistencies, carefully choose the start point for the replication-only task. DMS captures transactions that started after the chosen CDC start point.

Following are examples of how you can find the CDC native start point from supported source engines:

### SQL Server

In SQL Server, a log sequence number (LSN) has three parts:

- Virtual log file (VLF) sequence number
- Starting offset of a log block
- Slot number

An example LSN is as follows: `00000014:00000061:0001`

To get the start point for a SQL Server migration task based on your transaction log backup settings, use the `fn_dblog()` or `fn_dump_dblog()` function in SQL Server.

To use CDC native start point with SQL Server, create a publication on any table participating in ongoing replication. AWS DMS creates the publication automatically when you use CDC without using a CDC native start point.

## PostgreSQL

You can use a CDC recovery checkpoint for your PostgreSQL source database. This checkpoint value is generated at various points as an ongoing replication task runs for your source database (the parent task). For more information about checkpoints in general, see [Using a checkpoint as a CDC start point \(p. 484\)](#).

To identify the checkpoint to use as your native start point, use your database pg\_replication\_slots view or your parent task's overview details from the AWS Management Console

### To find the overview details for your parent task on the console

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.  
If you are signed in as an IAM user, make sure that you have the appropriate permissions to access AWS DMS. For more information about the permissions required, see [IAM permissions needed to use AWS DMS \(p. 628\)](#).
2. On the navigation pane, choose **Database migration tasks**.
3. Choose your parent task from the list on the **Database migration tasks** page. Doing this opens your parent task page, showing the overview details.
4. Find the checkpoint value under **Change data capture (CDC)**, **Change data capture (CDC) start position**, and **Change data capture (CDC) recovery checkpoint**.

The value appears similar to the following.

```
checkpoint:V1#1#000004AF/B00000D0#0#0#*#0#0
```

Here, the 4AF/B00000D0 component is what you need to specify this native CDC start point. Set the DMS API CdcStartPosition parameter to this value when you create the CDC task to begin replication at this start point for your PostgreSQL source. For information on using the AWS CLI to create this CDC task, see [Enabling CDC with an AWS-managed PostgreSQL DB instance with AWS DMS \(p. 199\)](#).

## Oracle

A system change number (SCN) is a logical, internal time stamp used by Oracle databases. SCNs order events that occur within the database, which is necessary to satisfy the ACID properties of a transaction. Oracle databases use SCNs to mark the location where all changes have been written to disk so that a recovery action doesn't apply already written changes. Oracle also uses SCNs to mark the point where no redo exists for a set of data so that recovery can stop.

To get the current SCN in an Oracle database, run the following command.

```
SELECT CURRENT_SCN FROM V$DATABASE
```

If you use the current SCN to start a CDC task, you miss the results of any open transactions and fail to migrate these results. *Open transactions* are transactions that were started earlier but not committed yet. You can identify the SCN and timestamp to start a CDC task at a point that includes all open transactions. For more information, see [Transactions](#) in the Oracle online documentation.

## MySQL

Before the release of MySQL version 5.6.3, the log sequence number (LSN) for MySQL was a 4-byte unsigned integer. In MySQL version 5.6.3, when the redo log file size limit increased from 4 GB to

512 GB, the LSN became an 8-byte unsigned integer. The increase reflects that additional bytes were required to store extra size information. Applications built on MySQL 5.6.3 or later that use LSN values should use 64-bit rather than 32-bit variables to store and compare LSN values. For more information about MySQL LSNs, see the [MySQL documentation](#).

To get the current LSN in a MySQL database, run the following command.

```
mysql> show master status;
```

The query returns a binlog file name, the position, and several other values. The CDC native start point is a combination of the binlogs file name and the position, for example mysql-bin-changelog.000024:373. In this example, mysql-bin-changelog.000024 is the binlogs file name and 373 is the position where AWS DMS needs to start capturing changes.

## Using a checkpoint as a CDC start point

An ongoing replication task migrates changes, and AWS DMS caches checkpoint information specific to AWS DMS from time to time. The checkpoint that AWS DMS creates contains information so the replication engine knows the recovery point for the change stream. You can use the checkpoint to go back in the timeline of changes and recover a failed migration task. You can also use a checkpoint to start another ongoing replication task for another target at any given point in time.

You can get the checkpoint information in one of the following three ways:

- Run the API operation `DescribeReplicationTasks` and view the results. You can filter the information by task and search for the checkpoint. You can retrieve the latest checkpoint when the task is in stopped or failed state. This information is lost if the task is deleted.
- View the metadata table named `awsdms_txn_state` on the target instance. You can query the table to get checkpoint information. To create the metadata table, set the `TaskRecoveryTableEnabled` parameter to `Yes` when you create a task. This setting causes AWS DMS to continuously write checkpoint information to the target metadata table. This information is lost if a task is deleted.

For example, the following is a sample checkpoint in the metadata table:  
`checkpoint:V1#34#00000132/0F000E48#0#0#*#0#121`

- From the navigation pane, choose **Database migration tasks**, and choose your parent task from the list that appears on the Database migration tasks page. Your parent task page opens, showing the overview details. Find the checkpoint value under Change data capture (CDC), Change data capture (CDC) start position, and Change data capture (CDC) recovery checkpoint. The checkpoint value appears similar to the following:

`checkpoint:V1#1#000004AF/B00000D0#0#0#*#0#0`

## Stopping a task at a commit or server time point

With the introduction of CDC native start points, AWS DMS can also stop a task at the following points:

- A commit time on the source
- A server time on the replication instance

You can modify a task and set a time in UTC to stop as needed. The task automatically stops based on the commit or server time that you set. Or, if you know an appropriate time to stop the migration task at task creation, you can set a stop time when you create the task.

## Performing bidirectional replication

You can use AWS DMS tasks to perform bidirectional replication between two systems. In *bidirectional replication*, you replicate data from the same table (or set of tables) between two systems in both directions.

For example, you can copy an EMPLOYEE table from database A to database B and replicate changes to the table from database A to database B. You can also replicate changes to the EMPLOYEE table from database B back to A. Thus, you're performing bidirectional replication.

**Note**

AWS DMS bidirectional replication isn't intended as a full multi-master solution including a primary node, conflict resolution, and so on.

Use bidirectional replication for situations where data on different nodes is operationally segregated. In other words, suppose that you have a data element changed by an application operating on node A, and that node A performs bidirectional replication with node B. That data element on node A is never changed by any application operating on node B.

AWS DMS supports bidirectional replication on these database engines:

- Oracle
- SQL Server
- MySQL
- PostgreSQL
- Amazon Aurora MySQL-Compatible Edition
- Aurora PostgreSQL-Compatible Edition

## Creating bidirectional replication tasks

To enable AWS DMS bidirectional replication, configure source and target endpoints for both databases (A and B). For example, configure a source endpoint for database A, a source endpoint for database B, a target endpoint for database A, and a target endpoint for database B.

Then create two tasks: one task for source A to move data to target B, and another task for source B to move data to target A. Also, make sure that each task is configured with loopback prevention. Doing this prevents identical changes from being applied to the targets of both tasks, thus corrupting the data for at least one of them. For more information, see [Preventing loopback \(p. 486\)](#).

For the easiest approach, start with identical datasets on both database A and database B. Then create two CDC only tasks, one task to replicate data from A to B, and another task to replicate data from B to A.

To use AWS DMS to instantiate a new dataset (database) on node B from node A, do the following:

1. Use a full load and CDC task to move data from database A to B. Make sure that no applications are modifying data on database B during this time.
2. When the full load is complete and before applications are allowed to modify data on database B, note the time or CDC start position of database B. For instructions, see [Performing replication starting from a CDC start point \(p. 482\)](#).
3. Create a CDC only task that moves data from database B back to A using this start time or CDC start position.

**Note**

Only one task in a bidirectional pair can be full load and CDC.

## Preventing loopback

To show preventing loopback, suppose that in a task T1 AWS DMS reads change logs from source database A and applies the changes to target database B.

Next, a second task, T2, reads change logs from source database B and applies them back to target database A. Before T2 does this, DMS must make sure that the same changes made to target database B from source database A aren't made to source database A. In other words, DMS must make sure that these changes aren't echoed (looped) back to target database A. Otherwise, the data in database A can be corrupted.

To prevent loopback of changes, add the following task settings to each bidirectional replication task. Doing this makes sure that loopback data corruption doesn't occur in either direction.

```
{
    ...
    "LoopbackPreventionSettings": {
        "EnableLoopbackPrevention": Boolean,
        "SourceSchema": String,
        "TargetSchema": String
    },
    ...
}
```

The LoopbackPreventionSettings task settings determine if a transaction is new or an echo from the opposite replication task. When AWS DMS applies a transaction to a target database, it updates a DMS table (`awsdms_loopback_prevention`) with an indication of the change. Before applying each transaction to a target, DMS ignores any transaction that includes a reference to this `awsdms_loopback_prevention` table. Therefore, it doesn't apply the change.

Include these task settings with each replication task in a bidirectional pair. These settings enable loopback prevention. They also specify the schema for each source and target database in the task that includes the `awsdms_loopback_prevention` table for each endpoint.

To enable each task to identify such an echo and discard it, set `EnableLoopbackPrevention` to `true`. To specify a schema at the source that includes `awsdms_loopback_prevention`, set `SourceSchema` to the name for that schema in the source database. To specify a schema at the target that includes the same table, set `TargetSchema` to the name for that schema in the target database.

In the example following, the `SourceSchema` and `TargetSchema` settings for a replication task T1 and its opposite replication task T2 are specified with opposite settings.

Settings for task T1 are as follows.

```
{
    ...
    "LoopbackPreventionSettings": {
        "EnableLoopbackPrevention": true,
        "SourceSchema": "LOOP-DATA",
        "TargetSchema": "loop-data"
    },
    ...
}
```

Settings for opposite task T2 are as follows.

```
{  
. . .  
    "LoopbackPreventionSettings": {  
        "EnableLoopbackPrevention": true,  
        "SourceSchema": "loop-data",  
        "TargetSchema": "LOOP-DATA"  
    },  
. . .  
}
```

**Note**

When using the AWS CLI, use only the `create-replication-task` or `modify-replication-task` commands to configure `LoopbackPreventionSettings` in your bidirectional replication tasks.

## Limitations of bidirectional replication

Bidirectional replication for AWS DMS has the following limitations:

- Loopback prevention tracks only data manipulation language (DML) statements. AWS DMS doesn't support preventing data definition language (DDL) loopback. To do this, configure one of the tasks in a bidirectional pair to filter out DDL statements.
- Tasks that use loopback prevention don't support committing changes in batches. To configure a task with loopback prevention, make sure to set `BatchApplyEnabled` to `false`.
- DMS bidirectional replication doesn't include conflict detection or resolution. To detect data inconsistencies, use data validation on both tasks.

## Modifying a task

You can modify a task if you need to change the task settings, table mapping, or other settings. You can also enable and run premigration assessments before running the modified task. You can modify a task in the console by selecting the task and choosing **Modify**. You can also use the CLI command or API operation [ModifyReplicationTask](#).

There are a few limitations to modifying a task. These include the following:

- You can't modify the source or target endpoint of a task.
- You can't change the migration type of a task.
- Tasks that have run must have a status of **Stopped** or **Failed** to be modified.

## Moving a task

You can move a task to a different replication instance when any of the following situations apply to your use case.

- You're currently using an instance of a certain type and you want to switch to a different instance type.
- Your current instance is overloaded by many replication tasks, and you want to split the load across multiple instances.
- Your instance storage is full, and you want to move tasks off that instance to a more powerful instance as an alternative to scaling storage or compute.

- You want to use a newly released feature of AWS DMS, but don't want to create a new task and restart the migration. Instead, you prefer to spin up a replication instance with a new AWS DMS version that supports the feature, and move the existing task to that instance.

You can move a task in the console by selecting the task and choosing **Move**. You can also run the CLI command or API operation `MoveReplicationTask` to move the task. You can move a task that has any database engine as its target endpoint.

Make sure that the target replication instance has enough storage space to accommodate the task that's being moved. Otherwise, scale the storage to make space for your target replication instance before moving the task.

Also, make sure that your target replication instance is created with the same or later AWS DMS engine version as the current replication instance.

**Note**

- You can't move a task to the same replication instance where it currently resides.
- You can't modify the settings of a task while it's moving.
- A task you have run must have a status of **Stopped**, **Failed**, or **Failed-move** before you can move it.

There are two task statuses that relate to moving a DMS task, **Moving** and **Failed-move**. For more information about those task status, see [Task status \(p. 559\)](#).

After moving a task, you can enable and run premigration assessments to check for blocking issues before running the moved task.

## Reloading tables during a task

While a task is running, you can reload a target database table using data from the source. You might want to reload a table if, during the task, an error occurs or data changes due to partition operations (for example, when using Oracle). You can reload up to 10 tables from a task.

Reloading tables does not stop the task.

To reload a table, the following conditions must apply:

- The task must be running.
- The migration method for the task must be either full load or full load with CDC.
- Duplicate tables aren't allowed.
- AWS DMS retains the previously read table definition and doesn't recreate it during the reload operation. Any DDL statements such as `ALTER TABLE ADD COLUMN` or `DROP COLUMN` that are made to the table before the table is reloaded can cause the reload operation to fail.

## AWS Management Console

### To reload a table using the AWS DMS console

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.

If you are signed in as an IAM user, make sure that you have the appropriate permissions to access AWS DMS. For more information about the permissions required, see [IAM permissions needed to use AWS DMS \(p. 628\)](#).

2. Choose **Tasks** from the navigation pane.
3. Choose the running task that has the table you want to reload.
4. Choose the **Table Statistics** tab.

Create task   Assess   Modify   Start/Resume   Stop

Filter:  Filter

ID	Status	Source
move-data	Running	from-mysql-sou

move-data

Overview   Task monitoring   **Table statistics**   Logs   A

Revalidate   **Reload table data**

Filter:  Filter

Schema	Table	Load State	Inserts	Deletes
employees	departments	Table completed	0	0
employees	dept_emp	Table completed	0	0
employees	dept_manager	Table completed	0	0

5. Choose the table you want to reload. If the task is no longer running, you can't reload the table.
6. Choose **Reload table data**.

When AWS DMS is preparing to reload a table, the console changes the table status to **Table is being reloaded**.

## Using table mapping to specify task settings

Table mapping uses several types of rules to specify the data source, source schema, data, and any transformations that should occur during the task. You can use table mapping to specify individual tables in a database to migrate and the schema to use for the migration.

When working with table mapping, you can use filters to specify data that you want replicated from table columns. In addition, you can use transformations to modify selected schemas, tables, or views before they are written to the target database.

### Topics

- [Specifying table selection and transformations rules from the console \(p. 491\)](#)
- [Specifying table selection and transformations rules using JSON \(p. 494\)](#)
- [Selection rules and actions \(p. 495\)](#)
- [Wildcards in table mapping \(p. 499\)](#)
- [Transformation rules and actions \(p. 500\)](#)
- [Using transformation rule expressions to define column content \(p. 513\)](#)
- [Table and collection settings rules and operations \(p. 523\)](#)

### Note

When working with table mapping for a MongoDB source endpoint, you can use filters to specify data that you want replicated, and specify a database name in place of the schema\_name. Or, you can use the default "%".

## Specifying table selection and transformations rules from the console

You can use the AWS Management Console to perform table mapping, including specifying table selection and transformations. On the console, use the **Where** section to specify the schema, table, and action (include or exclude). Use the **Filter** section to specify the column name in a table and the conditions that you want to apply to a replication task. Together, these two actions create a selection rule.

You can include transformations in a table mapping after you have specified at least one selection rule. You can use transformations to rename a schema or table, add a prefix or suffix to a schema or table, or remove a table column.

### Note

AWS DMS doesn't support more than one transformation rule per schema level or per table level. However, AWS DMS does support more than one transformation rule per column level.

The following procedure shows how to set up selection rules, based on a table called **Customers** in a schema called **EntertainmentAgencySample**.

### To specify a table selection, filter criteria, and transformations using the console

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.

If you are signed in as an IAM user, make sure that you have the appropriate permissions to access AWS DMS. For more information about the permissions required, see [IAM permissions needed to use AWS DMS \(p. 628\)](#).

2. On the **Dashboard** page, choose **Database migration tasks**.
3. Choose **Create Task**.
4. In the **Task configuration** section, enter the task information, including **Task identifier**, **Replication instance**, **Source database endpoint**, **Target database endpoint**, and **Migration type**.

DMS > Database migration tasks > Create database migration task

## Create database migration task

### Task configuration

#### Task identifier

Type a unique identifier for the task

#### Replication instance

Choose a replication instance

#### Source database endpoint

Choose a source database endpoint

#### Target database endpoint

Choose a target database endpoint

#### Migration type [Info](#)

Migrate existing data

5. In the **Table mapping** section, enter the schema name and table name. You can use "%" as a wildcard value when specifying the schema name or the table name. For information about other wildcards you can use, see [the section called "Wildcards in table mapping" \(p. 499\)](#). Specify the action to be taken, to include or exclude data defined by the filter.

The screenshot shows the 'Table mappings' interface. At the top, there are two editing modes: 'Wizard' (selected) and 'JSON editor'. Below this, a note says 'Specify at least one selection rule with an include action. After you do this, you can add one or more transformation rules.' Under 'Selection rules', there is a section for 'where schema name is like 'MySchema' and table name is like '%', include'. This section includes fields for 'Schema' (dropdown), 'Schema name' (text input with placeholder 'Enter a schema'), 'Table name' (text input with placeholder 'Use the % character as a wildcard'), and 'Action' (dropdown with 'Include' selected). A red circle highlights the 'Add new selection rule' button.

**6. Specify filter information using the **Add column filter** and the **Add condition** links.**

- Choose **Add column filter** to specify a column and conditions.
- Choose **Add condition** to add additional conditions.

The following example shows a filter for the **Customers** table that includes **AgencyIDs** between **01** and **85**.

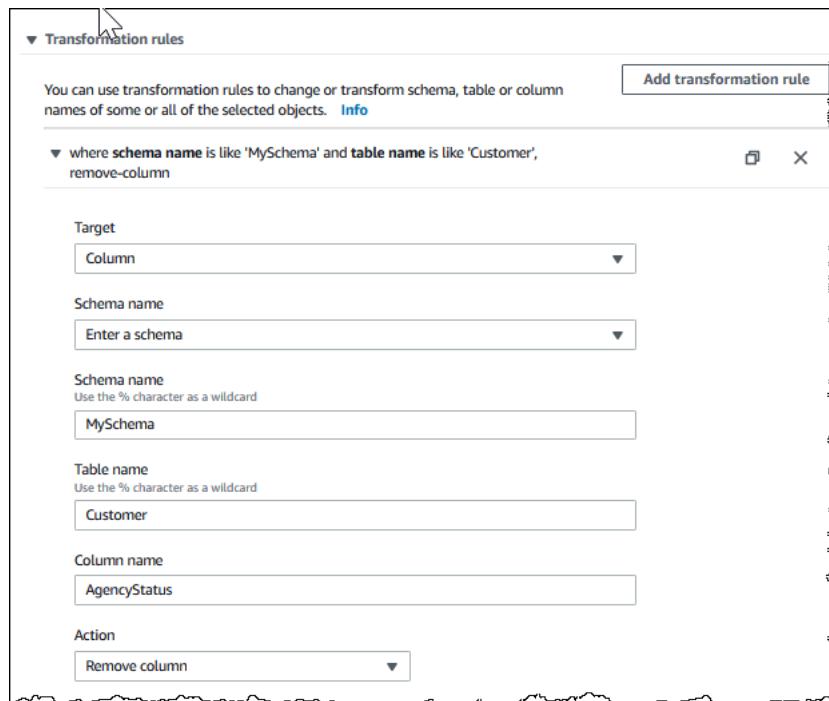
The screenshot shows the 'Source filters' section. It displays a 'Column filter 1' for the 'AgencyId' column. The 'Condition 1' dropdown is set to 'Equal to or between two values', with the values '01' and '85' entered. A red circle highlights the 'Add column filter' button.

**7. When you have created the selections you want, choose **Add new selection rule**.**

- After you have created at least one selection rule, you can add a transformation to the task. Choose **add transformation rule**.

The screenshot shows the 'Transformation rules' section. It contains a note: 'You can use transformation rules to change or transform schema, table or column names of some or all of the selected objects.' A red circle highlights the 'Add transformation rule' button.

9. Choose the target that you want to transform, and enter the additional information requested. The following example shows a transformation that deletes the **AgencyStatus** column from the **Customer** table.



10. Choose **Add transformation rule**.

11. Choose **Create task**.

**Note**

AWS DMS doesn't support more than one transformation rule per schema level or per table level. However, AWS DMS does support more than one transformation rule per column level.

## Specifying table selection and transformations rules using JSON

To specify the table mappings that you want to apply during migration, you can create a JSON file. If you create a migration task using the console, you can browse for this JSON file or enter the JSON directly into the table mapping box. If you use the CLI or API to perform migrations, you can specify this file using the `TableMappings` parameter of the `CreateReplicationTask` or `ModifyReplicationTask` API operation.

You can specify what tables, views, and schemas you want to work with. You can also perform table, view, and schema transformations and specify settings for how AWS DMS loads individual tables and views. You create table-mapping rules for these options using the following rule types:

- **selection rules** – Identify the types and names of source tables, views, and schemas to load. For more information, see [Selection rules and actions \(p. 495\)](#).
- **transformation rules** – Specify certain changes or additions to particular source tables and schemas on the source before they are loaded on the target. For more information, see [Transformation rules and actions \(p. 500\)](#).

Also, to define content of new and existing columns, you can use an expression within a transformation rule. For more information, see [Using transformation rule expressions to define column content \(p. 513\)](#).

- **table-settings** rules – Specify how DMS tasks load the data for individual tables. For more information, see [Table and collection settings rules and operations \(p. 523\)](#).

**Note**

For Amazon S3 targets, you can also tag S3 objects mapped to selected tables and schemas using the post-processing rule type and the add-tag rule action. For more information, see [Amazon S3 object tagging \(p. 336\)](#).

For the targets following, you can specify how and where selected schemas and tables are migrated to the target using the object-mapping rule type:

- Amazon DynamoDB – For more information, see [Using object mapping to migrate data to DynamoDB \(p. 364\)](#).
- Amazon Kinesis – For more information, see [Using object mapping to migrate data to a Kinesis data stream \(p. 382\)](#).
- Apache Kafka – For more information, see [Using object mapping to migrate data to a Kafka topic \(p. 397\)](#).

## Selection rules and actions

Using table mapping, you can specify what tables, views, and schemas you want to work with by using selection rules and actions. For table-mapping rules that use the selection rule type, you can apply the following values.

Parameter	Possible values	Description
rule-type	selection	A selection rule. Define at least one selection rule when specifying a table mapping.
rule-id	A numeric value.	A unique numeric value to identify the rule.
rule-name	An alphanumeric value.	A unique name to identify the rule.
rule-action	include, exclude, explicit	A value that includes or excludes the object or objects selected by the rule. If explicit is specified, you can select and include only one object that corresponds to an explicitly specified table and schema.
object-locator	An object with the following parameters: <ul style="list-style-type: none"> <li>• schema-name – The name of the schema.</li> <li>• table-name – The name of the table.</li> <li>• (Optional) table-type – table   view   all, to indicate if table-</li> </ul>	The name of each schema and table or view to which the rule applies. You can also specify if a rule includes only tables, only views, or both tables and views. If the rule-action is either include or exclude, you can use the "%" percent sign as a wildcard for all or part of the value for the schema-name and table-name parameter. For information about other wildcards

Parameter	Possible values	Description
	<p>name refers only to tables, views, or both tables and views. The default is table.</p> <p>AWS DMS loads views only in a full-load task. If you have only full-load and change data capture (CDC) tasks, configure at least one full-load-only task to load your views.</p> <p>Not all target endpoints accept views as a source of replication, even in full load (e.g. Amazon OpenSearch Service). Check the limitations of your target endpoint.</p>	<p>you can use, see <a href="#">the section called "Wildcards in table mapping" (p. 499)</a>. Thus, you can match these items:</p> <ul style="list-style-type: none"> <li>• A single table, view, or collection in a single schema</li> <li>• A single table, view, or collection in some or all schemas</li> <li>• Some or all tables and views in a single schema, or collections in a single database</li> <li>• Some or all tables and views in some or all schemas, or collections in some or all databases</li> </ul> <p>If the rule-action is explicit, you can only specify the exact name of a single table or view and its schema (with no wildcards).</p> <p>The supported sources for views include:</p> <ul style="list-style-type: none"> <li>• Oracle</li> <li>• Microsoft SQL Server</li> <li>• PostgreSQL</li> <li>• IBM Db2 LUW</li> <li>• SAP Adaptive Server Enterprise (ASE)</li> <li>• MySQL</li> <li>• AURORA</li> <li>• AURORA Serverless</li> <li>• MariaDB</li> </ul> <p><b>Note</b> AWS DMS never loads a source view to a target view. A source view is loaded to an equivalent table on the target with the same name as the view on the source.</p> <p>The supported sources for databases containing collections include:</p> <ul style="list-style-type: none"> <li>• MongoDB</li> <li>• Amazon DocumentDB</li> </ul>
load-order	A positive integer. The maximum value is 2,147,483,647.	The priority for loading tables and views. Tables and views with higher values are loaded first.

Parameter	Possible values	Description
filters	An array of objects.	One or more objects for filtering the source. You specify object parameters to filter on a single column in the source. You specify multiple objects to filter on multiple columns. For more information, see <a href="#">Using source filters (p. 543)</a> .

### Example Migrate all tables in a schema

The following example migrates all tables from a schema named Test in your source to your target endpoint.

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "Test",
        "table-name": "%"
      },
      "rule-action": "include"
    }
  ]
}
```

### Example Migrate some tables in a schema

The following example migrates all tables except those starting with DMS from a schema named Test in your source to your target endpoint.

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "Test",
        "table-name": "%"
      },
      "rule-action": "include"
    },
    {
      "rule-type": "selection",
      "rule-id": "2",
      "rule-name": "2",
      "object-locator": {
        "schema-name": "Test",
        "table-name": "DMS%"
      },
      "rule-action": "exclude"
    }
  ]
}
```

## Example Migrate a specified single table in single schema

The following example migrates the Customer table from the NewCust schema in your source to your target endpoint.

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "NewCust",
        "table-name": "Customer"
      },
      "rule-action": "explicit"
    }
  ]
}
```

### Note

You can explicitly select on multiple tables and schemas by specifying multiple selection rules.

## Example Migrate tables in a set order

The following example migrates two tables. Table loadfirst (with priority 2) is initialized before table loadsecond.

```
{
"rules": [
  {
    "rule-type": "selection",
    "rule-id": "1",
    "rule-name": "1",
    "object-locator": {
      "schema-name": "Test",
      "table-name": "loadsecond"
    },
    "rule-action": "include",
    "load-order": "1"
  },
  {
    "rule-type": "selection",
    "rule-id": "2",
    "rule-name": "2",
    "object-locator": {
      "schema-name": "Test",
      "table-name": "loadfirst"
    },
    "rule-action": "include",
    "load-order": "2"
  }
]
```

### Note

load-order is applicable for table initialization. The load of a successive table won't wait for a previous table load to complete if MaxFullLoadSubTasks is greater than 1.

## Example Migrate some views in a schema

The following example migrates some views from a schema named Test in your source to equivalent tables in your target.

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "2",
      "rule-name": "2",
      "object-locator": {
        "schema-name": "Test",
        "table-name": "view_DMS%",
        "table-type": "view"
      },
      "rule-action": "include"
    }
  ]
}
```

### Example Migrate all tables and views in a schema

The following example migrates all tables and views from a schema named `report` in your source to equivalent tables in your target.

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "3",
      "rule-name": "3",
      "object-locator": {
        "schema-name": "report",
        "table-name": "%",
        "table-type": "all"
      },
      "rule-action": "include"
    }
  ]
}
```

## Wildcards in table mapping

This section describes wildcards you can use when specifying the schema and table names for table mapping.

Wildcard	Matches
<code>%</code>	Zero or more characters
<code>-</code>	A single character
<code>[ ]</code>	A literal underscore character
<code>[ab]</code>	A set of characters. For example, <code>[ab]</code> matches either 'a' or 'b'.
<code>[a-d]</code>	A range of characters. For example, <code>[a-d]</code> matches either 'a', 'b', 'c', or 'd'.

For Oracle source and target endpoints, you can use the `escapeCharacter` extra connection attribute to specify an escape character. An escape character allows you to use a specified wildcard character in

expressions as if it was not wild. For example, `escapeCharacter=#` allows you to use '#' to make a wildcard character act as an ordinary character in an expression as in the this sample code.

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "542485267",
            "rule-name": "542485267",
            "object-locator": { "schema-name": "ROOT", "table-name": "TEST#_T%" },
            "rule-action": "include",
            "filters": []
        }
    ]
}
```

Here, the '#' escape character makes the '\_' wildcard character act as a normal character. AWS DMS selects tables in the schema named ROOT, where each table has a name with TEST\_T as its prefix.

## Transformation rules and actions

You use the transformation actions to specify any transformations you want to apply to the selected schema, table, or view. Transformation rules are optional.

### Limitations

- You can't apply more than one transformation rule action against the same object (schema, table, column, table-tablespace, or index-tablespace). You can apply several transformation rule actions on any level as long as each transformation action is applied against a different object.
- Column names in transformation rules are case-sensitive. For example, you must provide column names for an Oracle or Db2 database in upper case.
- Transformations are not supported for column names with Right-to-Left languages.
- Transformations cannot be performed on columns that contain special characters (e.g. #, \, /, -) in their name.
- The only supported transformation for columns that are mapped to BLOB/CLOB data types is to drop the column on the target.
- AWS DMS doesn't support replicating two source tables to a single target table. AWS DMS replicates records from table to table, and from column to column, according to the replication tasks's transformation rules. The object names must be unique to prevent overlapping.

For example, a source table has a column named ID and the corresponding target table has a pre-existing column called id. If a rule uses an ADD-COLUMN statement to add a new column called id, and a SQLite statement to populate the column with custom values, this creates a duplicate, ambiguous object named id and is not supported.

### Values

For table-mapping rules that use the transformation rule type, you can apply the following values.

Parameter	Possible values	Description
rule-type	transformation	A value that applies the rule to each object specified by the selection rule. Use transformation unless otherwise noted.

Parameter	Possible values	Description
rule-id	A numeric value.	A unique numeric value to identify the rule.
rule-name	An alphanumeric value.	A unique name to identify the rule.
object-locator	<p>An object with the following parameters:</p> <ul style="list-style-type: none"> <li>• schema-name – The name of the schema. For MongoDB and Amazon DocumentDB endpoints, this is the name of the database holding a set of collections.</li> <li>• table-name – The name of the table, view, or collection.</li> <li>• table-tablespace-name – The name of an existing table tablespace.</li> <li>• index-tablespace-name – The name of an existing index tablespace.</li> <li>• column-name – The name of an existing column.</li> <li>• data-type – The name of an existing column data type.</li> </ul>	<p>The name of each schema, table or view, table tablespace, index tablespace, and column to which the rule applies. You can use the "%" percent sign as a wildcard for all or part of the value of each object-locator parameter, except data-type. Thus, you can match these items:</p> <ul style="list-style-type: none"> <li>• A single table or view in a single schema</li> <li>• A single table or view in some or all schemas</li> <li>• Some or all tables and views in a single schema</li> <li>• Some or all tables and views in some or all schemas</li> <li>• One or more columns in the specified table or tables, view or views, and schema or schemas.</li> <li>• The columns with a given data-type when multiple columns are specified. For the possible values of data-type, see data-type described following in this table.</li> </ul> <p>Also, the table-tablespace-name or index-tablespace-name parameter is only available to match an Oracle source endpoint. You can specify either table-tablespace-name or index-tablespace-name in a single rule, but not both. Thus, you can match <i>either</i> of the following items:</p> <ul style="list-style-type: none"> <li>• One, some, or all table tablespaces</li> <li>• One, some, or all index tablespaces</li> </ul>

Parameter	Possible values	Description
rule-action	add-column, include-column, remove-column  rename  convert-lowercase, convert-uppercase  add-prefix, remove-prefix, replace-prefix  add-suffix, remove-suffix, replace-suffix  define-primary-key  change-data-type  add-before-image-columns	The transformation you want to apply to the object. All transformation rule actions are case-sensitive.  The add-column value of the rule-action parameter adds a column to a table. But you can't add a new column with the same name as an existing column of the same table.  When used with the expression and data-type parameters, add-column specifies the value of new column data.  The change-data-type value for rule-action is only available for column rule targets.  The include-column value of the rule-action parameter changes the mode of the table to <i>drop all columns by default</i> and <i>include the columns specified</i> . Multiple columns are included in the target by invoking the include-column rule multiple times.
rule-target	schema, table, column, table-tablespace, index-tablespace	The type of object that you're transforming.  The table-tablespace and index-tablespace values are only available for an Oracle target endpoint.  Make sure to specify a value for the parameter that you specify as part of the object-locator: table-tablespace-name or index-tablespace-name name.
value	An alphanumeric value that follows the naming rules for the target type.	The new value for actions that require input, such as rename.
old-value	An alphanumeric value that follows the naming rules for the target type.	The old value for actions that require replacement, such as replace-prefix.

Parameter	Possible values	Description
data-type	<p>type – The data type to use if the rule-action is add-column or the replacement data type if the rule-action is change-data-type.</p> <p>Or, the name of the replacement data type when rule-action is change-data-type, the value of column-name is "%", and an additional data-type parameter to identify the existing data type is included in the object-locator.</p> <p>AWS DMS supports column data type transformations for the following DMS data types: "bytes", "date", "time", "datetime", "int1", "int2", "int4", "int8", "numeric", "real4", "real8", "string", "uint1", "uint2", "uint4", "uint8", "wstring", "blob", "nclob", "clob", "boolean", "set", "list", "map", "tuple"</p> <p>precision – If the added column or replacement data type has a precision, an integer value to specify the precision.</p> <p>scale – If the added column or replacement data type has a scale, an integer value or date time value to specify the scale.</p> <p>length – The length of new column data (when used with add-column)</p>	<p>The following is an example of a data-type parameter to specify the existing data type to be replaced. This existing data-type parameter is included in the object-locator when the value of column-name is "%", shown following.</p> <pre> . . .  "object-locator": {     "schema-name": "dbo",     "table-name": "dms",     "column-name": "%",     "data-type": "int2" }, "data-type": {     "type": "int8" }</pre> <p>Here, any column with the int2 data type is replaced with the int8 data type.</p> <p>The length value for data-type is only available for use with column rule targets of add-column rule actions.</p>

Parameter	Possible values	Description
expression	An alphanumeric value that follows SQLite syntax.	<p>When used with the rule-action set to <code>rename-schema</code>, the expression parameter specifies a new schema. When used with the rule-action set to <code>rename-table</code>, expression specifies a new table. When used with the rule-action set to <code>rename-column</code>, expression specifies a new column name value.</p> <p>When used with the rule-action set to <code>add-column</code>, expression specifies data that makes up a new column.</p> <p>For more information about using expressions for transformation rules, see <a href="#">Using transformation rule expressions to define column content (p. 513)</a>.</p>
primary-key-def	<p>An object with the following parameters:</p> <ul style="list-style-type: none"> <li>• <code>name</code> – The name of a new primary key or unique index for the table or view.</li> <li>• (Optional) <code>origin</code> – The type of unique key to define: <code>primary-key</code> (the default) or <code>unique-index</code>.</li> <li>• <code>columns</code> – An array of strings listing the names of columns in the order they appear in the primary key or unique index.</li> </ul>	This parameter can define the name, type, and content of a unique key on the transformed table or view. It does so when the rule-action is set to <code>define-primary-key</code> and the rule-target is set to <code>table</code> . By default, the unique key is defined as a primary key.

Parameter	Possible values	Description
before-image-def	<p>An object with the following parameters:</p> <ul style="list-style-type: none"> <li>• column-prefix – A value prepended to a column name. The default value is BI_.</li> <li>• column-suffix – A value appended to the column name. The default is empty.</li> <li>• column-filter – Requires one of the following values: pk-only (default), non-lob (optional) and all (optional).</li> </ul>	<p>This parameter defines a naming convention to identify the before-image columns and specifies a filter to identify which source columns can have before-image columns created for them on the target. You can specify this parameter when the rule-action is set to add-before-image-columns and the rule-target is set to column.</p> <p>Don't set both column-prefix and column-suffix to empty strings.</p> <p>For column-filter, select:</p> <ul style="list-style-type: none"> <li>• pk-only – To add only columns that are part of table primary keys.</li> <li>• non-lob – To add only columns that are not of LOB type.</li> <li>• all – To add any column that has a before-image value.</li> </ul> <p>For more information about before-image support for AWS DMS target endpoints, see:</p> <ul style="list-style-type: none"> <li>• <a href="#">Using a before image to view original values of CDC rows for a Kinesis data stream as a target (p. 378)</a></li> <li>• <a href="#">Using a before image to view original values of CDC rows for Apache Kafka as a target (p. 394)</a></li> </ul>

## Examples

### Example Rename a schema

The following example renames a schema from Test in your source to Test1 in your target.

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "selection",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "Test1",
                "table-name": "%"
            },
            "rule-action": "include"
        }
    ]
}
```

```

    "rule-type": "transformation",
    "rule-id": "2",
    "rule-name": "2",
    "rule-action": "rename",
    "rule-target": "schema",
    "object-locator": {
        "schema-name": "Test"
    },
    "value": "Test1"
}
]
}

```

### Example Rename a table

The following example renames a table from Actor in your source to Actor1 in your target.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "rename",
            "rule-target": "table",
            "object-locator": {
                "schema-name": "Test",
                "table-name": "Actor"
            },
            "value": "Actor1"
        }
    ]
}

```

### Example Rename a column

The following example renames a column in table Actor from first\_name in your source to fname in your target.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "rename",
            "rule-target": "column",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "value": "first_name"
        }
    ]
}

```

```

        "rule-id": "4",
        "rule-name": "4",
        "rule-action": "rename",
        "rule-target": "column",
        "object-locator": {
            "schema-name": "test",
            "table-name": "Actor",
            "column-name" : "first_name"
        },
        "value": "fname"
    }
]
}

```

### **Example Rename an Oracle table tablespace**

The following example renames the table tablespace named SetSpace for a table named Actor in your Oracle source to SceneTblSpace in your Oracle target endpoint.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Play",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "rename",
            "rule-target": "table-tablespace",
            "object-locator": {
                "schema-name": "Play",
                "table-name": "Actor",
                "table-tablespace-name": "SetSpace"
            },
            "value": "SceneTblSpace"
        }
    ]
}

```

### **Example Rename an Oracle index tablespace**

The following example renames the index tablespace named SetISpace for a table named Actor in your Oracle source to SceneIdxSpace in your Oracle target endpoint.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "Play",
                "table-name": "%"
            },
            "rule-action": "include"
        }
    ]
}

```

```

        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "rename",
            "rule-target": "table-tablespace",
            "object-locator": {
                "schema-name": "Play",
                "table-name": "Actor",
                "table-tablespace-name": "SetISpace"
            },
            "value": "SceneIdxSpace"
        }
    ]
}

```

### Example Add a column

The following example adds a datetime column to the table Actor in schema test.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "add-column",
            "rule-target": "column",
            "object-locator": {
                "schema-name": "test",
                "table-name": "actor"
            },
            "value": "last_updated",
            "data-type": {
                "type": "datetime",
                "precision": 6
            }
        }
    ]
}

```

### Example Remove a column

The following example transforms the table named Actor in your source to remove all columns starting with the characters col from it in your target.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {

```

```

    "schema-name": "test",
    "table-name": "%"
},
"rule-action": "include"
}, {
"rule-type": "transformation",
"rule-id": "2",
"rule-name": "2",
"rule-action": "remove-column",
"rule-target": "column",
"object-locator": {
"schema-name": "test",
"table-name": "Actor",
"column-name": "col%"
}
}]
}

```

### Example Convert to lowercase

The following example converts a table name from ACTOR in your source to actor in your target.

```
{
"rules": [
{
"rule-type": "selection",
"rule-id": "1",
"rule-name": "1",
"object-locator": {
"schema-name": "test",
"table-name": "%"
},
"rule-action": "include"
}, {
"rule-type": "transformation",
"rule-id": "2",
"rule-name": "2",
"rule-action": "convert-lowercase",
"rule-target": "table",
"object-locator": {
"schema-name": "test",
"table-name": "ACTOR"
}
}
]
}
```

### Example Convert to uppercase

The following example converts all columns in all tables and all schemas from lowercase in your source to uppercase in your target.

```
{
"rules": [
{
"rule-type": "selection",
"rule-id": "1",
"rule-name": "1",
"object-locator": {
"schema-name": "test",
"table-name": "%"
},
"rule-action": "include"
}, {

```

```

    "rule-type": "transformation",
    "rule-id": "2",
    "rule-name": "2",
    "rule-action": "convert-uppercase",
    "rule-target": "column",
    "object-locator": {
        "schema-name": "%",
        "table-name": "%",
        "column-name": "%"
    }
}
]
}
}

```

### Example Add a prefix

The following example transforms all tables in your source to add the prefix DMS\_ to them in your target.

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "add-prefix",
            "rule-target": "table",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "value": "DMS_"
        }
    ]
}
```

### Example Replace a prefix

The following example transforms all columns containing the prefix Pre\_ in your source to replace the prefix with NewPre\_ in your target.

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "replace-prefix",
            "rule-target": "column",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "old-value": "Pre_",
            "new-value": "NewPre_"
        }
    ]
}
```

```

    "rule-name": "2",
    "rule-action": "replace-prefix",
    "rule-target": "column",
    "object-locator": {
        "schema-name": "%",
        "table-name": "%",
        "column-name": "%"
    },
    "value": "NewPre_",
    "old-value": "Pre_"
}
]
}

```

### Example Remove a suffix

The following example transforms all tables in your source to remove the suffix \_DMS from them in your target.

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "test",
        "table-name": "%"
      },
      "rule-action": "include"
    },
    {
      "rule-type": "transformation",
      "rule-id": "2",
      "rule-name": "2",
      "rule-action": "remove-suffix",
      "rule-target": "table",
      "object-locator": {
        "schema-name": "test",
        "table-name": "%"
      },
      "value": "_DMS"
    }
  ]
}
```

### Example Define a primary key

The following example defines a primary key named ITEM-primary-key on three columns of the ITEM table migrated to your target endpoint.

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "inventory",
        "table-name": "%"
      },
      "rule-action": "include"
    },
    {
      "rule-type": "transformation",
      "rule-id": "2",
      "rule-name": "2",
      "rule-action": "define-primary-key",
      "columns": [
        "id",
        "name",
        "category"
      ]
    }
  ]
}
```

```

    "rule-target": "table",
    "object-locator": {
        "schema-name": "inventory",
        "table-name": "ITEM"
    },
    "primary-key-def": {
        "name": "ITEM-primary-key",
        "columns": [
            "ITEM-NAME",
            "BOM-MODEL-NUM",
            "BOM-PART-NUM"
        ]
    }
}
]
}

```

### Example Define a unique index

The following example defines a unique index named ITEM-unique-idx on three columns of the ITEM table migrated to your target endpoint.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "inventory",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "transformation",
            "rule-id": "2",
            "rule-name": "2",
            "rule-action": "define-primary-key",
            "rule-target": "table",
            "object-locator": {
                "schema-name": "inventory",
                "table-name": "ITEM"
            },
            "primary-key-def": {
                "name": "ITEM-unique-idx",
                "origin": "unique-index",
                "columns": [
                    "ITEM-NAME",
                    "BOM-MODEL-NUM",
                    "BOM-PART-NUM"
                ]
            }
        }
    ]
}

```

### Example Change data type of target column

The following example changes the data type of a target column named SALE\_AMOUNT from an existing data type to int8.

```

{
    "rule-type": "transformation",
    "rule-id": "1",
    "rule-name": "RuleName 1",
    "rule-action": "change-data-type",

```

```
"rule-target": "column",
"object-locator": {
    "schema-name": "dbo",
    "table-name": "dms",
    "column-name": "SALE_AMOUNT"
},
"data-type": {
    "type": "int8"
}
}
```

### Example Add a before image column

For a source column named emp\_no, the transformation rule in the example following adds a new column named BI\_emp\_no in the target.

```
{
"rules": [
{
    "rule-type": "selection",
    "rule-id": "1",
    "rule-name": "1",
    "object-locator": {
        "schema-name": "%",
        "table-name": "%"
    },
    "rule-action": "include"
},
{
    "rule-type": "transformation",
    "rule-id": "2",
    "rule-name": "2",
    "rule-target": "column",
    "object-locator": {
        "schema-name": "%",
        "table-name": "employees"
    },
    "rule-action": "add-before-image-columns",
    "before-image-def": {
        "column-prefix": "BI_",
        "column-suffix": "",
        "column-filter": "pk-only"
    }
}
]
```

Here, the following statement populates a BI\_emp\_no column in the corresponding row with 1.

```
UPDATE employees SET emp_no = 3 WHERE emp_no = 1;
```

When writing CDC updates to supported AWS DMS targets, the BI\_emp\_no column makes it possible to tell which rows have updated values in the emp\_no column.

## Using transformation rule expressions to define column content

To define content for new and existing columns, you can use an expression within a transformation rule. For example, using expressions you can add a column or replicate source table headers to a target. You can also use expressions to flag records on target tables as inserted, updated, or deleted at the source.

## Topics

- [Adding a column using an expression \(p. 514\)](#)
- [Flagging target records using an expression \(p. 514\)](#)
- [Replicating source table headers using expressions \(p. 515\)](#)
- [Using SQLite functions to build expressions \(p. 517\)](#)
- [Adding metadata to a target table using expressions \(p. 522\)](#)

## Adding a column using an expression

To add columns to tables using an expression in a transformation rule, use an add-column rule action and a column rule target.

The following example adds a new column to the ITEM table. It sets the new column name to FULL\_NAME, with a data type of string, 50 characters long. The expression concatenates the values of two existing columns, FIRST\_NAME and LAST\_NAME, to evaluate to FULL\_NAME. The schema-name, table-name, and expression parameters refer to objects in the source database table. Value and the data-type block refer to objects in the target database table.

```
{  
    "rules": [  
        {  
            "rule-type": "selection",  
            "rule-id": "1",  
            "rule-name": "1",  
            "object-locator": {  
                "schema-name": "Test",  
                "table-name": "%"  
            },  
            "rule-action": "include"  
        },  
        {  
            "rule-type": "transformation",  
            "rule-id": "2",  
            "rule-name": "2",  
            "rule-action": "add-column",  
            "rule-target": "column",  
            "object-locator": {  
                "schema-name": "Test",  
                "table-name": "ITEM"  
            },  
            "value": "FULL_NAME",  
            "expression": "$FIRST_NAME||'_'||$LAST_NAME",  
            "data-type": {  
                "type": "string",  
                "length": 50  
            }  
        }  
    ]  
}
```

## Flagging target records using an expression

To flag records in target tables as inserted, updated, or deleted in the source table, use an expression in a transformation rule. The expression uses an operation\_indicator function to flag records. Records deleted from the source aren't deleted from the target. Instead, the target record is flagged with a user-provided value to indicate that it was deleted from the source.

**Note**

The `operation_indicator` function works only on tables that have a primary key on both source and target database.

For example, the following transformation rule first adds a new `Operation` column to a target table. It then updates the column with the value D whenever a record is deleted from a source table.

```
{
    "rule-type": "transformation",
    "rule-id": "2",
    "rule-name": "2",
    "rule-target": "column",
    "object-locator": {
        "schema-name": "%",
        "table-name": "%"
    },
    "rule-action": "add-column",
    "value": "Operation",
    "expression": "operation_indicator('D', 'U', 'I')",
    "data-type": {
        "type": "string",
        "length": 50
    }
}
```

## Replicating source table headers using expressions

By default, headers for source tables aren't replicated to the target. To indicate which headers to replicate, use a transformation rule with an expression that includes the table column header.

You can use the following column headers in expressions.

Header	Value in ongoing replication	Value in full load	Data type
AR_H_STREAM_POSITION	The stream position value from the source. This value might be the system change number (SCN) or the log sequence number (LSN), depending on the source endpoint.	An empty string.	STRING
AR_H_TIMESTAMP	A timestamp indicating the time of the change.	A timestamp indicating the current time data arrives at the target.	DATETIME (scale=7)
AR_H_COMMIT_TIMESTAMP	A timestamp indicating the time of the commit.	A timestamp indicating the current time.	DATETIME (scale=7)
AR_H_OPERATION	INSERT, UPDATE, or DELETE	INSERT	STRING
AR_H_USER	The user name, ID, or any other information that the source provides about the user that made the change.	The transformation that you want to apply to the object. Transformation rule	STRING

Header	Value in ongoing replication	Value in full load	Data type
	This header is supported on the SQL Server and Oracle (version 11.2.0.3 and higher) source endpoints only.	actions are case-sensitive.	
AR_H_CHANGE_SEQ	A unique incrementing number from the source database that consists of a timestamp and an auto incrementing number. The value depends on the source database system.	An empty string.	STRING

The following example adds a new column to the target by using the stream position value from the source. For SQL Server, the stream position value is the LSN for the source endpoint. For Oracle, the stream position value is the SCN for the source endpoint.

```
{
    "rule-type": "transformation",
    "rule-id": "2",
    "rule-name": "2",
    "rule-target": "column",
    "object-locator": {
        "schema-name": "%",
        "table-name": "%"
    },
    "rule-action": "add-column",
    "value": "transact_id",
    "expression": "$AR_H_STREAM_POSITION",
    "data-type": {
        "type": "string",
        "length": 50
    }
}
```

The following example adds a new column to the target that has a unique incrementing number from the source. This value represents a 35 digit unique number at task level. The first 16 digits are part of a timestamp, and the last 19 digits are the record\_id number incremented by the DBMS.

```
{
    "rule-type": "transformation",
    "rule-id": "2",
    "rule-name": "2",
    "rule-target": "column",
    "object-locator": {
        "schema-name": "%",
        "table-name": "%"
    },
    "rule-action": "add-column",
    "value": "transact_id",
    "expression": "$AR_H_CHANGE_SEQ",
    "data-type": {
        "type": "string",
        "length": 50
    }
}
```

```
}
```

## Using SQLite functions to build expressions

You use table settings to specify any settings that you want to apply to the selected table or view for a specified operation. Table-settings rules are optional.

### Note

Instead of the concept of tables and views, MongoDB and DocumentDB databases store data records as documents that are gathered together in *collections*. So then, when migrating from a MongoDB or DocumentDB source, consider the range segmentation type of parallel load settings for selected *collections* rather than tables and views.

### Topics

- [Using a CASE expression \(p. 520\)](#)
- [Examples \(p. 521\)](#)

Following, you can find string functions that you can use to build transformation rule expressions.

String functions	Description
<code>lower(x)</code>	The <code>lower(x)</code> function returns a copy of string <code>x</code> with all characters converted to lowercase. The default, built-in <code>lower</code> function works for ASCII characters only.
<code>ltrim(x,y)</code>	The <code>ltrim(x,y)</code> function returns a string formed by removing all characters that appear in <code>y</code> from the left side of <code>x</code> . If there is no value for <code>y</code> , <code>ltrim(x)</code> removes spaces from the left side of <code>x</code> .
<code>replace(x,y,z)</code>	The <code>replace(x,y,z)</code> function returns a string formed by substituting string <code>z</code> for every occurrence of string <code>y</code> in string <code>x</code> .
<code>rtrim(x,y)</code>	The <code>rtrim(x,y)</code> function returns a string formed by removing all characters that appear in <code>y</code> from the right side of <code>x</code> . If there is no value for <code>y</code> , <code>rtrim(x)</code> removes spaces from the right side of <code>x</code> .
<code>substr(x,y,z)</code>	The <code>substr(x,y,z)</code> function returns a substring of the input string <code>x</code> that begins with the <code>y</code> th character, and which is <code>z</code> characters long.  If <code>z</code> is omitted, <code>substr(x,y)</code> returns all characters through the end of string <code>x</code> beginning with the <code>y</code> th character. The leftmost character of <code>x</code> is number 1. If <code>y</code> is negative, the first character of the substring is found by counting from the right rather than the left. If <code>z</code> is negative, then the <code>abs(z)</code> characters preceding the <code>y</code> th character are returned. If <code>x</code> is a string, then the characters' indices refer to actual UTF-8 characters. If <code>x</code> is a BLOB, then the indices refer to bytes.
<code>trim(x,y)</code>	The <code>trim(x,y)</code> function returns a string formed by removing all characters that appear in <code>y</code> from both sides of <code>x</code> . If there is no value for <code>y</code> , <code>trim(x)</code> removes spaces from both sides of <code>x</code> .
<code>replaceChars(X,Y,Z)</code>	The <code>replaceChars(X,Y,Z)</code> function replaces any character in string <code>X</code> that also exists in string <code>Y</code> (characters to be replaced) with

String functions	Description
	<p><b>Z</b> (replacement characters) in the same position. This function is especially useful for removing characters that aren't valid from paths and file names.</p> <p>If string <b>Z</b> doesn't include a character that has a corresponding position in string <b>X</b>, that string <b>X</b> character is replaced with the first character in string <b>Z</b>.</p> <p>If string <b>X</b> includes a character that doesn't exist in string <b>Z</b>, the original character is kept unchanged.</p> <p>For example, specifying <code>replaceChars("abcde", "abcd", "123")</code> returns <b>1231e</b>.</p>

Following, you can find LOB functions that you can use to build transformation rule expressions.

LOB functions	Description
<code>hex(x)</code>	The <code>hex</code> function receives a BLOB as an argument and returns an uppercase hexadecimal string version of the BLOB content.
<code>randomblob(N)</code>	The <code>randomblob(N)</code> function returns an <b>N</b> -byte BLOB that contains pseudorandom bytes. If <b>N</b> is less than 1, a 1-byte random BLOB is returned.
<code>zeroblob(N)</code>	The <code>zeroblob(N)</code> function returns a BLOB that consists of <b>N</b> bytes of 0x00.

Following, you can find numeric functions that you can use to build transformation rule expressions.

Numeric functions	Description
<code>abs(x)</code>	The <code>abs(x)</code> function returns the absolute value of the numeric argument <b>x</b> . The <code>abs(x)</code> function returns NULL if <b>x</b> is NULL. The <code>abs(x)</code> function returns 0.0 if <b>x</b> is a string or BLOB that can't be converted to a numeric value.
<code>random()</code>	The <code>random</code> function returns a pseudorandom integer between -9,223,372,036,854,775,808 and +9,223,372,036,854,775,807.
<code>round(x,y)</code>	The <code>round(x,y)</code> function returns a floating-point value <b>x</b> rounded to <b>y</b> digits to the right of the decimal point. If there is no value for <b>y</b> , it's assumed to be 0.
<code>max(x,y...)</code>	<p>The multiargument <code>max</code> function returns the argument with the maximum value, or returns NULL if any argument is NULL.</p> <p>The <code>max</code> function searches its arguments from left to right for an argument that defines a collating function. If one is found, it uses that collating function for all string comparisons. If none of the arguments to <code>max</code> define a collating function, the BINARY collating function is used. The <code>max</code> function is a simple function when it has two or more arguments, but it operates as an aggregate function if it has a single argument.</p>

Numeric functions	Description
<code>min (x,y...)</code>	<p>The multiargument <code>min</code> function returns the argument with the minimum value.</p> <p>The <code>min</code> function searches its arguments from left to right for an argument that defines a collating function. If one is found, it uses that collating function for all string comparisons. If none of the arguments to <code>min</code> define a collating function, the BINARY collating function is used. The <code>min</code> function is a simple function when it has two or more arguments, but it operates as an aggregate function if it has a single argument.</p>

Following, you can find NULL check functions that you can use to build transformation rule expressions.

NULL check functions	Description
<code>coalesce (x,y...)</code>	The <code>coalesce</code> function returns a copy of its first non-NULL argument, but it returns NULL if all arguments are NULL. The <code>coalesce</code> function has at least two arguments.
<code>ifnull(x,y)</code>	The <code>ifnull</code> function returns a copy of its first non-NULL argument, but it returns NULL if both arguments are NULL. The <code>ifnull</code> function has exactly two arguments. The <code>ifnull</code> function is the same as <code>coalesce</code> with two arguments.
<code>nullif(x,y)</code>	<p>The <code>nullif(x,y)</code> function returns a copy of its first argument if the arguments are different, but it returns NULL if the arguments are the same.</p> <p>The <code>nullif(x,y)</code> function searches its arguments from left to right for an argument that defines a collating function. If one is found, it uses that collating function for all string comparisons. If neither argument to <code>nullif</code> defines a collating function, then the BINARY collating function is used.</p>

Following, you can find date and time functions that you can use to build transformation rule expressions.

Date and time functions	Description
<code>date(<i>timestring, modifier, modifier...</i>)</code>	The <code>date</code> function returns the date in the format YYYY-MM-DD.
<code>time(<i>timestring, modifier, modifier...</i>)</code>	The <code>time</code> function returns the time in the format HH:MM:SS.
<code>datetime(<i>timestring, modifier, modifier...</i>)</code>	The <code>datetime</code> function returns the date and time in the format YYYY-MM-DD HH:MM:SS.
<code>julianday(<i>timestring, modifier, modifier...</i>)</code>	The <code>julianday</code> function returns the number of days since noon in Greenwich on November 24, 4714 B.C.
<code>strftime(<i>format, timestring, modifier, modifier...</i>)</code>	The <code>strftime</code> function returns the date according to the format string specified as the first argument, using one of the following variables:

Date and time functions	Description
	%d: day of month
	%H: hour 00–24
	%f: ** fractional seconds SS.SSS
	%j: day of year 001–366
	%J: ** Julian day number
	%m: month 01–12
	%M: minute 00–59
	%s: seconds since 1970-01-01
	%S: seconds 00–59
	%w: day of week 0–6 sunday==0
	%W: week of year 00–53
	%Y: year 0000–9999
	%%: %

Following, you can find a hash function that you can use to build transformation rule expressions.

Hash function	Description
hash_sha256( <i>x</i> )	<p>The hash function generates a hash value for an input column (using the SHA-256 algorithm) and returns the hexadecimal value of the generated hash value.</p> <p>To use the hash function in an expression, add hash_sha256(<i>x</i>) to the expression and replace <i>x</i> with the source column name.</p>

## Using a CASE expression

The SQLite CASE expression evaluates a list of conditions and returns an expression based on the result. Syntax is shown following.

```

CASE case_expression
  WHEN when_expression_1 THEN result_1
  WHEN when_expression_2 THEN result_2
  ...
  [ ELSE result_else ]
END

# Or

CASE
  WHEN case_expression THEN result_1
  WHEN case_expression THEN result_2
  ...
  [ ELSE result_else ]

```

END

## Examples

### Example of adding a new string column to the target table using a case condition

The following example transformation rule adds a new string column, emp\_seniority, to the target table, employee. It uses the SQLite round function on the salary column, with a case condition to check if the salary equals or exceeds 20,000. If it does, the column gets the value SENIOR, and anything else has the value JUNIOR.

```
{  
    "rule-type": "transformation",  
    "rule-id": "2",  
    "rule-name": "2",  
    "rule-action": "add-column",  
    "rule-target": "column",  
    "object-locator": {  
        "schema-name": "public",  
        "table-name": "employee"  
    },  
    "value": "emp_seniority",  
    "expression": " CASE WHEN round($emp_salary)>=20000 THEN 'SENIOR' ELSE 'JUNIOR' END",  
    "data-type": {  
        "type": "string",  
        "length": 50  
    }  
}
```

### Example of adding a new date column to the target table

The following example adds a new date column, createdate, to the target table, employee. When you use the SQLite date function datetime, the date is added to the newly created table for each row inserted.

```
{  
    "rule-type": "transformation",  
    "rule-id": "2",  
    "rule-name": "2",  
    "rule-action": "add-column",  
    "rule-target": "column",  
    "object-locator": {  
        "schema-name": "public",  
        "table-name": "employee"  
    },  
    "value": "createdate",  
    "expression": "datetime ()",  
    "data-type": {  
        "type": "datetime",  
        "precision": 6  
    }  
}
```

### Example of adding a new numeric column to the target table

The following example adds a new numeric column, rounded\_emp\_salary, to the target table, employee. It uses the SQLite round function to add the rounded salary.

```
{  
    "rule-type": "transformation",  
    "rule-id": "2",  
    "rule-name": "2",  
    "rule-action": "add-column",  
    "rule-target": "column",  
    "object-locator": {  
        "schema-name": "public",  
        "table-name": "employee"  
    },  
    "value": "rounded_emp_salary",  
    "expression": "round($emp_salary)",  
    "data-type": {  
        "type": "int8"  
    }  
}
```

### Example of adding a new string column to the target table using the hash function

The following example adds a new string column, hashed\_emp\_number, to the target table, employee. The SQLite hash\_sha256(~~x~~) function creates hashed values on the target for the source column, emp\_number.

```
{  
    "rule-type": "transformation",  
    "rule-id": "2",  
    "rule-name": "2",  
    "rule-action": "add-column",  
    "rule-target": "column",  
    "object-locator": {  
        "schema-name": "public",  
        "table-name": "employee"  
    },  
    "value": "hashed_emp_number",  
    "expression": "hash_sha256($emp_number)",  
    "data-type": {  
        "type": "string",  
        "length": 50  
    }  
}
```

## Adding metadata to a target table using expressions

You can add the metadata information to the target table by using the expressions following:

- \$AR\_M\_SOURCE\_SCHEMA – The name of the source schema.
- \$AR\_M\_SOURCE\_TABLE\_NAME – The name of the source table.
- \$AR\_M\_SOURCE\_COLUMN\_NAME – The name of a column in the source table.
- \$AR\_M\_SOURCE\_COLUMN\_DATATYPE – The data type of a column in the source table.

### Example of adding a column for a schema name using the schema name from the source

The example following adds a new column named schema\_name to the target by using the schema name from the source.

```
{  
    "rule-type": "transformation",
```

```

    "rule-id": "2",
    "rule-name": "2",
    "rule-action": "add-column",
    "rule-target": "column",
    "object-locator": {
        "schema-name": "%",
        "table-name": "%"
    },
    "rule-action": "add-column",
    "value": "schema_name",
    "expression": "$AR_M_SOURCE_SCHEMA",
    "data-type": {
        "type": "string",
        "length": 50
    }
}

```

## Table and collection settings rules and operations

Use table settings to specify any settings that you want to apply to a selected table or view for a specified operation. Table-settings rules are optional, depending on your endpoint and migration requirements.

Instead of using tables and views, MongoDB and Amazon DocumentDB databases store data records as documents that are gathered together in *collections*. A single database for any MongoDB or Amazon DocumentDB endpoint is a specific set of collections identified by the database name.

When migrating from a MongoDB or Amazon DocumentDB source, you work with parallel load settings slightly differently. In this case, consider the autosegmentation or range segmentation type of parallel load settings for selected collections rather than tables and views.

### Topics

- [Wildcards in table-settings are restricted \(p. 530\)](#)
- [Using parallel load for selected tables, views, and collections \(p. 530\)](#)
- [Specifying LOB settings for a selected table or view \(p. 535\)](#)
- [Table-settings examples \(p. 536\)](#)

For table-mapping rules that use the table-settings rule type, you can apply the following parameters.

Parameter	Possible values	Description
rule-type	table-settings	A value that applies the rule to a table, view, or collection specified by the selection rule.
rule-id	A numeric value.	A unique numeric value to identify the rule.
rule-name	An alphanumeric value.	A unique name to identify the rule.
object-locator	An object with the following parameters: <ul style="list-style-type: none"> <li>• schema-name – The name of the schema. For MongoDB and Amazon</li> </ul>	The name of a specific schema and table or view or the name of a specific database and collection (no wildcards).

Parameter	Possible values	Description
	<p>DocumentDB endpoints, this is the name of the database holding a set of collections.</p> <ul style="list-style-type: none"><li>• <code>table-name</code> – The name of the table, view, or collection.</li></ul>	

Parameter	Possible values	Description
parallel-load	<p>An object with the following parameters:</p> <ul style="list-style-type: none"> <li>• <b>type</b> – Specifies whether parallel loading is turned on.</li> </ul> <p>If it is, this parameter also specifies the mechanism to identify the table or view partitions, subpartitions, or other segments to load in parallel. Partitions are segments that are already defined and identified by name in the source table or view.</p> <p>For MongoDB and Amazon DocumentDB endpoints, partitions are segments. AWS DMS can calculate these automatically given associated autosegmentation parameters. Or you can specify these manually using range segmentation parameters.</p> <p>For Oracle endpoints only, subpartitions are an additional level of segments that are already defined and identified by name in the source table or view. You can identify other segments in the table-settings rule by specifying boundaries in the range of values for one or more table or view columns.</p> <ul style="list-style-type: none"> <li>• <b>partitions</b> – When type is <b>partitions-list</b>, this value specifies all the partitions to load in parallel.</li> <li>• <b>subpartitions</b> – For Oracle endpoints only, when type is <b>partitions-list</b> this value specifies all the subpartitions to load in parallel.</li> <li>• <b>columns</b> – When type is <b>ranges</b>, this value specifies the names of columns used to identify range-based segments to load in parallel.</li> <li>• <b>boundaries</b> – When type is <b>ranges</b>, this value specifies the values of the columns used to identify range-based segments to load in parallel.</li> </ul>	<p>A value that specifies a parallel load (multithreaded) operation on the table or view identified by the <code>object-locator</code> option. In this case, you can load in parallel in any of these ways:</p> <ul style="list-style-type: none"> <li>• By segments specified by all available partitions or subpartitions.</li> <li>• By selected partitions and subpartitions.</li> <li>• By autosegmentation or range-based segments that you specify.</li> </ul> <p>For more information about parallel load, see <a href="#">Using parallel load for selected tables, views, and collections (p. 530)</a>.</p>

Parameter	Possible values	Description
type	<p>One of the following for parallel-load:</p> <ul style="list-style-type: none"> <li>• <b>partitions-auto</b> – All partitions of the table or view are loaded in parallel. Every partition is allocated to its own thread.</li> </ul> <p>This is a required setting for MongoDB and Amazon DocumentDB source endpoints to use the autosegmentation option of a parallel full load.</p> <ul style="list-style-type: none"> <li>• <b>subpartitions-auto</b> – (Oracle endpoints only) All subpartitions of the table or view are loaded in parallel. Every subpartition is allocated to its own thread.</li> <li>• <b>partitions-list</b> – All specified partitions of the table or view are loaded in parallel. For Oracle endpoints only, all specified subpartitions of the table or view are loaded in parallel. Each partition and subpartition that you specify is allocated to its own thread. You specify the partitions and subpartitions to load in parallel by partition names (<b>partitions</b>) and subpartition names (<b>subpartitions</b>).</li> <li>• <b>ranges</b> – All range-specified segments of the table, view, or collection are loaded in parallel. Each table, view, or collection segment that you identify is allocated to its own thread. You specify these segments by column names (<b>columns</b>) and column values (<b>boundaries</b>).</li> </ul> <p>PostgreSQL endpoints support only this type of a parallel load. MongoDB and Amazon DocumentDB as a source endpoints support both this range segmentation type and the autosegmentation type of a parallel full load (<b>partitions-auto</b>).</p> <ul style="list-style-type: none"> <li>• <b>none</b> – The table, view, or collection is loaded in a single-threaded task (the default), regardless of its partitions or subpartitions. For more information, see <a href="#">Creating a task (p. 442)</a>.</li> </ul>	The mechanism to identify the table, view, or collection partitions, subpartitions, or segments to load in parallel.

Parameter	Possible values	Description
number-of-partitions	(Optional) When type is partitions-auto for specified collections of a MongoDB or Amazon DocumentDB endpoint, this parameter specifies the total number of partitions (segments) used for migration. The default is 16.	Specifies the exact number of partitions to load in parallel.
collection-count-from-metadata	(Optional) When type is partitions-auto for specified collections of a MongoDB or Amazon DocumentDB endpoint and this parameter is set to true, AWS DMS uses an estimated collection count for determining the number of partitions. If this parameter is set to false, AWS DMS uses the actual collection count. The default is true.	Specifies whether to use an estimate collection count or the actual collection count to calculate the number of partitions to load in parallel.
max-records-skip-per-page	(Optional) When type is partitions-auto for specified collections of a MongoDB or Amazon DocumentDB endpoint, this is the number of records to skip at once when determining the boundaries for each partition. AWS DMS uses a paginated skip approach to determine the minimum boundary for a partition. The default is 10,000.	Specifies the number of records to skip at once when determining the boundaries for each partition. Setting a relatively large value from the default might result in cursor timeouts and task failures. Setting a relatively low value from the default results in more operations per page and a slower full load.
batch-size	(Optional) When type is partitions-auto for specified collections of a MongoDB or Amazon DocumentDB endpoint, this integer value limits the number of documents returned in one round-trip batch. If the batch size is zero (0), the cursor uses the server-defined maximum batch size. The default is 0.	Specifies the maximum number of documents returned in one batch. Each batch requires a round trip to the server.
partitions	When type is partitions-list, this is an array of strings that specify the names of partitions to load in parallel.	The names of partitions to load in parallel.
subpartitions	(Oracle endpoints only) When type is partitions-list, this is an array of strings that specifies the names of subpartitions to load in parallel.	The names of subpartitions to load in parallel.
columns	When type is ranges, an array of strings set to the names of columns that identify range-based table, view, or collection segments to load in parallel.	The names of columns that identify range-based table, view, or collection segments to load in parallel.

Parameter	Possible values	Description
boundaries	When type is ranges, an array of column-value arrays. Each column-value array contains column values in the quantity and order specified by columns. A column-value array specifies the upper boundary of a table, view, or collection segment. Each additional column-value array adds the upper boundary for one additional table, view, or collection segment. All such range-based table, view, or collection segments load in parallel.	Column values that identify range-based table, view, or collection partitions to load in parallel.
lob-settings	An object with the following parameters: <ul style="list-style-type: none"> <li>• mode – Specifies the migration handling mode for LOBs.</li> <li>• bulk-max-size – Specifies the maximum size of LOBs, depending on the mode setting.</li> </ul>	A value that specifies LOB handling for the table or view identified by the object-locator option. The specified LOB handling overrides any task LOB settings for this table or view only. For more information about using the LOB settings parameters, see <a href="#">Specifying LOB settings for a selected table or view (p. 535)</a> .

Parameter	Possible values	Description
mode	<p>Specifies the migration handling for LOBs in the specified table or view using the following values:</p> <ul style="list-style-type: none"> <li>• limited – (Default) This value sets migration to limited LOB mode, with all LOBs migrated inline together with all other column data types in the table or view. Use this value when replicating mostly small LOBs (100 MB or less). Also, specify a bulk-max-size value (zero is invalid). All migrated LOBs greater than bulk-max-size are truncated to the size that you set.</li> <li>• unlimited – This value sets migration to full LOB mode. Use this value when all or most of the LOBs that you want to replicate are larger than 1 GB. If you specify a bulk-max-size value of zero, all LOBs are migrated in <i>standard</i> full LOB mode. In this form of unlimited mode, all LOBs are migrated separately from other column data types using a lookup from the source table or view. If you specify a bulk-max-size value greater than zero, all LOBs are migrated in <i>combination</i> full LOB mode. In this form of unlimited mode, LOBs greater than bulk-max-size are migrated using a source table or view lookup, similar to standard full LOB mode. Otherwise, LOBs up to and including this size are migrated inline, similar to limited LOB mode. No LOB is ever truncated in unlimited mode, regardless of the form you use.</li> <li>• none – All table or view LOBs are migrated according to the task LOB settings.</li> </ul> <p>For more information about the task LOB settings, see <a href="#">Target metadata task settings (p. 452)</a>.</p> <p>For more information about how to migrate LOBs and how to specify these task LOB settings, see <a href="#">Setting LOB support for source databases in an AWS DMS task (p. 479)</a>.</p>	The mechanism used to migrate LOBs.

Parameter	Possible values	Description
bulk-max-size	The effect of this value depends on the mode.	The maximum size of LOBs in kilobyte increments. Specify this option only if you need to replicate small LOBs or if the target endpoint doesn't support unlimited LOB size.

## Wildcards in table-settings are restricted

Using the percent wildcard ("%"") in "table-settings" rules is not supported for source databases as shown following.

```
{
    "rule-type": "table-settings",
    "rule-id": "8",
    "rule-name": "8",
    "object-locator": {
        "schema-name": "ipipeline-prod",
        "table-name": "%"
    },
    "parallel-load": {
        "type": "partitions-auto",
        "number-of-partitions": 16,
        "collection-count-from-metadata": "true",
        "max-records-skip-per-page": 1000000,
        "batch-size": 50000
    }
}
```

If you use "%" in the "table-settings" rules as shown, AWS DMS returns the exception following.

Error in mapping rules. Rule with ruleId = x failed validation. Exact schema and table name required when using table settings rule.

In addition, AWS recommends that you don't load a great number of large collections using a single task with parallel-load. Note that AWS DMS limits resource contention as well as the number of segments loaded in parallel by the value of the MaxFullLoadSubTasks task settings parameter, with a maximum value of 49.

Instead, specify all collections for your source database for the largest collections by specifying each "schema-name" and "table-name" individually. Also, scale up your migration properly. For example, run multiple tasks across a sufficient number of replication instances to handle a great number of large collections in your database.

## Using parallel load for selected tables, views, and collections

To speed up migration and make it more efficient, you can use parallel load for selected relational tables, views, and collections. In other words, you can migrate a single segmented table, view, or collection using several threads in parallel. To do this, AWS DMS splits a full-load task into threads, with each table segment allocated to its own thread.

Using this parallel-load process, you can first have multiple threads unload multiple tables, views, and collections in parallel from the source endpoint. You can then have multiple threads migrate and load the same tables, views, and collections in parallel to the target endpoint. For some database engines, you can segment the tables and views by existing partitions or subpartitions. For other database

engines, you can have AWS DMS automatically segment collections according to specific parameters (autosegmentation). Otherwise, you can segment any table, view, or collection by ranges of column values that you specify.

Parallel load is supported for the following source endpoints:

- Oracle
- Microsoft SQL Server
- MySQL
- PostgreSQL
- IBM Db2
- SAP Adaptive Server Enterprise (ASE)
- MongoDB (only supports the autosegmentation and range segmentation options of a parallel full load)
- Amazon DocumentDB (only supports the autosegmentation and range segmentation options of a parallel full load)

For MongoDB and Amazon DocumentDB endpoints, AWS DMS supports the following data types for columns that are partition keys for the range segmentation option of a parallel full load.

- Double
- String
- ObjectId
- 32 bit integer
- 64 bit integer

Parallel load for use with table-setting rules are supported for the following target endpoints:

- Oracle
- Microsoft SQL Server
- MySQL
- PostgreSQL
- Amazon S3
- SAP Adaptive Server Enterprise (ASE)
- Amazon Redshift
- MongoDB (only supports the autosegmentation and range segmentation options of a parallel full load)
- Amazon DocumentDB (only supports the autosegmentation and range segmentation options of a parallel full load)

To specify the maximum number of tables and views to load in parallel, use the `MaxFullLoadSubTasks` task setting.

To specify the maximum number of threads per table or view for the supported targets of a parallel-load task, define more segments using column-value boundaries.

To specify the maximum number of threads per table for Amazon DynamoDB, Amazon Kinesis Data Streams, Apache Kafka, or Amazon Elasticsearch Service targets, use the `ParallelLoadThreads` target metadata task setting.

To specify the buffer size for a parallel load task when `ParallelLoadThreads` is used, use the `ParallelLoadBufferSize` target metadata task setting.

The availability and settings of `ParallelLoadThreads` and `ParallelLoadBufferSize` depend on the target endpoint.

For more information about the `ParallelLoadThreads` and `ParallelLoadBufferSize` settings, see [Target metadata task settings \(p. 452\)](#). For more information about the `MaxFullLoadSubTasks` setting, see [Full-load task settings \(p. 454\)](#). For information specific to target endpoints, see the related topics.

To use parallel load, create a table-mapping rule of type `table-settings` with the `parallel-load` option. Within the `table-settings` rule, you can specify the segmentation criteria for a single table, view, or collection that you want to load in parallel. To do so, set the `type` parameter of the `parallel-load` option to one of several options.

How to do this depends on how you want to segment the table, view, or collection for parallel load:

- By partitions (or segments) – Load all existing table or view partitions (or segments) using the `partitions-auto` type. Or load only selected partitions using the `partitions-list` type with a specified `partitions` array.

For MongoDB and Amazon DocumentDB endpoints only, load all or specified collections by segments that AWS DMS automatically calculates also using the `partitions-auto` type and additional optional `table-settings` parameters.

- (Oracle endpoints only) By subpartitions – Load all existing table or view subpartitions using the `subpartitions-auto` type. Or load only selected subpartitions using the `partitions-list` type with a specified `subpartitions` array.
- By segments that you define – Load table, view, or collection segments that you define by using column-value boundaries. To do so, use the `ranges` type with specified `columns` and `boundaries` arrays.

**Note**

PostgreSQL endpoints support only this type of a parallel load. MongoDB and Amazon DocumentDB as a source endpoints support both this range segmentation type and the autosegmentation type of a parallel full load (`partitions-auto`).

To identify additional tables, views, or collections to load in parallel, specify additional `table-settings` objects with `parallel-load` options.

In the following procedures, you can find out how to code JSON for each parallel-load type, from the simplest to the most complex.

**To specify all table, view, or collection partitions, or all table or view subpartitions**

- Specify `parallel-load` with either the `partitions-auto` type or the `subpartitions-auto` type (but not both).

Every table, view, or collection partition (or segment) or subpartition is then automatically allocated to its own thread.

For some endpoints, parallel load includes partitions or subpartitions only if they are already defined for the table or view. For MongoDB and Amazon DocumentDB source endpoints, you can have AWS DMS automatically calculate the partitions (or segments) based on optional additional parameters. These include `number-of-partitions`, `collection-count-from-metadata`, `max-records-skip-per-page`, and `batch-size`.

**To specify selected table or view partitions, subpartitions, or both**

1. Specify `parallel-load` with the `partitions-list` type.

2. (Optional) Include partitions by specifying an array of partition names as the value of `partitions`.

Each specified partition is then allocated to its own thread.

**Important**

For Oracle endpoints, make sure partitions and subpartitions aren't overlapping when choosing them for parallel load. If you use overlapping partitions and subpartitions to load data in parallel, it duplicates entries, or it fails due to a primary key duplicate violation.

3. (Optional), For Oracle endpoints only, include subpartitions by specifying an array of subpartition names as the value of `subpartitions`.

Each specified subpartition is then allocated to its own thread.

**Note**

Parallel load includes partitions or subpartitions only if they are already defined for the table or view.

You can specify table or view segments as ranges of column values. When you do so, be aware of these column characteristics:

- Specifying indexed columns significantly improves performance.
- You can specify up to 10 columns.
- You can't use columns to define segment boundaries with the following AWS DMS data types: `DOUBLE`, `FLOAT`, `BLOB`, `CLOB`, and `NCLOB`
- Records with null values aren't replicated.

### To specify table, view, or collection segments as ranges of column values

1. Specify `parallel-load` with the `ranges` type.
2. Define a boundary between table or view segments by specifying an array of column names as the value of `columns`. Do this for every column for which you want to define a boundary between table or view segments.

The order of columns is significant. The first column is the most significant and the last column is least significant in defining each boundary, as described following.

3. Define the data ranges for all the table or view segments by specifying a boundary array as the value of `boundaries`. A *boundary array* is an array of column-value arrays. To do so, take the following steps:
  - a. Specify each element of a column-value array as a value that corresponds to each column. A *column-value array* represents the upper boundary of each table or view segment that you want to define. Specify each column in the same order that you specified that column in the `columns` array.

Enter values for DATE columns in the format supported by the source.

- b. Specify each column-value array as the upper boundary, in order, of each segment from the bottom to the next-to-top segment of the table or view. If any rows exist above the top boundary that you specify, these rows complete the top segment of the table or view. Thus, the number of range-based segments is potentially one more than the number of segment boundaries in the boundary array. Each such range-based segment is allocated to its own thread.

All of the non-null data is replicated, even if you don't define data ranges for all of the columns in the table or view.

For example, suppose that you define three column-value arrays for columns COL1, COL2, and COL3 as follows.

COL1	COL2	COL3
10	30	105
20	20	120
100	12	99

You have defined three segment boundaries for a possible total of four segments.

To identify the ranges of rows to replicate for each segment, the replication instance applies a search to these three columns for each of the four segments. The search is like the following:

#### Segment 1

Replicate all rows where the following is true: The first two-column values are less than or equal to their corresponding **Segment 1** upper boundary values. Also, the values of the third column are less than its **Segment 1** upper boundary value.

#### Segment 2

Replicate all rows (except **Segment 1** rows) where the following is true: The first two-column values are less than or equal to their corresponding **Segment 2** upper boundary values. Also, the values of the third column are less than its **Segment 2** upper boundary value.

#### Segment 3

Replicate all rows (except **Segment 2** rows) where the following is true: The first two-column values are less than or equal to their corresponding **Segment 3** upper boundary values. Also, the values of the third column are less than its **Segment 3** upper boundary value.

#### Segment 4

Replicate all remaining rows (except the **Segment 1, 2, and 3** rows).

In this case, the replication instance creates a WHERE clause to load each segment as follows:

#### Segment 1

```
((COL1 < 10) OR ((COL1 = 10) AND (COL2 < 30)) OR ((COL1 = 10) AND (COL2 = 30) AND (COL3 < 105)))
```

#### Segment 2

```
NOT ((COL1 < 10) OR ((COL1 = 10) AND (COL2 < 30)) OR ((COL1 = 10) AND (COL2 = 30) AND (COL3 < 105))) AND ((COL1 < 20) OR ((COL1 = 20) AND (COL2 < 20)) OR ((COL1 = 20) AND (COL2 = 20) AND (COL3 < 120)))
```

#### Segment 3

```
NOT ((COL1 < 20) OR ((COL1 = 20) AND (COL2 < 20)) OR ((COL1 = 20) AND (COL2 = 20) AND (COL3 < 120))) AND ((COL1 < 100) OR ((COL1 = 100) AND (COL2 < 12)) OR ((COL1 = 100) AND (COL2 = 12) AND (COL3 < 99)))
```

#### Segment 4

```
NOT ((COL1 < 100) OR ((COL1 = 100) AND (COL2 < 12)) OR ((COL1 = 100) AND  
(COL2 = 12) AND (COL3 < 99)))
```

## Specifying LOB settings for a selected table or view

You can set task LOB settings for one or more tables by creating a table-mapping rule of type table-settings with the lob-settings option for one or more table-settings objects.

Specifying LOB settings for selected tables or views is supported for the following source endpoints:

- Oracle
- Microsoft SQL Server
- MySQL
- PostgreSQL
- IBM Db2, depending on the mode and bulk-max-size settings, described following
- SAP Adaptive Server Enterprise (ASE), depending on the mode and bulk-max-size settings, as described following

Specifying LOB settings for selected tables or views is supported for the following target endpoints:

- Oracle
- Microsoft SQL Server
- MySQL
- PostgreSQL
- SAP ASE, depending on the mode and bulk-max-size settings, as described following

#### Note

You can use LOB data types only with tables and views that include a primary key.

To use LOB settings for a selected table or view, you create a table-mapping rule of type table-settings with the lob-settings option. Doing this specifies LOB handling for the table or view identified by the object-locator option. Within the table-settings rule, you can specify a lob-settings object with the following parameters:

- mode – Specifies the mechanism for handling LOB migration for the selected table or view as follows:
  - limited – The default limited LOB mode is the fastest and most efficient mode. Use this mode only if all of your LOBs are small (within 100 MB in size) or the target endpoint doesn't support an unlimited LOB size. Also if you use limited, all LOBs need to be within the size that you set for bulk-max-size.

In this mode for a full load task, the replication instance migrates all LOBs inline together with other column data types as part of main table or view storage. However, the instance truncates any migrated LOB larger than your bulk-max-size value to the specified size. For a change data capture (CDC) load task, the instance migrates all LOBs using a source table lookup, as in standard full LOB mode (see the following). It does so regardless of LOB size.

#### Note

You can migrate views for full-load tasks only.

- unlimited – The migration mechanism for full LOB mode depends on the value you set for bulk-max-size as follows:

- **Standard full LOB mode** – When you set `bulk-max-size` to zero, the replication instance migrates all LOBs using standard full LOB mode. This mode requires a lookup in the source table or view to migrate every LOB, regardless of size. This approach typically results in a much slower migration than for limited LOB mode. Use this mode only if all or most of your LOBs are large (1 GB or larger).
- **Combination full LOB mode** – When you set `bulk-max-size` to a nonzero value, this full LOB mode uses a combination of limited LOB mode and standard full LOB mode. That is for a full load task, if a LOB size is within your `bulk-max-size` value, the instance migrates the LOB inline as in limited LOB mode. If the LOB size is greater than this value, the instance migrates the LOB using a source table or view lookup as in standard full LOB mode. For a change data capture (CDC) load task, the instance migrates all LOBs using a source table lookup, as in standard full LOB mode (see the following). It does so regardless of LOB size.

**Note**

You can migrate views for full-load tasks only.

This mode results in a migration speed that is a compromise between the faster, limited LOB mode and the slower, standard full LOB mode. Use this mode only when you have a mix of small and large LOBs, and most of the LOBs are small.

This combination full LOB mode is available only for the following endpoints:

- IBM Db2 as source
- SAP ASE as source or target

Regardless of the mechanism you specify for unlimited mode, the instance migrates all LOBs fully, without truncation.

- none – The replication instance migrates LOBs in the selected table or view using your task LOB settings. Use this option to help compare migration results with and without LOB settings for the selected table or view.

If the specified table or view has LOBs included in the replication, you can set the `BatchApplyEnabled` task setting to `true` only when using limited LOB mode.

In some cases, you might set `BatchApplyEnabled` to `true` and `BatchApplyPreserveTransaction` to `false`. In these cases, the instance sets `BatchApplyPreserveTransaction` to `true` if the table or view has LOBs and the source and target endpoints are Oracle.

- `bulk-max-size` – Set this value to a zero or non-zero value in kilobytes, depending on the mode as described for the previous items. In limited mode, you must set a nonzero value for this parameter.

The instance converts LOBs to binary format. Therefore, to specify the largest LOB you need to replicate, multiply its size by three. For example, if your largest LOB is 2 MB, set `bulk-max-size` to 6,000 (6 MB).

## Table-settings examples

Following, you can find some examples that demonstrate the use of table settings.

### Example Load a table segmented by partitions

The following example loads a SALES table in your source more efficiently by loading it in parallel based on all its partitions.

```
{  
    "rules": [  
        {"rule-type": "selection",
```

```

    "rule-id": "1",
    "rule-name": "1",
    "object-locator": {
        "schema-name": "%",
        "table-name": "%"
    },
    "rule-action": "include"
},
{
    "rule-type": "table-settings",
    "rule-id": "2",
    "rule-name": "2",
    "object-locator": {
        "schema-name": "HR",
        "table-name": "SALES"
    },
    "parallel-load": {
        "type": "partitions-auto"
    }
}
]
}

```

### **Example Load a table segmented by subpartitions**

The following example loads a SALES table in your Oracle source more efficiently by loading it in parallel based on all its subpartitions.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "HR",
                "table-name": "SALES"
            },
            "parallel-load": {
                "type": "subpartitions-auto"
            }
        }
    ]
}

```

### **Example Load a table segmented by a list of partitions**

The following example loads a SALES table in your source by loading it in parallel by a particular list of partitions. Here, the specified partitions are named after values starting with portions of the alphabet, for example ABCD, EFGH, and so on.

```

{
    "rules": [

```

```

    "rule-type": "selection",
    "rule-id": "1",
    "rule-name": "1",
    "object-locator": {
        "schema-name": "%",
        "table-name": "%"
    },
    "rule-action": "include"
},
{
    "rule-type": "table-settings",
    "rule-id": "2",
    "rule-name": "2",
    "object-locator": {
        "schema-name": "HR",
        "table-name": "SALES"
    },
    "parallel-load": {
        "type": "partitions-list",
        "partitions": [
            "ABCD",
            "EFGH",
            "IJKL",
            "MNOP",
            "QRST",
            "UVWXYZ"
        ]
    }
}
]
}

```

### **Example Load an Oracle table segmented by a selected list of partitions and subpartitions**

The following example loads a SALES table in your Oracle source by loading it in parallel by a selected list of partitions and subpartitions. Here, the specified partitions are named after values starting with portions of the alphabet, for example ABCD, EFGH, and so on. The specified subpartitions are named after values starting with numerals, for example 01234 and 56789.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "HR",
                "table-name": "SALES"
            },
            "parallel-load": {
                "type": "partitions-list",
                "partitions": [
                    "ABCD",
                    "EFGH",
                    "IJKL",

```

```

        "MNOP",
        "QRST",
        "UVWXYZ"
    ],
    "subpartitions": [
        "01234",
        "56789"
    ]
}
]
}
]
```

### **Example Load a table segmented by ranges of column values**

The following example loads a SALES table in your source by loading it in parallel by segments specified by ranges of the SALES\_NO and REGION column values.

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "HR",
                "table-name": "SALES"
            },
            "parallel-load": {
                "type": "ranges",
                "columns": [
                    "SALES_NO",
                    "REGION"
                ],
                "boundaries": [
                    [
                        [
                            "1000",
                            "NORTH"
                        ],
                        [
                            "3000",
                            "WEST"
                        ]
                    ]
                ]
            }
        }
    ]
}
```

Here, two columns are specified for the segment ranges with the names, SALES\_NO and REGION. Two boundaries are specified with two sets of column values ([ "1000", "NORTH" ] and [ "3000", "WEST" ]).

These two boundaries thus identify the following three table segments to load in parallel:

### Segment 1

Rows with SALES\_NO less than or equal to 1,000 and REGION less than "NORTH". In other words, sales numbers up to 1,000 in the EAST region.

### Segment 2

Rows other than **Segment 1** with SALES\_NO less than or equal to 3,000 and REGION less than "WEST". In other words, sales numbers over 1,000 up to 3,000 in the NORTH and SOUTH regions.

### Segment 3

All remaining rows other than **Segment 1** and **Segment 2**. In other words, sales numbers over 3,000 in the "WEST" region.

## **Example Load two tables: One segmented by ranges and another segmented by partitions**

The following example loads a SALES table in parallel by segment boundaries that you identify. It also loads an ORDERS table in parallel by all of its partitions, as with previous examples.

```
{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "HR",
                "table-name": "SALES"
            },
            "parallel-load": {
                "type": "ranges",
                "columns": [
                    "SALES_NO",
                    "REGION"
                ],
                "boundaries": [
                    [
                        [
                            "1000",
                            "NORTH"
                        ],
                        [
                            [
                                "3000",
                                "WEST"
                            ]
                        ]
                    ]
                ]
            }
        },
        {
            "rule-type": "table-settings",
            "rule-id": "3",
            "rule-name": "3",
            "object-locator": {
                "schema-name": "HR",
                "table-name": "ORDERS"
            }
        }
    ]
}
```

```

        },
        "parallel-load": {
            "type": "partitions-auto"
        }
    }
}
```

### **Example Load a table with LOBs using limited LOB mode**

The following example loads an ITEMS table including LOBs in your source using limited LOB mode (the default) with a maximum nontruncated size of 100 MB. Any LOBs that are larger than this size are truncated to 100 MB. All LOBs are loaded inline with all other column data types.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "INV",
                "table-name": "ITEMS"
            },
            "lob-settings": {
                "bulk-max-size": "100000"
            }
        }
    ]
}
```

### **Example Load a table with LOBs using standard full LOB mode**

The following example loads an ITEMS table in your source, including all its LOBs without truncation, using standard full LOB mode. All LOBs, regardless of size, are loaded separately from other data types using a lookup for each LOB in the source table.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
```

```

        "schema-name": "INV",
        "table-name": "ITEMS"
    },
    "lob-settings": {
        "mode": "unlimited",
        "bulk-max-size": "0"
    }
}
]
}
}

```

### **Example Load a table with LOBs using combination full LOB mode**

The following example loads an ITEMS table in your source, including all its LOBs without truncation, using combination full LOB mode. All LOBs within 100 MB in size are loaded inline along with other data types, as in limited LOB mode. All LOBs over 100 MB in size are loaded separately from other data types. This separate load uses a lookup for each such LOB in the source table, as in standard full LOB mode.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "INV",
                "table-name": "ITEMS"
            },
            "lob-settings": {
                "mode": "unlimited",
                "bulk-max-size": "100000"
            }
        }
    ]
}

```

### **Example Load a table with LOBs using the task LOB settings**

The following example loads an ITEMS table in your source, including all LOBs, using its task LOB settings. The bulk-max-size setting of 100 MB is ignored and left only for a quick reset to limited or unlimited mode.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "%",
                "table-name": "%"
            },
            "rule-action": "include"
        },
        {
            "rule-type": "table-settings",
            "rule-id": "2",
            "rule-name": "2",
            "object-locator": {
                "schema-name": "INV",
                "table-name": "ITEMS"
            },
            "lob-settings": {
                "mode": "unlimited",
                "bulk-max-size": "100000"
            }
        }
    ]
}

```

```
{
    "rule-type": "table-settings",
    "rule-id": "2",
    "rule-name": "2",
    "object-locator": {
        "schema-name": "INV",
        "table-name": "ITEMS"
    },
    "lob-settings": {
        "mode": "none",
        "bulk-max-size": "100000"
    }
}
]
```

## Using source filters

You can use source filters to limit the number and type of records transferred from your source to your target. For example, you can specify that only employees with a location of headquarters are moved to the target database. Filters are part of a selection rule. You apply filters on a column of data.

Source filters must follow these constraints:

- A selection rule can have no filters or one or more filters.
- Every filter can have one or more filter conditions.
- If more than one filter is used, the list of filters is combined as if using an AND operator between the filters.
- If more than one filter condition is used within a single filter, the list of filter conditions is combined as if using an OR operator between the filter conditions.
- Filters are only applied when `rule-action = 'include'`.
- Filters require a column name and a list of filter conditions. Filter conditions must have a filter operator that is associated with either one value, two values, or no value, depending on the operator.
- You can specify no more than one filter condition within a single filter if you specify a negative operator like `noteq` (not equal to), `notbetween` (not equal to or between two values), or `notnull` (no NULL values).
- Column names, table names, view names, and schema names are case-sensitive. Oracle and Db2 should always use UPPER case.

The following limitations apply to using source filters:

- Filters don't calculate columns of right-to-left languages.
- Don't apply filters to LOB columns.
- Apply filters only to *immutable* columns, which aren't updated after creation. If source filters are applied to *mutable* columns, which can be updated after creation, adverse behavior can result.

For example, a filter to exclude or include specific rows in a column always excludes or includes the specified rows even if the rows are later changed. Suppose that you exclude or include rows 1–10 in column A, and they later change to become rows 11–20. In this case, they continue to be excluded or included even when the data is no longer the same.

Similarly, suppose that a row outside of the filter's scope is later updated (or updated and deleted), and should then be excluded or included as defined by the filter. In this case, it's replicated at the target.

## Creating source filter rules in JSON

You can create source filters using the `filters` parameter of a selection rule. The `filters` parameter specifies an array of one or more JSON objects. Each object has parameters that specify the source filter type, column name, and filter conditions. These filter conditions include one or more filter operators and filter values.

The following table shows the parameters for specifying source filtering in a `filters` object.

Parameter	Value
<code>filter-type</code>	<code>source</code>
<code>column-name</code>	A parameter with the name of the source column to which you want the filter applied. The name is case-sensitive.
<code>filter-conditions</code>	An array of one or more objects containing a <code>filter-operator</code> parameter and zero or more associated value parameters, depending on the <code>filter-operator</code> value.
<code>filter-operator</code>	A parameter with one of the following values: <ul style="list-style-type: none"> <li>• <code>lte</code> – less than or equal to one value</li> <li>• <code>ste</code> – less than or equal to one value (<code>lte</code> alias)</li> <li>• <code>gte</code> – greater than or equal to one value</li> <li>• <code>eq</code> – equal to one value</li> <li>• <code>noteq</code> – not equal to one value</li> <li>• <code>between</code> – equal to or between two values</li> <li>• <code>notbetween</code> – not equal to or between two values</li> <li>• <code>null</code> – NULL values</li> <li>• <code>notnull</code> – no NULL values</li> </ul>
<code>value</code> or <code>start-value</code> and <code>end-value</code> or <code>no values</code>	Zero or more value parameters associated with <code>filter-operator</code> : <ul style="list-style-type: none"> <li>• If <code>filter-operator</code> is <code>lte</code>, <code>ste</code>, <code>gte</code>, <code>eq</code>, or <code>noteq</code>, use <code>value</code> to specify one value parameter.</li> <li>• If <code>filter-operator</code> is <code>between</code> or <code>notbetween</code>, use <code>start-value</code> and <code>end-value</code> to specify two value parameters.</li> <li>• If <code>filter-operator</code> is <code>null</code> or <code>notnull</code>, specify no value parameters.</li> </ul>

The following examples show some common ways to use source filters.

### Example Single filter

The following filter replicates all employees where `empid >= 100` to the target database.

```
{
  "rules": [
    {
      "rule-type": "selection",
      "rule-id": "1",
      "rule-name": "1",
      "object-locator": {
        "schema-name": "test",
        "table-name": "employees"
      },
      "filter": [
        {
          "operator": "gte",
          "value": 100
        }
      ]
    }
  ]
}
```

```

        "table-name": "employee"
    },
    "rule-action": "include",
    "filters": [
        {
            "filter-type": "source",
            "column-name": "empid",
            "filter-conditions": [
                {
                    "filter-operator": "gte",
                    "value": "100"
                }
            ]
        }
    ]
}

```

### Example Multiple filter operators

The following filter applies multiple filter operators to a single column of data. The filter replicates all employees where (empid <= 10) OR (empid is between 50 and 75) OR (empid >= 100) to the target database.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "employee"
            },
            "rule-action": "include",
            "filters": [
                {
                    "filter-type": "source",
                    "column-name": "empid",
                    "filter-conditions": [
                        {
                            "filter-operator": "lte",
                            "value": "10"
                        },
                        {
                            "filter-operator": "between",
                            "start-value": "50",
                            "end-value": "75"
                        },
                        {
                            "filter-operator": "gte",
                            "value": "100"
                        }
                    ]
                }
            ]
        }
    ]
}

```

### Example Multiple filters

The following filters apply multiple filters to two columns in a table. The filter replicates all employees where (empid <= 100) AND (dept = tech) to the target database.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",

```

```

"object-locator": {
    "schema-name": "test",
    "table-name": "employee"
},
"rule-action": "include",
"filters": [
    {
        "filter-type": "source",
        "column-name": "empid",
        "filter-conditions": [
            {
                "filter-operator": "lte",
                "value": "100"
            }
        ]
    },
    {
        "filter-type": "source",
        "column-name": "dept",
        "filter-conditions": [
            {
                "filter-operator": "eq",
                "value": "tech"
            }
        ]
    }
]
}

```

### Example Filtering NULL values

The following filter shows how to filter on empty values. It replicates all employees where dept = NULL to the target database.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "employee"
            },
            "rule-action": "include",
            "filters": [
                {
                    "filter-type": "source",
                    "column-name": "dept",
                    "filter-conditions": [
                        {
                            "filter-operator": "null"
                        }
                    ]
                }
            ]
        }
    ]
}

```

### Example Filtering using NOT operators

Some of the operators can be used in the negative form. The following filter replicates all employees where (empid is < 50) OR (empid is > 75) to the target database.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {

```

```

        "schema-name": "test",
        "table-name": "employee"
    },
    "rule-action": "include",
    "filters": [
        {
            "filter-type": "source",
            "column-name": "empid",
            "filter-conditions": [
                {
                    "filter-operator": "notbetween",
                    "start-value": "50",
                    "end-value": "75"
                }
            ]
        }
    ]
}

```

### Example Multiple filters using NOT operators

You can use only one negative filter condition within a single filter (OR operation). However, you can use multiple filters (AND operation) with the a negative filter condition. The following filter replicates all employees where (empid != 50) AND (dept is not NULL) to the target database.

```

{
    "rules": [
        {
            "rule-type": "selection",
            "rule-id": "1",
            "rule-name": "1",
            "object-locator": {
                "schema-name": "test",
                "table-name": "employee"
            },
            "rule-action": "include",
            "filters": [
                {
                    "filter-type": "source",
                    "column-name": "empid",
                    "filter-conditions": [
                        {
                            "filter-operator": "noteq",
                            "value": "50"
                        }
                    ],
                    {
                        "filter-type": "source",
                        "column-name": "dept",
                        "filter-conditions": [
                            {
                                "filter-operator": "notnull"
                            }
                        ]
                    }
                ]
            }
        }
    ]
}

```

## Filtering by time and date

When selecting data to import, you can specify a date or time as part of your filter criteria. AWS DMS uses the date format YYYY-MM-DD and the time format YYYY-MM-DD HH:MM:SS for filtering. The AWS DMS comparison functions follow the SQLite conventions. For more information about SQLite data types and date comparisons, see [Datatypes in SQLite version 3](#) in the SQLite documentation.

The following filter shows how to filter on a date. It replicates all employees where empstartdate >= January 1, 2002 to the target database.

### Example Single date filter

```
{  
    "rules": [  
        {  
            "rule-type": "selection",  
            "rule-id": "1",  
            "rule-name": "1",  
            "object-locator": {  
                "schema-name": "test",  
                "table-name": "employee"  
            },  
            "rule-action": "include",  
            "filters": [  
                {  
                    "filter-type": "source",  
                    "column-name": "empstartdate",  
                    "filter-conditions": [  
                        {  
                            "filter-operator": "gte",  
                            "value": "2002-01-01"  
                        }  
                    ]  
                }  
            ]  
        }  
    ]  
}
```

## Enabling and working with premigration assessments for a task

A premigration assessment evaluates specified components of a database migration task to help identify any problems that might prevent a migration task from running as expected. This assessment gives you a chance to identify issues before you run a new or modified task. You can then fix problems before they occur while running the migration task itself. This can avoid delays in completing a given database migration needed to repair data and your database environment.

AWS DMS provides access to two different types of premigration assessments. The first type of premigration assessment, a premigration assessment run, is a functional superset of the second type, a data type assessment. They are described in the following topics:

**Note**

If you do a premigration assessment run that includes the data type assessment, you don't need to do a data type assessment separately.

1. [Specifying, starting, and viewing premigration assessment runs \(p. 550\)](#) – A premigration assessment run specifies one or more individual assessments to run based on a new or existing migration task configuration. Each individual assessment evaluates a specific element of a supported relational source or target database depending on considerations such as the migration type, supported objects, index configuration, and other task settings, such as table mappings that identify the schemas and tables to migrate.

For example, an individual assessment might evaluate what source data types or primary key formats can and can't be migrated, possibly based on the AWS DMS engine version. You can start and view the results of the latest assessment run and view the results of all prior assessment runs for a task either using the AWS DMS Management Console or using the AWS CLI and SDKs to access the AWS DMS API. You can also view the results of prior assessment runs for a task in an Amazon S3 bucket that you have selected for AWS DMS to store these results.

**Note**

The number and types of available individual assessments can increase over time. For more information about periodic updates, see [Specifying individual assessments \(p. 550\)](#).

2. [Starting and viewing data type assessments \(p. 555\)](#) – A data type assessment returns the results of a single type of premigration assessment in a single JSON structure: the data types that might not be migrated correctly in a supported relational source database instance. This report returns the results for all problem data types found in the columns of every schema and table in the source database that is mapped for migration. You can create and view the results of the latest data type assessment using the AWS CLI and SDKs to access the AWS DMS API. You can also view the results of the latest data type assessment using the AWS DMS Management Console. You can view the results of prior data type assessments in an Amazon S3 bucket for your account where AWS DMS stores these reports.

## Storing premigration assessment runs in an S3 Bucket

The following Identity and Access Management (IAM) policy allows DMS to store preassessment results in the S3 bucket that you create.

### To access an S3 bucket for a premigration assessment

1. Create a service role using IAM and attach an IAM policy like the following to your service role. For information about creating a service role in the console, see [Creating a role for an AWS service \(console\)](#).

```
//Policy to access S3 bucket

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:PutObjectTagging"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3>ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket"
            ]
        }
    ]
}
```

2. Edit the trust relationship and attach the following IAM role to your service role, allowing DMS to assume the role.

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

```

        "Sid":"",
        "Effect":"Allow",
        "Principal":{
            "Service":"dms.amazonaws.com"
        },
        "Action":"sts:AssumeRole"
    }
]
}

```

## Specifying, starting, and viewing premigration assessment runs

A premigration assessment run specifies one or more individual assessments to run based on a new or existing migration task configuration. Each individual assessment evaluates a specific element of the source or target database depending on considerations such as the migration type, supported objects, index configuration, and other task settings, such as table mappings to identify the schemas and tables to migrate. For example, an individual assessment might evaluate what source data types or primary key formats can and cannot be migrated, possibly based on the AWS DMS engine version.

### Specifying individual assessments

The following table provides a brief overview of the individual assessments that are applicable to a given migration task based on its configuration. You can choose from among the individual assessments to include in a new assessment run that are applicable to your task configuration.

Assessment name (console and API)	Description	Source must be relational	Target must be relational	Target must be OpenSea Service	Target must be DynamoDB	Migration type must perform CDC	Applicable AWS DMS engine versions
Console – Unsupported data types  API – unsupported-data-types-in-source	Checks for data types unsupported by AWS DMS in the source endpoint. Not all data types can be migrated between engines.	–	–	–	–	–	All supported versions
Console – Large objects (LOBs) are used but target LOB columns are not nullable  API – full-lob-not-nullable-at-target	Checks for the nullability of a LOB column in the target when full LOB mode or inline LOB mode is used. AWS DMS requires a LOB column to be null when using these LOB modes.	X	X	–	–	–	All supported versions

Assessment name (console and API)	Description	Source must be relational	Target must be relational	Target must be OpenSea Service	Target must be DynamoDB	Migration type must perform CDC	Applicable AWS DMS engine versions
Console – Source table with LOBs but without primary keys or unique constraints  API – table-with-lob-but-without-primary-key-or-unique-constraint	Checks for the presence of source tables with LOBs but without a primary key or a unique key. Currently, a table must have a primary key or a unique key for AWS DMS to migrate LOBs.	X	-	-	-	-	All supported versions
Console – Source table without primary key for CDC or full load and CDC tasks only  API – table-with-no-primary-key-or-unique-constraint	Checks for the presence of a primary key or a unique key in source tables for a full-load and change data capture (CDC) migration or a CDC-only migration. Lack of a primary key or a unique key can cause performance issues during the CDC migration.	X	-	-	-	X	All supported versions
Console – Target table without primary keys for CDC tasks only  API – target-table-has-unique-key-or-primary-key-for-cdc	Checks for the presence of a primary key or a unique key in already-created target tables for a CDC-only migration. Lack of a primary key or a unique key can cause full table scans in the target when AWS DMS applies updates and deletes resulting in performance issues during the CDC migration.	-	X	-	-	X	All supported versions

Assessment name (console and API)	Description	Source must be relational	Target must be relational	Target must be OpenSearch Service	Target must be DynamoDB	Migration type must perform CDC	Applicable AWS DMS engine versions
Console – Unsupported source primary key types - composite primary keys  API – unsupported-source-pk-type-for-elasticsearch-target	Checks for the presence of composite primary keys in source tables when migrating to Amazon OpenSearch Service (OpenSearch Service). The primary key of the source table must consist of a single column.	X	–	–	X	–	All supported versions

**Note**

AWS DMS supports migrating a source database to an OpenSearch Service target where the source primary key consists of multiple columns.

AWS DMS supports premigration assessment runs for the following relational databases:

- Oracle
- SQL Server
- PostgreSQL
- MySQL
- MariaDB
- Amazon Aurora

## Starting and viewing premigration assessment runs

You can start a premigration assessment run for a new or existing migration task using the AWS DMS Management Console, the AWS CLI, and the AWS DMS API.

### To start a premigration assessment run for a new or existing task

1. From the **Database migration tasks** page in the AWS DMS Management Console, do one of the following:
  - Choose **Create task**. The **Create database migration task** page opens:
    1. Enter the task settings required to create your task, including table mapping.
    2. In the **Premigration assessment** section, select **Enable premigration assessment run**. The section expands with options to specify an assessment run for the new task.

**Note**

When creating a new task, enabling a premigration assessment run disables the option to start the task automatically on task creation. You can start the task manually after the assessment run completes.

- Choose the **Identifier** for an existing task on the **Database migration tasks** page. The task page for the chosen existing task opens:
  1. Choose **Actions** and select **Create premigration assessment**. A **Create premigration assessment** page opens with options to specify an assessment run for the existing task.

These options include:

2. Enter a unique name for your assessment run.
3. Select the available individual assessments that you want to include in this assessment run. You can only select the available individual assessments based on your current task settings. By default, all available individual assessments are enabled and selected. All other individual assessments are disabled for this assessment run.
4. Search for and choose an Amazon S3 bucket and folder in your account to store your assessment result report.
5. Select or enter an IAM role with full account access to your chosen Amazon S3 bucket and folder.
6. Optionally choose a setting to encrypt the assessment result report in your Amazon S3 bucket.
7. Choose **Create task** for a new task or choose **Create** for an existing task.

The **Database migration tasks** page opens listing your new or modified task with a **Status of Creating...** and a banner message indicating that your premigration assessment run will start once the task is created.

AWS DMS provides access to the latest and all prior premigration assessment runs using the AWS DMS Management Console, the AWS CLI, or the AWS DMS API.

#### To view results for the latest assessment run

1. From the AWS DMS Management Console, choose the **Identifier** for your existing task on the **Database migration tasks** page. The task page for the existing task opens.
2. Choose the **Premigration assessments** tab on the existing task page. This opens a **Latest assessment results** section on that page showing results of the latest assessment run for this task.

These assessment run results start with the name of the latest assessment run and an overview of its status followed by a listing of the specified individual assessments and their status. You can then explore details of the status of each individual assessment by choosing its name in the list, with results available down to the table column level.

Both the status overview for an assessment run and each individual assessment shows a **Status** value. This value indicates the overall status of the assessment run and a similar status for each individual assessment. Following is a list of the **Status** values for the assessment run:

- "cancelling" – The assessment run was cancelled.
- "deleting" – The assessment run was deleted.
- "failed" – At least one individual assessment completed with a failed status.
- "error-provisioning" – An internal error occurred while resources were provisioned (during provisioning status).
- "error-executing" – An internal error occurred while individual assessments ran (during running status).
- "invalid state" – The assessment run is in an unknown state.
- "passed" – All individual assessments have completed, and none has a failed status.
- "provisioning" – Resources required to run individual assessments are being provisioned.
- "running" – Individual assessments are being run.

- "starting" – The assessment run is starting, but resources are not yet being provisioned for individual assessments.

Following is a list of the **Status** values for each individual assessment of the assessment run:

- "cancelled" – The individual assessment was cancelled as part of cancelling the assessment run.
- "error" – The individual assessment did not complete successfully.
- "failed" – The individual assessment completed successfully with a failed validation result: view the details of the result for more information.
- "invalid state" – The individual assessment is in an unknown state.
- "passed" – The individual assessment completed with a successful validation result.
- "pending" – The individual assessment is waiting to run.
- "running" – The individual assessment is running.
- "warning" – The individual assessment completed successfully with a warning validation result: view the details of the result for more information.

You can also view the JSON files for the assessment run results on Amazon S3.

### To view the JSON files for the assessment run on Amazon S3

1. From the AWS DMS Management Console, choose the Amazon S3 bucket link shown in the status overview. This displays a list of bucket folders and other Amazon S3 objects stored in the bucket. If your results are stored in a bucket folder, open the folder.
2. You can find your assessment run results in several JSON files. A summary .json file contains the overall results of the assessment run. The remaining files are each named for an individual assessment that was specified for the assessment run, such as unsupported-data-types-in-source .json. These files each contain the results for the corresponding individual assessment from the chosen assessment run.

### To view the results for all previous assessment runs

1. Choose **Previous assessment results** under the **Latest assessment results** section. This shows a list of the previous assessment runs listed by name in reverse chronological order.
2. Choose the name of the previous assessment run whose results you want to view. The status overview and individual assessment results for your chosen assessment run display in place of the **Latest assessment results** section.
3. You can then view the results for the chosen assessment run in the same way as for the latest assessment results shown initially.

To start and view the results of premigration assessment runs for an existing migration task, you can run the following CLI commands and AWS DMS API operations:

- CLI: [describe-applicable-individual-assessments](#), API: [DescribeApplicableIndividualAssessments](#) – Provides a list of individual assessments that you can specify for a new premigration assessment run, given one or more task configuration parameters.
- CLI: [start-replication-task-assessment-run](#), API: [StartReplicationTaskAssessmentRun](#) – Starts a new premigration assessment run for one or more individual assessments of an existing migration task.
- CLI: [describe-replication-task-assessment-runs](#), API: [DescribeReplicationTaskAssessmentRuns](#) – Returns a paginated list of premigration assessment runs based on filter settings.

- CLI: [describe-replication-task-individual-assessments](#), API: [DescribeReplicationTaskIndividualAssessments](#) – Returns a paginated list of individual assessments based on filter settings.
- CLI: [cancel-replication-task-assessment-run](#), API: [CancelReplicationTaskAssessmentRun](#) – Cancels, but doesn't delete, a single premigration assessment run.
- CLI: [delete-replication-task-assessment-run](#), API: [DeleteReplicationTaskAssessmentRun](#) – Deletes the record of a single premigration assessment run.

## Starting and viewing data type assessments

A data type assessment identifies data types in a source database that might not get migrated correctly. During this assessment, AWS DMS reads the source database schemas for a migration task and creates a list of the column data types. It then compares this list to a predefined list of data types supported by AWS DMS. AWS DMS creates a report that you can look at to see if your migration task has any unsupported data types.

AWS DMS supports creating data type assessment reports for the following relational databases:

- Oracle
- SQL Server
- PostgreSQL
- MySQL
- MariaDB
- Amazon Aurora

You can start and view a data type assessment report using the CLI and SDKs to access the AWS DMS API:

- The CLI uses the [start-replication-task-assessment](#) command to start a data type assessment and uses the [describe-replication-task-assessment-results](#) command to view the latest data type assessment report in JSON format.
- The AWS DMS API uses the [StartReplicationTaskAssessment](#) operation to start a data type assessment and uses the [DescribeReplicationTaskAssessmentResults](#) operation to view the latest data type assessment report in JSON format.

You can also view the latest data type assessment report in the AWS DMS Management Console.

The data type assessment report is a single JSON file that includes a summary that lists the unsupported data types and the column count for each one. It includes a list of data structures for each unsupported data type including the schemas, tables, and columns that have the unsupported data type. You can use the report to modify the source data types and improve the migration success.

There are two levels of unsupported data types. Data types that appear on the report as not supported can't be migrated. Data types that appear on the report as partially supported might be converted to another data type, but not migrate as you expect.

The following example shows a sample data type assessment report that you might view.

```
{  
  "summary": {  
    "task-name": "test15",  
    ...  
  },  
  "data": [ ... ]  
}
```

```

"not-supported": {
    "data-type": [
        "sql-variant"
    ],
    "column-count": 3
},
"partially-supported": {
    "data-type": [
        "float8",
        "jsonb"
    ],
    "column-count": 2
}
},
"types": [
{
    "data-type": "float8",
    "support-level": "partially-supported",
    "schemas": [
        {
            "schema-name": "schema1",
            "tables": [
                {
                    "table-name": "table1",
                    "columns": [
                        "column1",
                        "column2"
                    ]
                },
                {
                    "table-name": "table2",
                    "columns": [
                        "column3",
                        "column4"
                    ]
                }
            ]
        },
        {
            "schema-name": "schema2",
            "tables": [
                {
                    "table-name": "table3",
                    "columns": [
                        "column5",
                        "column6"
                    ]
                },
                {
                    "table-name": "table4",
                    "columns": [
                        "column7",
                        "column8"
                    ]
                }
            ]
        }
    ]
},
{
    "datatype": "int8",
    "support-level": "partially-supported",
    "schemas": [
        {
            "schema-name": "schema1",
            "tables": [

```

```
{  
    "table-name": "table1",  
    "columns": [  
        "column9",  
        "column10"  
    ]  
},  
{  
    "table-name": "table2",  
    "columns": [  
        "column11",  
        "column12"  
    ]  
}  
]  
]  
]  
}
```

To view the latest data type assessment report from the AWS DMS Management Console, use the **Assessment results** tab on the task page that opens when you select the **Identifier** for a given task on the **Database migration tasks** page.

AWS DMS also stores the latest and all previous data type assessments in an Amazon S3 bucket created by AWS DMS in your account. The Amazon S3 bucket name has the following format, where *customerId* is your customer ID and *customerDNS* is an internal identifier.

dms-*customerId*-*customerDNS*

**Note**

By default, you can create up to 100 Amazon S3 buckets in each of your AWS accounts. Because AWS DMS creates a bucket in your account, make sure that it doesn't exceed your bucket limit. Otherwise, the data type assessment fails.

All data type assessment reports for a given migration task are stored in a bucket folder named with the task identifier. Each report's file name is the date of the data type assessment in the format yyyy-mm-dd-hh-mm. You can view and compare previous data type assessment reports from the Amazon S3 Management Console.

AWS DMS also creates an AWS Identity and Access Management (IAM) role to allow access to the S3 bucket created for these reports. The role name is dms-access-for-tasks. The role uses the AmazonDMSRedshiftS3Role policy.

## Specifying supplemental data for task settings

When you create or modify a replication task for some AWS DMS endpoints, the task might require additional information to perform the migration. You can specify this additional information using an option in the DMS console. Or you can specify it using the TaskData parameter for the DMS API operation `CreateReplicationTask` or `ModifyReplicationTask`.

If your target endpoint is Amazon Neptune, you need to specify mapping data, supplemental to table mapping. This supplemental mapping data specifies how to convert source relational data into the target graph data that a Neptune database can consume. In this case, you can use one of two possible formats. For more information, see [Specifying graph-mapping rules using Gremlin and R2RML for Amazon Neptune as a target \(p. 420\)](#).

# Monitoring AWS DMS tasks

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS DMS and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. AWS provides several tools for monitoring your AWS DMS tasks and resources, and responding to potential incidents:

## AWS DMS events and notifications

AWS DMS uses Amazon Simple Notification Service (Amazon SNS) to provide notifications when an AWS DMS event occurs, for example the creation or deletion of a replication instance. AWS DMS groups events into categories that you can subscribe to, so you can be notified when an event in that category occurs. For example, if you subscribe to the Creation category for a given replication instance, you are notified whenever a creation-related event occurs that affects your replication instance. You can work with these notifications in any form supported by Amazon SNS for an AWS Region, such as an email message, a text message, or a call to an HTTP endpoint. For more information, see [Working with Amazon SNS events and notifications in AWS Database Migration Service \(p. 576\)](#)

## Task status

You can monitor the progress of your task by checking the task status and by monitoring the task's control table. Task status indicates the condition of a AWS DMS task and its associated resources. It includes such indications as if the task is being created, starting, running, or stopped. It also includes the current state of the tables that the task is migrating, such as if a full load of a table has begun or is in progress and details such as the number of inserts, deletes, and updates have occurred for the table. For more information about monitoring task and task resource condition, see [Task status \(p. 559\)](#) and [Table state during tasks \(p. 560\)](#). For more information about control tables, see [Control table task settings \(p. 461\)](#).

## Amazon CloudWatch alarms and logs

Using Amazon CloudWatch alarms, you watch one or more task metrics over a time period that you specify. If a metric exceeds a given threshold, a notification is sent to an Amazon SNS topic. CloudWatch alarms do not invoke actions because they are in a particular state. Rather the state must have changed and been maintained for a specified number of periods. AWS DMS also uses CloudWatch to log task information during the migration process. You can use the AWS CLI or the AWS DMS API to view information about the task logs. For more information about using CloudWatch with AWS DMS, see [Monitoring replication tasks using Amazon CloudWatch \(p. 561\)](#). For more information about monitoring AWS DMS metrics, see [AWS Database Migration Service metrics \(p. 562\)](#). For more information about using AWS DMS task logs, see [Viewing and managing AWS DMS task logs \(p. 567\)](#).

## Time Travel logs

To log and debug replication tasks, you can use AWS DMS Time Travel. In this approach, you use Amazon S3 to store logs and encrypt them using your encryption keys. You can retrieve your S3 logs using date-time filters, then view, download, and obfuscate logs as needed. By doing this, you can "travel back in time" to investigate database activities.

You can use Time Travel with DMS-supported PostgreSQL source endpoints and DMS-supported PostgreSQL and MySQL target endpoints. You can turn on Time Travel only for full-load and CDC tasks and for CDC only tasks. To turn on Time Travel or to modify any existing Time Travel settings, ensure that your task is stopped.

For more information about Time Travel logs, see [Time Travel task settings \(p. 455\)](#). For best practices for using Time Travel logs, see [Troubleshooting replication tasks with Time Travel \(p. 97\)](#).

## AWS CloudTrail logs

AWS DMS is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, IAM role, or an AWS service in AWS DMS. CloudTrail captures all API calls for AWS DMS as events, including calls from the AWS DMS console and from code calls to the AWS DMS API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS DMS. 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 DMS, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see [Logging AWS DMS API calls with AWS CloudTrail \(p. 568\)](#).

## Database logs

You can view, download, and watch database logs for your task endpoints using the AWS Management Console, AWS CLI, or the API for your AWS database service. For more information, see the documentation for your database service at [AWS documentation](#).

For more information, see the following topics.

### Topics

- [Task status \(p. 559\)](#)
- [Table state during tasks \(p. 560\)](#)
- [Monitoring replication tasks using Amazon CloudWatch \(p. 561\)](#)
- [AWS Database Migration Service metrics \(p. 562\)](#)
- [Viewing and managing AWS DMS task logs \(p. 567\)](#)
- [Logging AWS DMS API calls with AWS CloudTrail \(p. 568\)](#)

# Task status

The task status indicated the condition of the task. The following table shows the possible statuses a task can have:

Task status	Description
<b>Creating</b>	AWS DMS is creating the task.
<b>Running</b>	The task is performing the migration duties specified.
<b>Stopped</b>	The task is stopped.
<b>Stopping</b>	The task is being stopped. This is usually an indication of user intervention in the task.
<b>Deleting</b>	The task is being deleted, usually from a request for user intervention.
<b>Failed</b>	The task has failed. For more information, see the task log files.
<b>Starting</b>	The task is connecting to the replication instance and to the source and target endpoints. Any filters and transformations are being applied.
<b>Ready</b>	The task is ready to run. This status usually follows the "creating" status.

Task status	Description
<b>Modifying</b>	The task is being modified, usually due to a user action that modified the task settings.
<b>Moving</b>	The task is in the process of being moved to another replication instance. The replication remains in this state until the move is complete. Deleting the task is the only operation allowed on the replication task while it's being moved.
<b>Failed-move</b>	The task move has failed for any reason, such as not having enough storage space on the target replication instance. When a replication task is in this state, it can be started, modified, moved, or deleted.

The task status bar gives an estimation of the task's progress. The quality of this estimate depends on the quality of the source database's table statistics; the better the table statistics, the more accurate the estimation. For tasks with only one table that has no estimated rows statistic, we are unable to provide any kind of percentage complete estimate. In this case, the task state and the indication of rows loaded can be used to confirm that the task is indeed running and making progress.

Note that the "last updated" column in the DMS console only indicates the time that AWS DMS last updated the table statistics record for a table. It does not indicate the time of the last update to the table.

## Table state during tasks

The AWS DMS console updates information regarding the state of your tables during migration. The following table shows the possible state values:

DMS > Database migration tasks > dms-gs-task

**dms-gs-task**

**Summary**

Status	Type
Running with errors	Full load, ongoing replication

Overview details **Table statistics** CloudWatch metrics Mapping rules Premigration assessments Tags

**Table statistics (157)**  
Total rows include loaded source table rows from Inserts, Deletes, Updates, DDLs, and Full load rows.

Find schema	Load state	Elapsed load time
Schema name	Table	
mysql	user	Table error < 1 s
mysql	server_cost	Table completed < 1 s
mysql	tables_priv	Table completed < 1 s
mysql	gtid_executed	Table completed < 1 s
mysql	replication_asynchronous_connection_failover	Table completed < 1 s

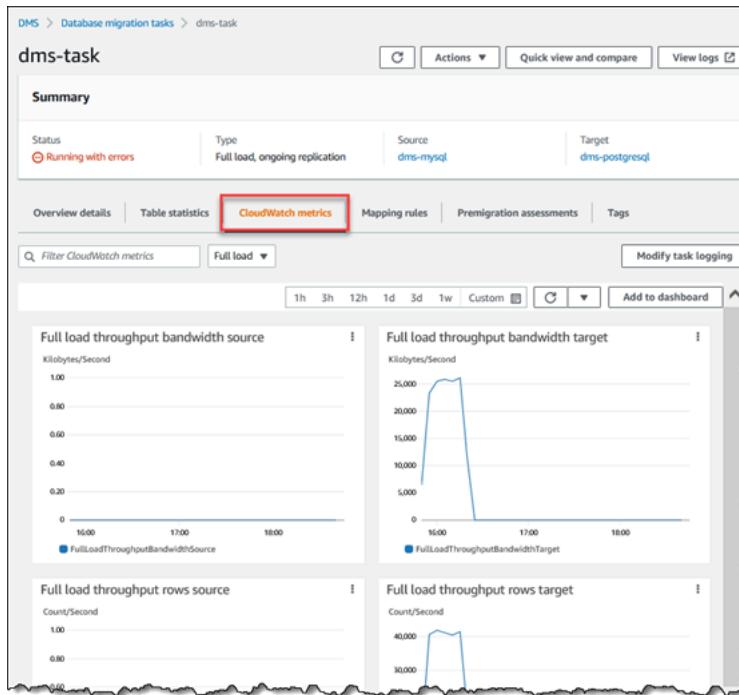
State	Description
<b>Table does not exist</b>	AWS DMS cannot find the table on the source endpoint.
<b>Before load</b>	The full load process has been enabled, but it hasn't started yet.
<b>Full load</b>	The full load process is in progress.
<b>Table completed</b>	Full load has completed.
<b>Table cancelled</b>	Loading of the table has been cancelled.
<b>Table error</b>	An error occurred when loading the table.

## Monitoring replication tasks using Amazon CloudWatch

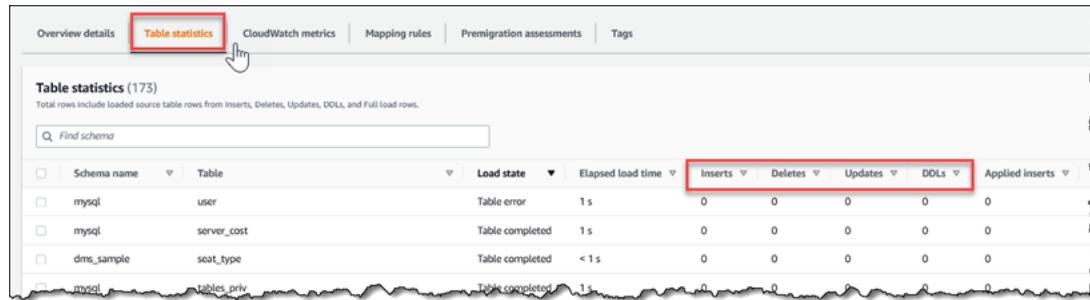
You can use Amazon CloudWatch alarms or events to more closely track your migration. For more information about Amazon CloudWatch, see [What are Amazon CloudWatch, Amazon CloudWatch Events, and Amazon CloudWatch Logs?](#) in the Amazon CloudWatch User Guide. Note that there is a charge for using Amazon CloudWatch.

If your replication task doesn't create CloudWatch logs, see [AWS DMS does not create CloudWatch logs \(p. 656\)](#) in the troubleshooting guide.

The AWS DMS console shows basic CloudWatch statistics for each task, including the task status, percent complete, elapsed time, and table statistics, as shown following. Select the replication task and then select the **CloudWatch metrics** tab.

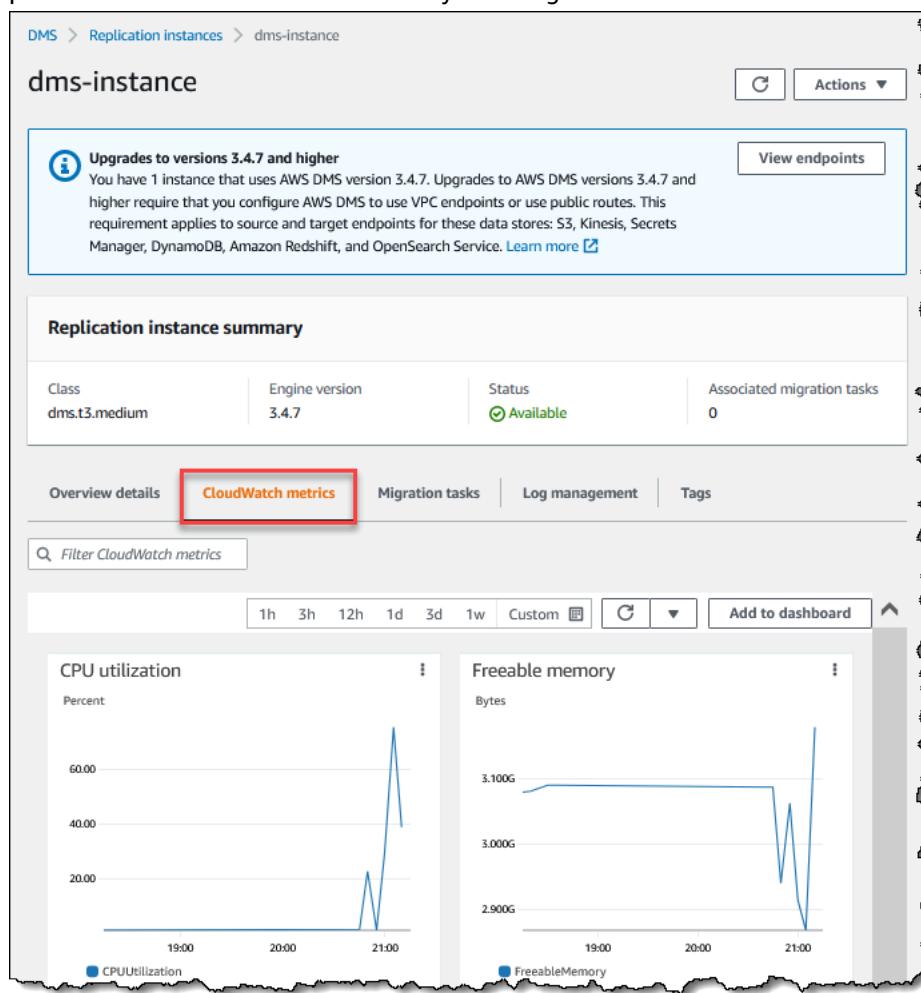


The AWS DMS console shows performance statistics for each table, including the number of inserts, deletions, and updates, when you select the **Table statistics** tab.



The screenshot shows the 'Table statistics' tab in the AWS DMS console. It displays a table of 173 rows, with the last row being 'tables\_priv'. The table includes columns for Schema name, Table, Load state, Elapsed load time, Inserts, Deletes, Updates, DDLs, and Applied inserts. A red box highlights the 'Inserts' column header and the data for the 'user' table, which shows 0 inserts. Another red box highlights the 'Tables completed' status for the 'server\_cost' and 'seat\_type' tables.

In addition, if you select a replication instance from the **Replication Instance** page, you can view performance metrics for the instance by choosing the **CloudWatch metrics** tab.



The screenshot shows the 'dms-instance' page in the AWS DMS console under 'Replication instances'. It includes a summary box about upgrades to version 3.4.7, a 'View endpoints' button, and a 'Replication instance summary' table. The table shows Class: dms.t3.medium, Engine version: 3.4.7, Status: Available, and Associated migration tasks: 0. Below this is a navigation bar with 'Overview details' (highlighted with a red box), 'CloudWatch metrics' (highlighted with a red box), 'Migration tasks', 'Log management', and 'Tags'. A search bar for 'Filter CloudWatch metrics' is also present. Two line charts are displayed: 'CPU utilization' (Percent) and 'Freeable memory' (Bytes). The CPU utilization chart shows a sharp peak around 21:00. The freeable memory chart shows a dip around 21:00.

## AWS Database Migration Service metrics

AWS DMS provides statistics for the following:

- **Host Metrics** – Performance and utilization statistics for the replication host, provided by Amazon CloudWatch. For a complete list of the available metrics, see [Replication instance metrics \(p. 564\)](#).
- **Replication Task Metrics** – Statistics for replication tasks including incoming and committed changes, and latency between the replication host and both the source and target databases. For a complete list of the available metrics, see [Replication task metrics \(p. 565\)](#).
- **Table Metrics** – Statistics for tables that are in the process of being migrated, including the number of insert, update, delete, and DDL statements completed.

Task metrics are divided into statistics between the replication host and the source endpoint, and statistics between the replication host and the target endpoint. You can determine the total statistic for a task by adding two related statistics together. For example, you can determine the total latency, or replica lag, for a task by combining the **CDCLatencySource** and **CDCLatencyTarget** values.

Task metric values can be influenced by current activity on your source database. For example, if a transaction has begun, but has not been committed, then the **CDCLatencySource** metric continues to grow until that transaction has been committed.

For the replication instance, the **FreeableMemory** metric requires clarification. Freeable memory is not a indication of the actual free memory available. It is the memory that is currently in use that can be freed and used for other uses; it's a combination of buffers and cache in use on the replication instance.

While the **FreeableMemory** metric does not reflect actual free memory available, the combination of the **FreeableMemory** and **SwapUsage** metrics can indicate if the replication instance is overloaded.

Monitor these two metrics for the following conditions:

- The **FreeableMemory** metric approaching zero.
- The **SwapUsage** metric increases or fluctuates.

If you see either of these two conditions, they indicate that you should consider moving to a larger replication instance. You should also consider reducing the number and type of tasks running on the replication instance. Full Load tasks require more memory than tasks that just replicate changes.

To estimate the actual memory requirements for a migration task, AWS DMS roughly uses the following methods.

#### Full LOB mode (using single row+update, commit rate)

Memory: (# of lob columns in a table) x (Number of table in parallel, default is 8) x (lob chunk size) x (Commit rate during full load) =  $2 * 8 * 64(k) * 10000k$

##### Note

You can modify your task to reduce **Commit rate during full load**. To change this number in the AWS Management Console, open the console, choose **Tasks**, choose to create or modify a task, and then choose **Advanced Settings**. Under **Tuning Settings**, change the **Commit rate during full load** option.

#### Limited LOB mode (using array)

Memory: (# of lob columns in a table) x (Number of table in parallel, default is 8) x maxlobSize x bulkArraySize =  $2 * 8 * 4096(k) * 1000$

For AWS DMS to perform conversions optimally, the CPU must be available when the conversions happen. Overloading the CPU and not having enough CPU resources can result in slow migrations. AWS DMS can be CPU-intensive, especially when performing heterogeneous migrations and replications such as migrating from Oracle to PostgreSQL. Use of a C4 replication instance class can be a good choice for

these situations. For more information, see [Choosing the right AWS DMS replication instance for your migration \(p. 103\)](#).

## Replication instance metrics

Replication instance monitoring includes Amazon CloudWatch metrics for the following statistics.

Metric	Description
AvailableMemory	An estimate of how much memory is available for starting new applications, without swapping. For more information, see MemAvailable value in /proc/memInfo section of the <a href="#">Linux man-pages</a> .  Units: Bytes
CPUAllocated	The percentage of CPU maximally allocated for the task (0 means no limit).  Units: Percent
CPUUtilization	The percentage of allocated vCPU (virtual CPU) currently in use on the instance.  Units: Percent
DiskQueueDepth	The number of outstanding read/write requests (I/Os) waiting to access the disk.  Units: Count
FreeStorageSpace	The amount of available storage space.  Units: Bytes
FreeMemory	The amount of physical memory available for use by applications, page cache, and for the kernel's own data structures. For more information, see MemFree value in /proc/memInfo section of the <a href="#">Linux man-pages</a> .  Units: Bytes
FreeableMemory	The amount of available random access memory.  Units: Bytes
MemoryAllocated	The maximum allocation of memory for the task (0 means no limits).  Units: MiB
WriteIOPS	The average number of disk write I/O operations per second.  Units: Count/Second
ReadIOPS	The average number of disk read I/O operations per second.  Units: Count/Second
WriteThroughput	The average number of bytes written to disk per second.  Units: Bytes/Second
ReadThroughput	The average number of bytes read from disk per second.

Metric	Description
	Units: Bytes/Second
WriteLatency	The average amount of time taken per disk I/O (output) operation. Units: Milliseconds
ReadLatency	The average amount of time taken per disk I/O (input) operation. Units: Milliseconds
SwapUsage	The amount of swap space used on the replication instance. Units: Bytes
NetworkTransmitThroughput	The outgoing (Transmit) network traffic on the replication instance, including both customer database traffic and AWS DMS traffic used for monitoring and replication. Units: Bytes/second
NetworkReceiveThroughput	The incoming (Receive) network traffic on the replication instance, including both customer database traffic and AWS DMS traffic used for monitoring and replication. Units: Bytes/second

## Replication task metrics

Replication task monitoring includes metrics for the following statistics.

Metric	Description
FullLoadThroughputBandwidth	Outgoing data transmitted from a full load for the target in KB per second.
FullLoadThroughputRowsTarget	Outgoing changes from a full load for the target in rows per second.
CDCIncomingChanges	The total number of change events at a point-in-time that are waiting to be applied to the target. Note that this is not the same as a measure of the transaction change rate of the source endpoint. A large number for this metric usually indicates AWS DMS is unable to apply captured changes in a timely manner, thus causing high target latency.
CDCChangesMemorySource	Amount of rows accumulating in a memory and waiting to be committed from the source. You can view this metric together with CDCChangesDiskSource.
CDCChangesMemoryTarget	Amount of rows accumulating in a memory and waiting to be committed to the target. You can view this metric together with CDCChangesDiskTarget.
CDCChangesDiskSource	Amount of rows accumulating on disk and waiting to be committed from the source. You can view this metric together with CDCChangesMemorySource.
CDCChangesDiskTarget	Amount of rows accumulating on disk and waiting to be committed to the target. You can view this metric together with CDCChangesMemoryTarget.

Metric	Description
CDCThroughputBandwidthTarget	Outgoing data transmitted for the target in KB per second. CDCThroughputBandwidth records outgoing data transmitted on sampling points. If no task network traffic is found, the value is zero. Because CDC does not issue long-running transactions, network traffic may not be recorded.
CDCThroughputRowsSource	Incoming task changes from the source in rows per second.
CDCThroughputRowsTarget	Outgoing task changes for the target in rows per second.
CDCLatencySource	<p>The gap, in seconds, between the last event captured from the source endpoint and current system time stamp of the AWS DMS instance. CDCLatencySource represents the latency between source and replication instance. High CDCLatencySource means the process of capturing changes from source is delayed. To identify latency in an ongoing replication, you can view this metric together with CDCLatencyTarget. If both CDCLatencySource and CDCLatencyTarget are high, investigate CDCLatencySource first.</p> <p>CDCSourceLatency can be 0 when there is no replication lag between the source and the replication instance. CDCSourceLatency can also become zero when the replication task attempts to read the next event in the source's transaction log and there are no new events compared to the last time it read from the source. When this happens, the task resets the CDCSourceLatency to 0.</p>
CDCLatencyTarget	<p>The gap, in seconds, between the first event timestamp waiting to commit on the target and the current timestamp of the AWS DMS instance. Target latency is the difference between the replication instance server time and the oldest unconfirmed event id forwarded to a target component. In other words, target latency is the timestamp difference between the replication instance and the oldest event applied but unconfirmed by TRG endpoint (99%). When CDCLatencyTarget is high, it indicates the process of applying change events to the target is delayed. To identify latency in an ongoing replication, you can view this metric together with CDCLatencySource. If CDCLatencyTarget is high but CDCLatencySource isn't high, investigate if:</p> <ul style="list-style-type: none"> <li>• No primary keys or indexes are in the target</li> <li>• Resource bottlenecks occur in the target or replication instance</li> <li>• Network issues reside between replication instance and target</li> </ul>
CPUUtilization	<p>The percentage of CPU being used by a task across multiple cores. The semantics of task CPUUtilization is slightly different from replication CPUUtilizaiton. If 1 vCPU is fully used, it indicates 100%, but if multiple vCPUs are in use, the value could be above 100%.</p> <p>Units: Percent</p>
SwapUsage	<p>The amount of swap used by the task.</p> <p>Units: Bytes</p>

Metric	Description
MemoryUsage	The control group (cgroup) <b>memory.usage_in_bytes</b> consumed by a task. DMS uses cgroups to control the usage of system resources such as memory and CPU. This metric indicates a task's memory usage in Megabytes within the cgroup allocated for that task. The cgroup limits are based on the resources available for your DMS replication instance class. <b>memory.usage_in_bytes</b> consists of resident set size (RSS), cache, and swap components of memory. The operating system can reclaim cache memory if needed. We recommend that you also monitor the replication instance metric, <b>AvailableMemory</b> .

## Viewing and managing AWS DMS task logs

You can use Amazon CloudWatch to log task information during an AWS DMS migration process. You enable logging when you select task settings. For more information, see [Logging task settings \(p. 460\)](#).

To view logs of a task that ran, follow these steps:

1. Open the AWS DMS console, and choose **Database migration tasks** from the navigation pane. The Database migration tasks dialog appears.
2. Select the name of your task. The Overview details dialog appears.
3. Locate the **Migration task logs** section and choose **View CloudWatch Logs**.

In addition, you can use the AWS CLI or AWS DMS API to view information about task logs. To do this, use the `describe-replication-instance-task-logs` AWS CLI command or the AWS DMS API action `DescribeReplicationInstanceTaskLogs`.

For example, the following AWS CLI command shows the task log metadata in JSON format.

```
$ aws dms describe-replication-instance-task-logs \
--replication-instance-arn arn:aws:dms:us-east-1:237565436:rep:CDSFSFFFSSUFCAY
```

A sample response from the command is as follows.

```
{
    "ReplicationInstanceTaskLogs": [
        {
            "ReplicationTaskArn": "arn:aws:dms:us-east-1:237565436:task:MY34U6Z4MSY52GRTIX304AY",
            "ReplicationTaskName": "mysql-to-ddb",
            "ReplicationInstanceTaskLogSize": 3726134
        }
    ],
    "ReplicationInstanceArn": "arn:aws:dms:us-east-1:237565436:rep:CDSFSFFFSSUFCAY"
}
```

In this response, there is a single task log (`mysql-to-ddb`) associated with the replication instance. The size of this log is 3,726,124 bytes.

You can use the information returned by `describe-replication-instance-task-logs` to diagnose and troubleshoot problems with task logs. For example, if you enable detailed debug logging for a task,

the task log will grow quickly—potentially consuming all of the available storage on the replication instance, and causing the instance status to change to `storage-full`. By describing the task logs, you can determine which ones you no longer need; then you can delete them, freeing up storage space.

To delete the task logs for a task, set the task setting `DeleteTaskLogs` to true. For example, the following JSON deletes the task logs when modifying a task using the AWS CLI `modify-replication-task` command or the AWS DMS API `ModifyReplicationTask` action.

```
{  
  "Logging": {  
    "DeleteTaskLogs":true  
  }  
}
```

## Logging AWS DMS API calls with AWS CloudTrail

AWS DMS is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS DMS. CloudTrail captures all API calls for AWS DMS as events, including calls from the AWS DMS console and from code calls to the AWS DMS API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS DMS. 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 DMS, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the [AWS CloudTrail User Guide](#).

## AWS DMS information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS DMS, 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 DMS, 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 AWS Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see:

- [Overview for creating a trail](#)
- [CloudTrail supported services and integrations](#)
- [Configuring Amazon SNS notifications for CloudTrail](#)
- [Receiving CloudTrail log files from multiple AWS Regions](#) and [Receiving CloudTrail log files from multiple accounts](#)

All AWS DMS actions are logged by CloudTrail and are documented in the [AWS Database Migration Service API Reference](#). For example, calls to the `CreateReplicationInstance`, `TestConnection` and `StartReplicationTask` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or IAM user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the [CloudTrail userIdentity element](#).

## Understanding AWS DMS log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `RebootReplicationInstance` action.

```
{  
    "eventVersion": "1.05",  
    "userIdentity": {  
        "type": "AssumedRole",  
        "principalId": "AKIAIOSFODNN7EXAMPLE:johndoe",  
        "arn": "arn:aws:sts::123456789012:assumed-role/admin/johndoe",  
        "accountId": "123456789012",  
        "accessKeyId": "ASIAZYFI33SINAD0JJZEZW",  
        "sessionContext": {  
            "attributes": {  
                "mfaAuthenticated": "false",  
                "creationDate": "2018-08-01T16:42:09Z"  
            },  
            "sessionIssuer": {  
                "type": "Role",  
                "principalId": "AKIAIOSFODNN7EXAMPLE",  
                "arn": "arn:aws:iam::123456789012:role/admin",  
                "accountId": "123456789012",  
                "userName": "admin"  
            }  
        }  
    },  
    "eventTime": "2018-08-02T00:11:44Z",  
    "eventSource": "dms.amazonaws.com",  
    "eventName": "RebootReplicationInstance",  
    "awsRegion": "us-east-1",  
    "sourceIPAddress": "72.21.198.64",  
    "userAgent": "console.amazonaws.com",  
    "requestParameters": {  
        "forceFailover": false,  
        "replicationInstanceArn": "arn:aws:dms:us-  
east-1:123456789012:rep:EX4MBJ2NMRDL3BMAYJ0XUGYPUE"  
    },  
    "responseElements": {  
        "replicationInstance": {  
            "replicationInstanceIdentifier": "replication-instance-1",  
            "replicationInstanceStatus": "rebooting",  
            "allocatedStorage": 50,  
            "replicationInstancePrivateIpAddresses": [  
                "172.31.20.204"  
            ],  
            "instanceCreateTime": "Aug 1, 2018 11:56:21 PM",  
            "autoMinorVersionUpgrade": true,  
            "replicationGroup": {  
                "replicationGroupIdentifier": "replication-group-1",  
                "replicationGroupStatus": "active",  
                "replicationInstances": [  
                    "arn:aws:dms:us-east-1:123456789012:rep:EX4MBJ2NMRDL3BMAYJ0XUGYPUE"  
                ]  
            }  
        }  
    }  
}
```

```

"engineVersion": "2.4.3",
"publiclyAccessible": true,
"replicationInstanceClass": "dms.t2.medium",
"availabilityZone": "us-east-1b",
"kmsKeyId": "arn:aws:kms:us-east-1:123456789012:key/f7bc0f8e-1a3a-4ace-9faa-e8494fa3921a",
"replicationSubnetGroup": {
    "vpcId": "vpc-1f6a9c6a",
    "subnetGroupStatus": "Complete",
    "replicationSubnetGroupArn": "arn:aws:dms:us-east-1:123456789012:subgrp:EDHRVRBAAAPONQAIYWP4NUW22M",
    "subnets": [
        {
            "subnetIdentifier": "subnet-cbfff283",
            "subnetAvailabilityZone": {
                "name": "us-east-1b"
            },
            "subnetStatus": "Active"
        },
        {
            "subnetIdentifier": "subnet-d7c825e8",
            "subnetAvailabilityZone": {
                "name": "us-east-1e"
            },
            "subnetStatus": "Active"
        },
        {
            "subnetIdentifier": "subnet-6746046b",
            "subnetAvailabilityZone": {
                "name": "us-east-1f"
            },
            "subnetStatus": "Active"
        },
        {
            "subnetIdentifier": "subnet-bac383e0",
            "subnetAvailabilityZone": {
                "name": "us-east-1c"
            },
            "subnetStatus": "Active"
        },
        {
            "subnetIdentifier": "subnet-42599426",
            "subnetAvailabilityZone": {
                "name": "us-east-1d"
            },
            "subnetStatus": "Active"
        },
        {
            "subnetIdentifier": "subnet-da327bf6",
            "subnetAvailabilityZone": {
                "name": "us-east-1a"
            },
            "subnetStatus": "Active"
        }
    ],
    "replicationSubnetGroupIdentifier": "default-vpc-1f6a9c6a",
    "replicationSubnetGroupDescription": "default group created by console for vpc id vpc-1f6a9c6a"
},
"replicationInstanceEniId": "eni-0d6db8c7137cb9844",
"vpcSecurityGroups": [
    {
        "vpcSecurityGroupId": "sg-f839b688",
        "status": "active"
    }
],

```

```
    "pendingModifiedValues": {},
    "replicationInstancePublicIpAddresses": [
        "18.211.48.119"
    ],
    "replicationInstancePublicIpAddress": "18.211.48.119",
    "preferredMaintenanceWindow": "fri:22:44-fri:23:14",
    "replicationInstanceArn": "arn:aws:dms:us-
east-1:123456789012:rep:EX4MBJ2NMRDL3BMAYJOXUGYPUE",
    "replicationInstanceEniIds": [
        "eni-0d6db8c7137cb9844"
    ],
    "multiAZ": false,
    "replicationInstancePrivateIpAddress": "172.31.20.204",
    "patchingPrecedence": 0
}
},
"requestID": "a3c83c11-95e8-11e8-9d08-4b8f2b45bfd5",
"eventID": "b3c4adb1-e34b-4744-bdeb-35528062a541",
"eventType": "AwsApiCall",
"recipientAccountId": "123456789012"
}
```

# Working with Amazon EventBridge events and notifications in AWS Database Migration Service

You can use Amazon EventBridge to provide notification of when an AWS DMS event occurs, for example the creation or deletion of a replication instance. EventBridge receives events and routes notification of an event as defined by event rules. You can work with notifications in any form supported by Amazon EventBridge for an AWS Region. For more information about using Amazon EventBridge, see [What is Amazon EventBridge?](#) in the *Amazon EventBridge User Guide*.

**Note**

Working with Amazon EventBridge events is supported in AWS DMS version 3.4.6 and later.

EventBridge receives an event, an indicator of a change in AWS DMS environment, and applies a rule to route the event to a notification mechanism. Rules match events to notification mechanisms based on the structure of the event, called an *event pattern*.

AWS DMS groups events into categories that you can apply an event rule to, so you can be notified when an event in that category occurs. For example, suppose that you apply an EventBridge event rule to the Creation category for a given replication instance. You're then notified whenever a creation-related event occurs that affects your replication instance. If you apply a rule to a Configuration Change category for a replication instance, you're notified when the replication instance's configuration is changed. For a list of the event categories provided by AWS DMS, see the AWS DMS event categories and event messages, following.

**Note**

To allow publishing from [events.amazonaws.com](https://events.amazonaws.com), make sure to update your Amazon SNS topics' access policies. For more information, see [Using resource-based policies for Amazon EventBridge](#) in the *Amazon EventBridge User Guide*.

For more information about moving event subscriptions to Amazon EventBridge, see [migrate active event subscriptions from DMS to Amazon EventBridge. \(p. 576\)](#)

For more information on using text messages with Amazon SNS, see [Sending and receiving SMS notifications using Amazon SNS](#).

## Using Amazon EventBridge event rules for AWS DMS

Amazon EventBridge sends event notifications to the addresses that you provide when you create an EventBridge event rule. You might want to create several different rules. For example, you might create one rule receiving all event notifications and another rule that includes only critical events for your production DMS resources. You can also turn on or turn off event notifications in EventBridge.

### To create Amazon EventBridge rules that react to AWS DMS events

- Perform the steps described in [Creating Amazon EventBridge rules that react to events](#) in the *Amazon EventBridge User Guide*, and create a rule for AWS DMS events:

- a. Specify a notification action to take when EventBridge receives an event that matches the event pattern in the rule. When an event matches, EventBridge sends the event and invokes the action defined in the rule.
- b. For **Service provider**, choose **AWS**.
- c. For **Service name**, choose **Database Migration Service (DMS)**.

You can then begin to receive event notifications.

The following JSON example shows an EventBridge events model for an AWS DMS service.

```
{
    "version": "0",
    "id": "11a11b11-222b-333a-44d4-01234a5b67890",
    "detail-type": "DMS Replication Task State Change",
    "source": "aws.dms",
    "account": "0123456789012",
    "time": "1970-01-01T00:00:00Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:dms:us-east-1:012345678901:task:AAAABBBB0CCCCDDDEEEE1FFFF2GGG3FFFFFF3"
    ],
    "detail": {
        "type": "ReplicationTask",
        "category": "StateChange",
        "eventType": "REPLICATION_TASK_STARTED",
        "eventName": "DMS-EVENT-0069",
        "resourceLink": "https://console.aws.amazon.com/dms/v2/home?region=us-east-1#taskDetails/taskName",
        "detailMessage": "Replication task started, with flag = fresh start"
    }
}
```

For the list of categories and events that you can be notified of, see the following section.

## AWS DMS event categories and event messages

AWS DMS generates a significant number of events in categories that you can identify. Each category applies to a replication instance or replication task source types.

The following table shows the possible categories and events for the replication instance source type.

Category	Description
Configuration Change	The replication instance class for this replication instance is being changed.
Configuration Change	The replication instance class for this replication instance has changed.
Configuration Change	The storage for the replication instance is being increased.
Configuration Change	The storage for the replication instance has been increased.
Configuration Change	The replication instance is transitioning to a Multi-AZ configuration.
Configuration Change	The replication instance finished transitioning to a Multi-AZ configuration.

<b>Category</b>	<b>Description</b>
Configuration Change	The replication instance is transitioning to a Single-AZ configuration.
Configuration Change	The replication instance has finished transitioning to a Single-AZ configuration.
Creation	A replication instance is being created.
Creation	A replication instance is created.
Deletion	The replication instance is being deleted.
Deletion	The replication instance is deleted.
Maintenance	Management software on the replication instance has been updated.
Maintenance	Offline maintenance of the replication instance is taking place. The replication instance is currently unavailable.
Maintenance	Offline maintenance of the replication instance is complete. The replication instance is now available.
Maintenance	A replication instance is in a state that can't be upgraded.
LowStorage	The replication instance has consumed more than 90% of its allocated storage. You can monitor the storage space for a replication instance using the Free Storage Space metric.
Failover	Failover started for a Multi-AZ replication instance.
Failover	Failover is complete for a Multi-AZ replication instance.
Failover	Multi-AZ failover to standby is complete.
Failover	Multi-AZ activation has started.
Failover	Multi-AZ activation had completed.
Failover	If you request failover too frequently, this event occurs instead of regular failover events.
Failure	The replication instance has gone into storage failure.
Failure	The replication instance has failed due to an incompatible network.
Failure	The service can't access the AWS KMS key that was used to encrypt the data volume.

The following table shows the possible categories and events for the replication task source type.

<b>Category</b>	<b>Description</b>
State Change	The replication task has started.
State Change	A reload of table details has been requested.
State Change	The replication task has stopped.
State Change	Reading was paused because the swap files limit was reached.

Category	Description
State Change	Reading was paused because the swap files limit was reached.
State Change	Reading resumed.
Failure	The replication task has failed.
Failure	A call to delete the task has failed to clean up task data.
Configuration Change	The replication task is modified.
Deletion	The replication task is deleted.
Creation	The replication task is created.

# Working with Amazon SNS events and notifications in AWS Database Migration Service

Beginning with the release of AWS DMS 3.4.6 and with later versions, we recommend that you use Amazon EventBridge to provide notifications when an AWS DMS event occurs. For more information about using EventBridge events with AWS DMS, see [Working with Amazon EventBridge events and notifications in AWS Database Migration Service \(p. 572\)](#).

## Moving event subscriptions to Amazon EventBridge

You can use the following AWS CLI command to migrate active event subscriptions from DMS to Amazon EventBridge, up to 10 at a time.

```
update-subscriptions-to-event-bridge [--force-move | --no-force-move]
```

By default, AWS DMS only migrates active event subscriptions when your replication instance is current with AWS DMS 3.4.6 and higher. To override this default behavior, use the `--force-move` option. However, some types of events might not be available by using Amazon EventBridge if your replication instances aren't upgraded.

To run the `update-subscriptions-to-event-bridge` CLI command, an AWS Identity and Access Management (IAM) user must have the following policy permissions.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "SNS:GetTopicAttributes",
                "SNS:SetTopicAttributes",
                "events:PutTargets",
                "events:EnableRule",
                "events:PutRule"
            ],
            "Resource": "*"
        }
    ]
}
```

For more information about moving subscriptions to EventBridge, see [UpdateSubscriptionsToEventBridge](#) in the *AWS Database Migration Service API Reference*.

# Working with Amazon SNS events and notifications

AWS DMS versions 3.4.5 and earlier support working with events and notifications as described following.

AWS Database Migration Service (AWS DMS) can use Amazon Simple Notification Service (Amazon SNS) to provide notifications when an AWS DMS event occurs, for example the creation or deletion of a replication instance. You can work with these notifications in any form supported by Amazon SNS for an AWS Region, such as an email message, a text message, or a call to an HTTP endpoint.

AWS DMS groups events into categories that you can subscribe to, so you can be notified when an event in that category occurs. For example, if you subscribe to the Creation category for a given replication instance, you are notified whenever a creation-related event occurs that affects your replication instance. If you subscribe to a Configuration Change category for a replication instance, you are notified when the replication instance's configuration is changed. You also receive notification when an event notification subscription changes. For a list of the event categories provided by AWS DMS, see [AWS DMS event categories and event messages for SNS notifications \(p. 578\)](#), following.

AWS DMS sends event notifications to the addresses you provide when you create an event subscription. You might want to create several different subscriptions, such as one subscription receiving all event notifications and another subscription that includes only critical events for your production DMS resources. You can easily turn off notification without deleting a subscription by deselecting the **Enabled** option in the AWS DMS console, or by setting the Enabled parameter to *false* using the AWS DMS API.

## Note

AWS DMS event notifications using SMS text messages are currently available for AWS DMS resources in all AWS Regions where Amazon SNS is supported. For a list of AWS Regions and countries where Amazon SNS supports SMS messaging, see [Supported Regions and countries](#). For more information on using text messages with SNS, see [Sending and receiving SMS notifications using Amazon SNS](#).

AWS DMS event notifications differ from CloudTrail events in CloudWatch or EventBridge. CloudTrail event notifications can be generated by any API invocation. DMS sends a notification only when a DMS event occurs.

AWS DMS uses a subscription identifier to identify each subscription. You can have multiple AWS DMS event subscriptions published to the same Amazon SNS topic. When you use event notification, Amazon SNS fees apply; for more information on Amazon SNS billing, see [Amazon SNS pricing](#).

To subscribe to AWS DMS events with Amazon SNS, use the following process:

1. Create an Amazon SNS topic. In the topic, you specify what type of notification you want to receive and to what address or number the notification will go to.
2. Create an AWS DMS event notification subscription by using the AWS Management Console, AWS CLI, or AWS DMS API.
3. AWS DMS sends an approval email or SMS message to the addresses you submitted with your subscription. To confirm your subscription, click the link in the approval email or SMS message.
4. When you have confirmed the subscription, the status of your subscription is updated in the AWS DMS console's **Event subscriptions** section.
5. You then begin to receive event notifications.

For the list of categories and events that you can be notified of, see the following section. For more details about subscribing to and working with AWS DMS event subscriptions, see [Subscribing to AWS DMS event notification using SNS \(p. 580\)](#).

# AWS DMS event categories and event messages for SNS notifications

## Important

Beginning with the release of AWS DMS 3.4.6 and with later versions, we recommend that you use Amazon EventBridge to provide notifications when an AWS DMS event occurs. For more information about using EventBridge events with AWS DMS, see [Working with Amazon EventBridge events and notifications in AWS Database Migration Service \(p. 572\)](#).

AWS DMS generates a significant number of events in categories that you can subscribe to using the AWS DMS console or the AWS DMS API. Each category applies to a source type; currently AWS DMS supports the replication instance and replication task source types.

The following table shows the possible categories and events for the replication instance source type.

Category	DMS event ID	Description
Configuration Change	DMS-EVENT-0012	The replication instance class for this replication instance is being changed.
Configuration Change	DMS-EVENT-0014	The replication instance class for this replication instance has changed.
Configuration Change	DMS-EVENT-0018	The storage for the replication instance is being increased.
Configuration Change	DMS-EVENT-0017	The storage for the replication instance has been increased.
Configuration Change	DMS-EVENT-0024	The replication instance is transitioning to a Multi-AZ configuration.
Configuration Change	DMS-EVENT-0025	The replication instance finished transitioning to a Multi-AZ configuration.
Configuration Change	DMS-EVENT-0030	The replication instance is transitioning to a Single-AZ configuration.
Configuration Change	DMS-EVENT-0029	The replication instance has finished transitioning to a Single-AZ configuration.
Creation	DMS-EVENT-0067	A replication instance is being created.
Creation	DMS-EVENT-0005	A replication instance is created.
Deletion	DMS-EVENT-0066	The replication instance is being deleted.
Deletion	DMS-EVENT-0003	The replication instance is deleted.
Maintenance	DMS-EVENT-0047	Management software on the replication instance has been updated.
Maintenance	DMS-EVENT-0026	Offline maintenance of the replication instance is taking place. The replication instance is currently unavailable.
Maintenance	DMS-EVENT-0027	Offline maintenance of the replication instance is complete. The replication instance is now available.

Category	DMS event ID	Description
Maintenance	DMS-EVENT-0068	A replication instance is in a state that can't be upgraded.
LowStorage	DMS-EVENT-0007	The replication instance has consumed more than 90% of its allocated storage. You can monitor the storage space for a replication instance using the Free Storage Space metric.
Failover	DMS-EVENT-0013	Failover started for a Multi-AZ replication instance.
Failover	DMS-EVENT-0049	Failover is complete for a Multi-AZ replication instance.
Failover	DMS-EVENT-0015	Multi-AZ failover to standby is complete.
Failover	DMS-EVENT-0050	Multi-AZ activation has started.
Failover	DMS-EVENT-0051	Multi-AZ activation had completed.
Failover	DMS-EVENT-0034	If you request failover too frequently, this event occurs instead of regular failover events.
Failure	DMS-EVENT-0031	The replication instance has gone into storage failure.
Failure	DMS-EVENT-0036	The replication instance has failed due to an incompatible network.
Failure	DMS-EVENT-0037	The service can't access the AWS KMS key used to encrypt the data volume.

The following table shows the possible categories and events for the replication task source type.

Category	DMS event ID	Description
State Change	DMS-EVENT-0069	The replication task has started.
State Change	DMS-EVENT-0081	A reload of table details has been requested.
State Change	DMS-EVENT-0079	The replication task has stopped.
State Change	DMS-EVENT-0091	Reading paused, swap files limit reached.
State Change	DMS-EVENT-0092	Reading paused, disk usage limit reached.
State Change	DMS-EVENT-0093	Reading resumed.
Failure	DMS-EVENT-0078	The replication task has failed.
Failure	DMS-EVENT-0082	A call to delete the task has failed to clean up task data.
Configuration Change	DMS-EVENT-0080	The replication task is modified.
Deletion	DMS-EVENT-0073	The replication task is deleted.
Creation	DMS-EVENT-0074	The replication task is created.

The following example shows an AWS DMS event subscription with the State Change category.

```
Resources:  
  DMSEvent:  
    Type: AWS::DMS::EventSubscription  
    Properties:  
      Enabled: true  
      EventCategories: State Change  
      SnsTopicArn: arn:aws:sns:us-east-1:123456789:testSNS  
      SourceIds: []  
      SourceType: replication-task
```

## Subscribing to AWS DMS event notification using SNS

### Important

Beginning with the release of AWS DMS 3.4.6 and with later versions, we recommend that you use Amazon EventBridge to provide notifications when an AWS DMS event occurs. For more information about using EventBridge events with AWS DMS, see [Working with Amazon EventBridge events and notifications in AWS Database Migration Service \(p. 572\)](#).

You can create an AWS DMS event notification subscription so you can be notified when an AWS DMS event occurs. The simplest way to create a subscription is with the AWS DMS console. In a notification subscription, you choose how and where to send notifications. You specify the type of source you want to be notified of; currently AWS DMS supports the replication instance and replication task source types. And, depending on the source type you select, you choose the event categories and identify the source you want to receive event notifications for.

## Using the AWS Management Console

### Important

Beginning with the release of AWS DMS 3.4.6 and with later versions, we recommend that you use Amazon EventBridge to provide notifications when an AWS DMS event occurs. For more information about using EventBridge events with AWS DMS, see [Working with Amazon EventBridge events and notifications in AWS Database Migration Service \(p. 572\)](#).

### To subscribe to AWS DMS event notification with Amazon SNS by using the console

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.  
If you're signed in as an IAM user, make sure that you have the appropriate permissions to access AWS DMS.
2. In the navigation pane, choose **Event subscriptions**.
3. On the **Event subscriptions** page, choose **Create event subscription**.
4. On the **Create event subscription** page, do the following:
  - a. Under **Details**, for **Name**, enter a name for the event notification subscription.
  - b. Choose **Enabled** to enable the subscription. If you want to create the subscription but not have notifications sent yet, don't choose **Enabled**.
  - c. Under **Target**, choose either **Existing topics**, **Create new email topic** or **Create new SMS topic** to send notifications. Make sure that you either have an existing Amazon SNS topic to send notices to or create the topic. If you create a topic, you can enter an email address where notifications will be sent.

- d. Under **Event source**, for **Source type**, choose a source type. The only options are **replication-instance** and **replication-task**.
- e. Depending on the source type you selected, choose the event categories and sources you want to receive event notifications for.

## Create event subscription

### Details

#### Name

The name for your event subscription

EventSubscriptionExample

Enabled

### Target

Send notification to

- Existing topics
- Create new email topic
- Create new SMS topic

#### Topic name

ProdEventSubscription

#### With these recipients

Email addresses or phone numbers of SMS enabled devices to send the notifications to

user@domain.com

### Event source

#### Source type

Source Type of resource this subscription will consume events from

replication-instance

- f. Select **Create event subscription**.

The AWS DMS console indicates that the subscription is being created.

**Note**

You can also create Amazon SNS event notification subscriptions using the AWS DMS API and CLI. For more information, see the [CreateEventSubscription](#) in the *AWS DMS API Reference* and [create-event-subscription](#) in the *AWS DMS CLI Reference* documentation.

## Validating the access policy of your SNS topic

Your SNS access policy requires permissions that allow AWS DMS to publish events to your SNS topic. You can validate and update your access policy as described in the following procedures.

### To validate your access policy

1. Open the **Amazon SNS** console.
2. From the navigation panel, choose **Topics** and select the topic that you want to receive DMS notifications about.
3. Select the **Access policy** tab.

You can update your policy if your SNS access policy doesn't allow AWS DMS to publish events to your SNS topic.

### To update your access policy

1. From the **Details** section of your topic page, choose **Edit**.
2. Expand the **Access policy** section, and attach the following policy into the JSON editor.

```
{  
    "Sid": "dms-allow-publish",  
    "Effect": "Allow",  
    "Principal": {  
        "Service": "dms.amazonaws.com"  
    },  
    "Action": "sns:Publish",  
    "Resource": "your-SNS-topic-ARN"  
}
```

We recommend that you further restrict the access to your SNS topic by specifying the `aws:SourceArn` condition, which is the DMS EventSubscription Arn that publishes events to the topic.

```
...  
"Resource": "your-SNS-topic-ARN"  
"Condition": {  
    "StringEquals": {  
        "aws:SourceArn": "arn:partition:dms:your-AWS-region:your-AWS-account-ID:es:your-dms-es-arn or *"  
    }  
}
```

3. Choose **Save changes**.

# AWS DMS data validation

## Topics

- [Replication task statistics \(p. 584\)](#)
- [Replication task statistics with Amazon CloudWatch \(p. 586\)](#)
- [Revalidating tables during a task \(p. 587\)](#)
- [Using JSON editor to modify validation rules \(p. 587\)](#)
- [Validation only tasks \(p. 588\)](#)
- [Troubleshooting \(p. 589\)](#)
- [Limitations \(p. 590\)](#)

AWS DMS provides support for data validation to ensure that your data was migrated accurately from the source to the target. If enabled, validation begins immediately after a full load is performed for a table. Validation compares the incremental changes for a CDC-enabled task as they occur.

During data validation, AWS DMS compares each row in the source with its corresponding row at the target, verifies the rows contain the same data, and reports any mismatches. To accomplish this AWS DMS issues appropriate queries to retrieve the data. Note that these queries will consume additional resources at the source and target as well as additional network resources.

For a CDC only task with validation enabled, all pre-existing data in a table is validated before starting validation of new data.

Data validation works with the following databases wherever AWS DMS supports them as source and target endpoints:

- Oracle
- PostgreSQL-compatible database (PostgreSQL, Aurora PostgreSQL, or Aurora Serverless for PostgreSQL)
- MySQL-compatible database (MySQL, MariaDB, Aurora MySQL, or Aurora Serverless for MySQL)
- Microsoft SQL Server
- IBM Db2 LUW

For more information about the supported endpoints, see [Working with AWS DMS endpoints \(p. 131\)](#).

Data validation requires additional time, beyond the amount required for the migration itself. The extra time required depends on how much data was migrated.

For more information about these settings, see [Data validation task settings \(p. 467\)](#).

For an example of ValidationSettings task settings in a JSON file, see [Task settings example \(p. 450\)](#).

## Replication task statistics

When data validation is enabled, AWS DMS provides the following statistics at the table level:

- **ValidationState**—The validation state of the table. The parameter can have the following values:
  - **Not enabled**—Validation is not enabled for the table in the migration task.
  - **Pending records**—Some records in the table are waiting for validation.

- **Mismatched records**—Some records in the table don't match between the source and target. A mismatch might occur for a number of reasons; For more information, check the awsdms\_validation\_failures\_v1 table on the target endpoint.
- **Suspended records**—Some records in the table can't be validated.
- **No primary key**—The table can't be validated because it had no primary key.
- **Table error**—The table wasn't validated because it was in an error state and some data wasn't migrated.
- **Validated**—All rows in the table are validated. If the table is updated, the status can change from Validated.
- **Error**—The table can't be validated because of an unexpected error.
- **Pending validation**—The table is waiting validation.
- **Preparing table**—Preparing the table enabled in the migration task for validation.
- **Pending revalidation**—All rows in the table are pending validation after the table was updated.
- **ValidationPending**—The number of records that have been migrated to the target, but that haven't yet been validated.
- **ValidationSuspended**—The number of records that AWS DMS can't compare. For example, if a record at the source is constantly being updated, AWS DMS can't compare the source and the target.
- **ValidationFailed**—The number of records that didn't pass the data validation phase.

For an example of ValidationSettings task settings in a JSON file, see [Task settings example \(p. 450\)](#).

You can view the data validation information using the console, the AWS CLI, or the AWS DMS API.

- On the console, you can choose to validate a task when you create or modify the task. To view the data validation report using the console, choose the task on the **Tasks** page and choose the **Table statistics** tab in the details section.
- Using the CLI, set the EnableValidation parameter to true when creating or modifying a task to begin data validation. The following example creates a task and enables data validation.

```
create-replication-task
--replication-task-settings '{"ValidationSettings":{"EnableValidation":true}}'
--replication-instance-arn arn:aws:dms:us-east-1:5731014:
    rep:36KWVMB7Q
--source-endpoint-arn arn:aws:dms:us-east-1:5731014:
    endpoint:CSZAEFQURFYMM
--target-endpoint-arn arn:aws:dms:us-east-1:5731014:
    endpoint:CGPP7MF6WT4JQ
--migration-type full-load-and-cdc
--table-mappings '{"rules": [{"rule-type": "selection", "rule-id": "1",
    "rule-name": "1", "object-locator": {"schema-name": "data_types", "table-name": "%"}, "rule-action": "include"}]}'
```

Use the `describe-table-statistics` command to receive the data validation report in JSON format. The following command shows the data validation report.

```
aws dms describe-table-statistics --replication-task-arn arn:aws:dms:us-east-1:5731014:
rep:36KWVMB7Q
```

The report would be similar to the following.

```
{ "ReplicationTaskArn": "arn:aws:dms:us-west-2:5731014:task:VFPFTYKK2RYSI",
```

```
"TableStatistics": [
    {
        "ValidationPendingRecords": 2,
        "Inserts": 25,
        "ValidationState": "Pending records",
        "ValidationSuspendedRecords": 0,
        "LastUpdateTime": 1510181065.349,
        "FullLoadErrorRows": 0,
        "FullLoadCondtnlChkFailedRows": 0,
        "Ddls": 0,
        "TableName": "t_binary",
        "ValidationFailedRecords": 0,
        "Updates": 0,
        "FullLoadRows": 10,
        "TableState": "Table completed",
        "SchemaName": "d_types_s_sqlserver",
        "Deletes": 0
    }
]
```

- Using the AWS DMS API, create a task using the **CreateReplicationTask** action and set the **EnableValidation** parameter to **true** to validate the data migrated by the task. Use the **DescribeTableStatistics** action to receive the data validation report in JSON format.

## Replication task statistics with Amazon CloudWatch

When Amazon CloudWatch is enabled, AWS DMS provides the following replication task statistics:

- **ValidationSucceededRecordCount**— Number of rows that AWS DMS validated, per minute.
- **ValidationAttemptedRecordCount**— Number of rows that validation was attempted, per minute.
- **ValidationFailedOverallCount**— Number of rows where validation failed.
- **ValidationSuspendedOverallCount**— Number of rows where validation was suspended.
- **ValidationPendingOverallCount**— Number of rows where the validation is still pending.
- **ValidationBulkQuerySourceLatency**— AWS DMS can do data validation in bulk, especially in certain scenarios during a full-load or on-going replication when there are many changes. This metric indicates the latency required to read a bulk set of data from the source endpoint.
- **ValidationBulkQueryTargetLatency**— AWS DMS can do data validation in bulk, especially in certain scenarios during a full-load or on-going replication when there are many changes. This metric indicates the latency required to read a bulk set of data on the target endpoint.
- **ValidationItemQuerySourceLatency**— During on-going replication, data validation can identify on-going changes and validate those changes. This metric indicates the latency in reading those changes from the source. Validation can run more queries than required, based on number of changes, if there are errors during validation.
- **ValidationItemQueryTargetLatency**— During on-going replication, data validation can identify on-going changes and validate the changes row by row. This metric gives us the latency in reading those changes from the target. Validation may run more queries than required, based on number of changes, if there are errors during validation.

To collect data validation information from CloudWatch enabled statistics, select **Enable CloudWatch logs** when you create or modify a task using the console. Then, to view the data validation information and ensure that your data was migrated accurately from source to target, do the following.

1. Choose the task on the **Database migration tasks** page.

2. Choose the **CloudWatch metrics** tab.
3. Select **Validation** from the drop down menu.

## Revalidating tables during a task

While a task is running, you can request AWS DMS to perform data validation.

### AWS Management Console

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.  
If you're signed in as an AWS Identity and Access Management (IAM) user, make sure that you have the appropriate permissions to access AWS DMS. The permissions required, see [IAM permissions needed to use AWS DMS \(p. 628\)](#).
2. Choose **Tasks** from the navigation pane.
3. Choose the running task that has the table you want to revalidate.
4. Choose the **Table Statistics** tab.
5. Choose the table you want to revalidate (you can choose up to 10 tables at one time). If the task is no longer running, you can't revalidate the table(s).
6. Choose **Revalidate**.

## Using JSON editor to modify validation rules

To add a validation rule to a task using the JSON editor from the AWS DMS Console, do the following:

1. Select **Database migration tasks**.
2. Select your task from the list of migration tasks.
3. If your task is running, select **Stop** from the **Actions** drop down menu.
4. Once the task has stopped, to modify your task, select **Modify** from the **Actions** drop down menu.
5. In the **Table mappings** section, select **JSON editor** and add your validation rule to your table mappings.

For example, you can add the following validation rule to run a replace function on the source. In this case, if the validation rule encounters a null byte, it validates it as a space.

```
{  
    "rule-type": "validation",  
    "rule-id": "1",  
    "rule-name": "1",  
    "rule-target": "column",  
    "object-locator": {  
        "schema-name": "Test-Schema",  
        "table-name": "Test-Table",  
        "column-name": "Test-Column"  
    },  
    "rule-action": "override-validation-function",  
    "source-function": "REPLACE(${column-name}, chr(0), chr(32))",  
    "target-function": "${column-name}"  
}
```

## Validation only tasks

You can create validation only tasks to preview and validate data without performing any migration or data replication. To create a validation only task, set the `EnableValidation` and `ValidationOnly` settings to `true`. When enabling `ValidationOnly`, additional requirements apply. For more information, see [Data validation task settings \(p. 467\)](#).

For a full load only migration type, a validation only task completes much faster than its CDC equivalent when many failures are reported. But changes to the source or target endpoint are reported as failures for full load mode, a possible disadvantage.

A CDC validation only task delays validation based on average latency, and retries failures multiple times before reporting them. If the majority of data comparisons result in failures, a validation only task for CDC mode is very slow, a potential drawback.

### Full load validation only

Beginning with AWS DMS version 3.4.6 and later, a full load validation only task quickly compares all rows from the source and target tables in a single pass, immediately reports any failures, and then shuts down. Validation never is suspended due to failures in this mode, it is optimized for speed. But changes to the source or target endpoint are reported as failures.

**Note**

Beginning with AWS DMS version 3.4.6 and later, this validation behavior also applies to full load migration task with validation enabled.

### CDC validation only

A CDC validation only task validates all existing rows between the source and target tables on a fresh start. In addition, a CDC validation only task runs continuously, re-validates ongoing replication changes, limits the number of failures reported each pass, and retries mismatched rows before failing them. It is optimized to prevent false positives.

Validation for a table (or the entire task) is suspended if the `FailureMaxCount` or `TableFailureMaxCount` thresholds are breached. This also applies for a CDC or Full Load+CDC migration task with validation enabled. And a CDC task with validation enabled delays re-validation for each changed row based on average source and target latency.

But a CDC *validation only task* doesn't migrate data and has no latency. It sets `ValidationQueryCdcDelaySeconds` to 180 by default. And you can increase the amount to account for high latency environments and help prevent false positives.

## Validation only use cases

Use cases for splitting the data validation portion of a migration or replication task into a separate *validation only task* includes, but is not limited to, the following:

- *Control exactly when validation occurs* — Validation queries add an additional load to both source and target endpoints. So, migrating or replicating data in one task first, then validating the results in another task can be beneficial.
- *Reduce load on the replication instance* — Splitting data validation to run on its own instance can be advantageous.
- *Quickly obtain how many rows don't match at a given moment in time* — For example, just before or during a maintenance window production cut-over to a target endpoint, you can create a Full Load validation only task to get an answer to your question.

- When validation failures are expected for a migration task with a CDC component — For example, if migrating Oracle varchar2 to PostgreSQL jsonb, CDC validation keeps retrying these failed rows and limits the number of failures reported each time. But, you can create a Full Load validation only task and obtain a quicker answer.
- You've developed a data repair script/utility that reads the validation failure table — (See also, [Troubleshooting \(p. 589\)](#)). A Full Load validation only task quickly reports failures for the data repair script to act upon.

For an example of ValidationSettings task settings in a JSON file, see [Task settings example \(p. 450\)](#).

## Troubleshooting

During validation, AWS DMS creates a new table at the target endpoint: `awsdms_validation_failures_v1`. If any record enters the `ValidationSuspended` or the `ValidationFailed` state, AWS DMS writes diagnostic information to `awsdms_validation_failures_v1`. You can query this table to help troubleshoot validation errors.

For information about changing the default schema the table is created in on the target, see [Control table task settings \(p. 461\)](#).

Following is a description of the `awsdms_validation_failures_v1` table:

Column name	Data type	Description
<code>TASK_NAME</code>	<code>VARCHAR(128) NOT NULL</code>	AWS DMS task identifier.
<code>TABLE_OWNER</code>	<code>VARCHAR(128) NOT NULL</code>	Schema (owner) of the table.
<code>TABLE_NAME</code>	<code>VARCHAR(128) NOT NULL</code>	Table name.
<code>FAILURE_TIME</code>	<code>DATETIME(3) NOT NULL</code>	Time when the failure occurred.
<code>KEY_TYPE</code>	<code>VARCHAR(128) NOT NULL</code>	Reserved for future use (value is always 'Row')
<code>KEY</code>	<code>TEXT NOT NULL</code>	This is the primary key for row record type.
<code>FAILURE_TYPE</code>	<code>VARCHAR(128) NOT NULL</code>	Severity of validation error. Can be either RECORD_DIFF, MISSING_SOURCE or MISSING_TARGET.
<code>DETAILS</code>	<code>VARCHAR(8000) NOT NULL</code>	JSON formatted string of all source/target column values which do not match for the given key.

The following query will show you all the failures for a task by querying the `awsdms_validation_failures_v1` table. The task name should be the external resource ID of the task. The external resource ID of the task is the last value in the task ARN. For example, for a task with an ARN value of `arn:aws:dms:us-west-2:5599:task: VFPFKH4FJR3FTYKK2RYSI`, the external resource ID of the task would be `VFPFKH4FJR3FTYKK2RYSI`.

```
select * from awsdms_validation_failures_v1 where TASK_NAME = 'VFPFKH4FJR3FTYKK2RYSI'

TASK_NAME      VFPFKH4FJR3FTYKK2RYSI
TABLE_OWNER    DB2PERF
TABLE_NAME     PERFTEST
FAILURE_TIME   2020-06-11 21:58:44
KEY_TYPE       Row
KEY            {"key": ["3451491"]}
FAILURE_TYPE   RECORD_DIFF
DETAILS        [{"MYREAL": '+1.10106036e-01'}, {'MYREAL': '+1.10106044e-01'}], ]
```

You can look at the DETAILS field to determine which columns don't match. Since you have the primary key of the failed record, you can query the source and target endpoints to see what part of the record does not match.

## Limitations

- Data validation requires that the table has a primary key or unique index.
  - Primary key columns can't be of type CLOB, BLOB, or BYTE.
  - For primary key columns of type VARCHAR or CHAR, the length must be less than 1024.
  - An Oracle key created with the NOVALIDATE clause is *not* considered a primary key or unique index.
  - For an Oracle table with no primary key and only a unique key, the columns with the unique constraint must also have a NOT NULL constraint.
- Validation of NULL PK/UK values aren't supported.
- If the collation of the primary key column in the target PostgreSQL instance isn't set to "C", the sort order of the primary key is different compared to the sort order in Oracle. If the sort order is different between PostgreSQL and Oracle, data validation fails to validate the records.
- Data validation generates additional queries against the source and target databases. You must ensure that both databases have enough resources to handle this additional load.
- Data validation isn't supported when consolidating several databases into one.
- For a source or target Oracle endpoint, AWS DMS uses DBMS\_CRYPTO to validate LOBs. If your Oracle endpoint uses LOBs, then you must grant the execute permission on dbms\_crypto to the user account used to access the Oracle endpoint. You can do this by running the following statement:

```
grant execute on sys.dbms_crypto to dms_endpoint_user;
```

- If the target database is modified outside of AWS DMS during validation, then discrepancies might not be reported accurately. This result can occur if one of your applications writes data to the target table, while AWS DMS is performing validation on that same table.
- If one or more rows are being continuously modified during validation, then AWS DMS can't validate those rows.
- If AWS DMS detects more than 10,000 failed or suspended records, it stops the validation. Before you proceed further, resolve any underlying problems with the data.
- AWS DMS doesn't support data validation of views.
- AWS DMS doesn't support data validation when character substitution task settings are used.

# Tagging resources in AWS Database Migration Service

You can use tags in AWS Database Migration Service (AWS DMS) to add metadata to your resources. In addition, you can use these tags with AWS Identity and Access Management (IAM) policies to manage access to AWS DMS resources and to control what actions can be applied to the AWS DMS resources. Finally, you can use these tags to track costs by grouping expenses for similarly tagged resources.

All AWS DMS resources can be tagged:

- Certificates
- Endpoints
- Event subscriptions
- Replication instances
- Replication subnet groups
- Replication tasks

An AWS DMS tag is a name-value pair that you define and associate with an AWS DMS resource. The name is referred to as the key. Supplying a value for the key is optional. You can use tags to assign arbitrary information to an AWS DMS resource. A tag key could be used, for example, to define a category, and the tag value could be a item in that category. For example, you could define a tag key of "project" and a tag value of "Salix", indicating that the AWS DMS resource is assigned to the Salix project. You could also use tags to designate AWS DMS resources as being used for test or production by using a key such as environment=test or environment =production. We recommend that you use a consistent set of tag keys to make it easier to track metadata associated with AWS DMS resources.

Use tags to organize your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. Then, to see the cost of combined resources, organize your billing information according to resources with the same tag key values. For example, you can tag several resources with a specific application name, and then organize your billing information to see the total cost of that application across several services. For more information, see [Using Cost Allocation Tags](#) in the [AWS Billing User Guide](#).

Each AWS DMS resource has a tag set, which contains all the tags that are assigned to that AWS DMS resource. A tag set can contain as many as ten tags, or it can be empty. If you add a tag to an AWS DMS resource that has the same key as an existing tag on resource, the new value overwrites the old value.

AWS does not apply any semantic meaning to your tags; tags are interpreted strictly as character strings. AWS DMS might set tags on an AWS DMS resource, depending on the settings that you use when you create the resource.

The following list describes the characteristics of an AWS DMS tag.

- The tag key is the required name of the tag. The string value can be from 1 to 128 Unicode characters in length and cannot be prefixed with "aws:" or "dms:". The string might contain only the set of Unicode letters, digits, white-space, '\_', '!', '/', '=', '+', '-' (Java regex: "^( [\u00p{L}]\u00p{Z}]\u00p{N}\_.:/=+\\[-]\* )\$").

- The tag value is an optional string value of the tag. The string value can be from 1 to 256 Unicode characters in length and cannot be prefixed with "aws:" or "dms:". The string might contain only the set of Unicode letters, digits, white-space, '\_', ':', '/', '=', '+', '-' (Java regex: " $^([\\p{L}\\p{Z}]\\p{N}_.:/+\\[-]*$)$ ").

Values do not have to be unique in a tag set and can be null. For example, you can have a key-value pair in a tag set of project/Trinity and cost-center/Trinity.

You can use the AWS CLI or the AWS DMS API to add, list, and delete tags on AWS DMS resources. When using the AWS CLI or the AWS DMS API, you must provide the Amazon Resource Name (ARN) for the AWS DMS resource you want to work with. For more information about constructing an ARN, see [Constructing an Amazon Resource Name \(ARN\) for AWS DMS \(p. 11\)](#).

Note that tags are cached for authorization purposes. Because of this, additions and updates to tags on AWS DMS resources might take several minutes before they are available.

## API

You can add, list, or remove tags for a AWS DMS resource using the AWS DMS API.

- To add a tag to an AWS DMS resource, use the [AddTagsToResource](#) operation.
- To list tags that are assigned to an AWS DMS resource, use the [ListTagsForResource](#) operation.
- To remove tags from an AWS DMS resource, use the [RemoveTagsFromResource](#) operation.

To learn more about how to construct the required ARN, see [Constructing an Amazon Resource Name \(ARN\) for AWS DMS \(p. 11\)](#).

When working with XML using the AWS DMS API, tags use the following schema:

```
<Tagging>
  <TagSet>
    <Tag>
      <Key>Project</Key>
      <Value>Trinity</Value>
    </Tag>
    <Tag>
      <Key>User</Key>
      <Value>Jones</Value>
    </Tag>
  </TagSet>
</Tagging>
```

The following table provides a list of the allowed XML tags and their characteristics. Note that values for Key and Value are case dependent. For example, project=Trinity and PROJECT=Trinity are two distinct tags.

Tagging element	Description
TagSet	A tag set is a container for all tags assigned to an Amazon RDS resource. There can be only one tag set per resource. You work with a TagSet only through the AWS DMS API.
Tag	A tag is a user-defined key-value pair. There can be from 1 to 10 tags in a tag set.

Tagging element	Description
Key	<p>A key is the required name of the tag. The string value can be from 1 to 128 Unicode characters in length and cannot be prefixed with "dms:" or "aws:". The string might only contain only the set of Unicode letters, digits, white-space, '_', '!', '/', '=', '+', '-' (Java regex: "<math>^([\u00p{L}\u00p{Z}\u00p{N}_.:/=+\u00p{-}]*)\$</math>").</p> <p>Keys must be unique to a tag set. For example, you cannot have a key-pair in a tag set with the key the same but with different values, such as project/Trinity and project/Xanadu.</p>
Value	<p>A value is the optional value of the tag. The string value can be from 1 to 256 Unicode characters in length and cannot be prefixed with "dms:" or "aws:". The string might only contain only the set of Unicode letters, digits, white-space, '_', '!', '/', '=', '+', '-' (Java regex: "<math>^([\u00p{L}\u00p{Z}\u00p{N}_.:/=+\u00p{-}]*)\$</math>").</p> <p>Values do not have to be unique in a tag set and can be null. For example, you can have a key-value pair in a tag set of project/Trinity and cost-center/Trinity.</p>

# Security in AWS Database Migration Service

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The [shared responsibility model](#) describes this as security of the cloud and security *in the cloud*:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the [AWS compliance programs](#). To learn about the compliance programs that apply to AWS DMS, 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 organization's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS DMS. The following topics show you how to configure AWS DMS to meet your security and compliance objectives. You also learn how to use other AWS services that help you monitor and secure your AWS DMS resources.

You can manage access to your AWS DMS resources and your databases (DBs). The method you use to manage access depends on the replication task you need to perform with AWS DMS:

- Use AWS Identity and Access Management (IAM) policies to assign permissions that determine who is allowed to manage AWS DMS resources. AWS DMS requires that you have the appropriate permissions if you sign in as an IAM user. For example, you can use IAM to determine who is allowed to create, describe, modify, and delete DB instances and clusters, tag resources, or modify security groups. For more information about IAM and using it with AWS DMS, see [Identity and access management for AWS Database Migration Service \(p. 598\)](#).
- AWS DMS uses Secure Sockets Layer (SSL) for your endpoint connections with Transport Layer Security (TLS). For more information about using SSL/TLS with AWS DMS, see [Using SSL with AWS Database Migration Service \(p. 647\)](#).
- AWS DMS uses AWS Key Management Service (AWS KMS) encryption keys to encrypt the storage used by your replication instance and its endpoint connection information. AWS DMS also uses AWS KMS encryption keys to secure your target data at rest for Amazon S3 and Amazon Redshift target endpoints. For more information, see [Setting an encryption key and specifying AWS KMS permissions \(p. 643\)](#).
- AWS DMS always creates your replication instance in a virtual private cloud (VPC) based on the Amazon VPC service for the greatest possible network access control. For your DB instances and instance clusters, use the same VPC as your replication instance, or additional VPCs to match this level of access control. Each Amazon VPC that you use must be associated with a security group that has rules that allow all traffic on all ports to leave (egress) the VPC. This approach allows communication from the replication instance to your source and target database endpoints, as long as correct ingress is enabled on those endpoints.

For more information about available network configurations for AWS DMS, see [Setting up a network for a replication instance \(p. 112\)](#). For more information about creating a DB instance or instance

cluster in a VPC, see the security and cluster management documentation for your Amazon databases at [AWS documentation](#). For more information about network configurations that AWS DMS supports, see [Setting up a network for a replication instance \(p. 112\)](#).

- To view database migration logs, you need the appropriate Amazon CloudWatch Logs permissions for the IAM role you are using. For more information about logging for AWS DMS, see [Monitoring replication tasks using Amazon CloudWatch \(p. 561\)](#).

## Topics

- [Data protection in AWS Database Migration Service \(p. 596\)](#)
- [Identity and access management for AWS Database Migration Service \(p. 598\)](#)
- [Compliance validation for AWS Database Migration Service \(p. 624\)](#)
- [Resilience in AWS Database Migration Service \(p. 625\)](#)
- [Infrastructure security in AWS Database Migration Service \(p. 626\)](#)
- [IAM permissions needed to use AWS DMS \(p. 628\)](#)
- [Creating the IAM roles to use with the AWS CLI and AWS DMS API \(p. 632\)](#)
- [Fine-grained access control using resource names and tags \(p. 636\)](#)
- [Setting an encryption key and specifying AWS KMS permissions \(p. 643\)](#)
- [Network security for AWS Database Migration Service \(p. 645\)](#)
- [Using SSL with AWS Database Migration Service \(p. 647\)](#)
- [Changing the database password \(p. 651\)](#)

# Data protection in AWS Database Migration Service

## Data encryption

You can enable encryption for data resources of supported AWS DMS target endpoints. AWS DMS also encrypts connections to AWS DMS and between AWS DMS and all its source and target endpoints. In addition, you can manage the keys that AWS DMS and its supported target endpoints use to enable this encryption.

### Topics

- [Encryption at rest \(p. 596\)](#)
- [Encryption in transit \(p. 596\)](#)
- [Key management \(p. 597\)](#)

## Encryption at rest

AWS DMS supports encryption at rest by allowing you to specify the server-side encryption mode that you want used to push your replicated data to Amazon S3 before it is copied to supported AWS DMS target endpoints. You can specify this encryption mode by setting the `encryptionMode` extra connection attribute for the endpoint. If this `encryptionMode` setting specifies KMS key encryption mode, you can also create custom AWS KMS keys specifically to encrypt the target data for the following AWS DMS target endpoints:

- Amazon Redshift – For more information about setting `encryptionMode`, see [Extra connection attributes when using Amazon Redshift as a target for AWS DMS \(p. 324\)](#). For more information about creating a custom AWS KMS encryption key, see [Creating and using AWS KMS keys to encrypt Amazon Redshift target data \(p. 319\)](#).
- Amazon S3 – For more information about setting `encryptionMode`, see [Extra connection attributes when using Amazon S3 as a target for AWS DMS \(p. 350\)](#). For more information about creating a custom AWS KMS encryption key, see [Creating AWS KMS keys to encrypt Amazon S3 target objects \(p. 341\)](#).

## Encryption in transit

AWS DMS supports encryption in transit by ensuring that the data it replicates moves securely from the source endpoint to the target endpoint. This includes encrypting an S3 bucket on the replication instance that your replication task uses for intermediate storage as the data moves through the replication pipeline. To encrypt task connections to source and target endpoints AWS DMS uses Secure Socket Layer (SSL) with Transport Layer Security (TLS). By encrypting connections to both endpoints, AWS DMS ensures that your data is secure as it moves both from the source endpoint to your replication task and from your task to the target endpoint. For more information about using SSL/TLS with AWS DMS, see [Using SSL with AWS Database Migration Service \(p. 647\)](#)

AWS DMS supports both default and custom keys to encrypt both intermediate replication storage and connection information. You manage these keys by using AWS KMS. For more information, see [Setting an encryption key and specifying AWS KMS permissions \(p. 643\)](#).

## Key management

AWS DMS supports default or custom keys to encrypt replication storage, connection information, and the target data storage for certain target endpoints. You manage these keys by using AWS KMS. For more information, see [Setting an encryption key and specifying AWS KMS permissions \(p. 643\)](#).

## Internetwork traffic privacy

Connections are provided with protection between AWS DMS and source and target endpoints in the same AWS Region, whether running on premises or as part of an AWS service in the cloud. (At least one endpoint, source or target, must run as part of an AWS service in the cloud.) This protection applies whether these components share the same virtual private cloud (VPC) or exist in separate VPCs, if the VPCs are all in the same AWS Region. For more information about the supported network configurations for AWS DMS, see [Setting up a network for a replication instance \(p. 112\)](#). For more information about the security considerations when using these network configurations, see [Network security for AWS Database Migration Service \(p. 645\)](#).

# Identity and access management for AWS Database Migration Service

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

## Topics

- [Audience \(p. 598\)](#)
- [Authenticating with identities \(p. 598\)](#)
- [Managing access using policies \(p. 600\)](#)
- [How AWS Database Migration Service works with IAM \(p. 602\)](#)
- [AWS Database Migration Service identity-based policy examples \(p. 606\)](#)
- [Resource-based policy examples for AWS KMS \(p. 612\)](#)
- [Using secrets to access AWS Database Migration Service endpoints \(p. 615\)](#)
- [Troubleshooting AWS Database Migration Service identity and access \(p. 621\)](#)

## Audience

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

**Service user** – If you use the AWS DMS service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more AWS DMS 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 DMS, see [Troubleshooting AWS Database Migration Service identity and access \(p. 621\)](#).

**Service administrator** – If you're in charge of AWS DMS resources at your company, you probably have full access to AWS DMS. It's your job to determine which AWS DMS 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 DMS, see [How AWS Database Migration Service works with IAM \(p. 602\)](#).

**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 DMS. To view example AWS DMS identity-based policies that you can use in IAM, see [AWS Database Migration Service identity-based policy examples \(p. 606\)](#).

## 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 (successor to AWS Single Sign-On) (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 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 [Signature Version 4 signing process](#) in the [AWS General Reference](#).

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 \(successor to AWS Single Sign-On\) User Guide](#) and [Using multi-factor authentication \(MFA\) in AWS](#) in the [IAM User Guide](#).

## AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the *AWS account root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see [Tasks that require root user credentials](#) in the [AWS Account Management Reference Guide](#).

## IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see [Rotate access keys regularly for use cases that require long-term credentials](#) in the [IAM User Guide](#).

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see [When to create an IAM user \(instead of a role\)](#) in the [IAM User Guide](#).

## IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by [switching roles](#). You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see [Using IAM roles](#) in the [IAM User Guide](#).

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for

federation, see [Creating a role for a third-party Identity Provider](#) in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see [Permission sets](#) in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide*.

- **Temporary IAM user permissions** – An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access** – You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see [How IAM roles differ from resource-based policies](#) in the *IAM User Guide*.
- **Cross-service access** – Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Principal permissions** – When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see [Actions, Resources, and Condition Keys for AWS Database Migration Service](#) in the *Service Authorization Reference*.
  - **Service role** – A service role is an [IAM role](#) that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see [Creating a role to delegate permissions to an AWS service](#) in the *IAM User Guide*.
  - **Service-linked role** – A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see [Using an IAM role to grant permissions to applications running on Amazon EC2 instances](#) in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see [When to create an IAM role \(instead of a user\)](#) in the *IAM User Guide*.

## Managing access using policies

You control access in AWS by creating policies and attaching them to 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**.

Every IAM entity (user or role) starts with no permissions. By default, users can do nothing, not even change their own password. To give a user permission to do something, an administrator must attach a permissions policy to a user. Or the administrator can add the user to a group that has the intended

permissions. When an administrator gives permissions to a group, all users in that group are granted those permissions.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

## Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see [Creating IAM policies](#) in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see [Choosing between managed policies and inline policies](#) in the *IAM User Guide*.

## Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must [specify a principal](#) in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

## Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see [Access control list \(ACL\) overview](#) in the *Amazon Simple Storage Service Developer Guide*.

## Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see [Permissions boundaries for IAM entities](#) in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all

features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see [How SCPs work](#) in the *AWS Organizations User Guide*.

- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see [Session policies](#) in the *IAM User Guide*.

## Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see [Policy evaluation logic](#) in the *IAM User Guide*.

# How AWS Database Migration Service works with IAM

Before you use IAM to manage access to AWS DMS, you should understand what IAM features are available to use with AWS DMS. To get a high-level view of how AWS DMS and other AWS services work with IAM, see [AWS services that work with IAM](#) in the *IAM User Guide*.

### Topics

- [AWS DMS identity-based policies \(p. 602\)](#)
- [AWS DMS resource-based policies \(p. 605\)](#)
- [Authorization based on AWS DMS tags \(p. 605\)](#)
- [IAM roles for AWS DMS \(p. 605\)](#)

## AWS DMS identity-based policies

With IAM identity-based policies, you can specify allowed or denied actions and resources, and also the conditions under which actions are allowed or denied. AWS DMS supports specific actions, resources, and condition keys. To learn about all of the elements that you use in a JSON policy, see [IAM JSON policy elements reference](#) in the *IAM User Guide*.

### Actions

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The Action element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

Policy actions in AWS DMS use the following prefix before the action: `dms:`. For example, to grant someone permission to create a replication task with the AWS DMS `CreateReplicationTask` API operation, you include the `dms:CreateReplicationTask` action in their policy. Policy statements must include either an Action or NotAction element. AWS DMS defines its own set of actions that describe tasks that you can perform with this service.

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

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

You can specify multiple actions using wildcards (\*). For example, to specify all actions that begin with the word `Describe`, include the following action.

```
"Action": "dms:Describe*"
```

To see a list of AWS DMS actions, see [Actions Defined by AWS Database Migration Service](#) in the *IAM User Guide*.

## Resources

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The Resource JSON policy element specifies the object or objects to which the action applies. Statements must include either a Resource or a NotResource element. As a best practice, specify a resource using its [Amazon Resource Name \(ARN\)](#). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (\*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

AWS DMS works with the following resources:

- Certificates
- Endpoints
- Event subscriptions
- Replication instances
- Replication subnet (security) groups
- Replication tasks

The resource or resources that AWS DMS requires depends on the action or actions that you invoke. You need a policy that permits these actions on the associated resource or resources specified by the resource ARNs.

For example, an AWS DMS endpoint resource has the following ARN:

```
arn:${Partition}:dms:${Region}:${Account}:endpoint/${InstanceId}
```

For more information about the format of ARNs, see [Amazon Resource Names \(ARNs\) and AWS service namespaces](#).

For example, to specify the 1A2B3C4D5E6F7G8H9I0J1K2L3M endpoint instance for the use-east-2 region in your statement, use the following ARN.

```
"Resource": "arn:aws:dms:us-east-2:987654321098:endpoint/1A2B3C4D5E6F7G8H9I0J1K2L3M"
```

To specify all endpoints that belong to a specific account, use the wildcard (\*).

```
"Resource": "arn:aws:dms:us-east-2:987654321098:endpoint/*"
```

Some AWS DMS actions, such as those for creating resources, cannot be performed on a specific resource. In those cases, you must use the wildcard (\*).

```
"Resource": "*"
```

Some AWS DMS API actions involve multiple resources. For example, `StartReplicationTask` starts and connects a replication task to two database endpoint resources, a source and a target, so an IAM user must have permissions to read the source endpoint and to write to the target endpoint. To specify multiple resources in a single statement, separate the ARNs with commas.

```
"Resource": [  
    "resource1",  
    "resource2" ]
```

For more information on controlling access to AWS DMS resources using policies, see [Using resource names to control access \(p. 636\)](#). To see a list of AWS DMS resource types and their ARNs, see [Resources Defined by AWS Database Migration Service](#) in the *IAM User Guide*. To learn with which actions you can specify the ARN of each resource, see [Actions Defined by AWS Database Migration Service](#).

## Condition keys

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The Condition element (or Condition *block*) lets you specify conditions in which a statement is in effect. The Condition element is optional. You can create conditional expressions that use [condition operators](#), such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple Condition elements in a statement, or multiple keys in a single Condition element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see [IAM policy elements: variables and tags](#) in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see [AWS global condition context keys](#) in the *IAM User Guide*.

AWS DMS defines its own set of condition keys and also supports using some global condition keys. To see all AWS global condition keys, see [AWS global condition context keys](#) in the *IAM User Guide*.

AWS DMS defines a set of standard tags that you can use in its condition keys and also allows you to define your own custom tags. For more information, see [Using tags to control access \(p. 638\)](#).

To see a list of AWS DMS condition keys, see [Condition Keys for AWS Database Migration Service](#) in the *IAM User Guide*. To learn with which actions and resources you can use a condition key, see [Actions Defined by AWS Database Migration Service](#) and [Resources Defined by AWS Database Migration Service](#).

## Examples

To view examples of AWS DMS identity-based policies, see [AWS Database Migration Service identity-based policy examples \(p. 606\)](#).

## AWS DMS resource-based policies

Resource-based policies are JSON policy documents that specify what actions a specified principal can perform on a given AWS DMS resource and under what conditions. AWS DMS supports resource-based permissions policies for AWS KMS encryption keys that you create to encrypt data migrated to supported target endpoints. The supported target endpoints include Amazon Redshift and Amazon S3. By using resource-based policies, you can grant the permission for using these encryption keys to other accounts for each target endpoint.

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, you must also grant the principal entity permission to access the resource. Grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see [How IAM roles differ from resource-based policies](#) in the *IAM User Guide*.

The AWS DMS service supports only one type of resource-based policy called a *key policy*, which is attached to an AWS KMS encryption key. This policy defines which principal entities (accounts, users, roles, and federated users) can encrypt migrated data on the supported target endpoint.

To learn how to attach a resource-based policy to an encryption key that you create for the supported target endpoints, see [Creating and using AWS KMS keys to encrypt Amazon Redshift target data \(p. 319\)](#) and [Creating AWS KMS keys to encrypt Amazon S3 target objects \(p. 341\)](#).

## Examples

For examples of AWS DMS resource-based policies, see [Resource-based policy examples for AWS KMS \(p. 612\)](#).

## Authorization based on AWS DMS tags

You can attach tags to AWS DMS resources or pass tags in a request to AWS DMS. To control access based on tags, you provide tag information in the [condition element](#) of a policy using the `dms:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition key. AWS DMS defines a set of standard tags that you can use in its condition keys and also enables you to define your own custom tags. For more information, see [Using tags to control access \(p. 638\)](#).

For an example identity-based policy that limits access to a resource based on tags, see [Accessing AWS DMS resources based on tags \(p. 612\)](#).

## IAM roles for AWS DMS

An [IAM role](#) is an entity within your AWS account that has specific permissions.

### Using temporary credentials with AWS DMS

You can use temporary credentials to sign in with federation, assume an IAM role, or assume a cross-account role. You get temporary security credentials by calling AWS STS API operations such as [AssumeRole](#) or [GetFederationToken](#).

AWS DMS supports using temporary credentials.

## Service-linked roles

**Service-linked roles** allow AWS services to access resources in other services to complete an action on your behalf. Service-linked roles appear in your IAM account and are owned by the service. An IAM administrator can view but not edit the permissions for service-linked roles.

AWS DMS does not support service-linked roles.

## Service roles

This feature allows a service to assume a **service role** on your behalf. This role allows the service to access resources in other services to complete an action on your behalf. Service roles appear in your IAM account and are owned by the account. This means that an IAM administrator can change the permissions for this role. However, doing so might break the functionality of the service.

AWS DMS supports two types of service roles that you must create to use certain source or target endpoints:

- Roles with permissions to allow AWS DMS access to the following source and target endpoints (or their resources):
  - Amazon DynamoDB as a target – For more information see [Prerequisites for using DynamoDB as a target for AWS Database Migration Service \(p. 363\)](#).
  - OpenSearch as a target – For more information see [Prerequisites for using Amazon OpenSearch Service as a target for AWS Database Migration Service \(p. 406\)](#).
  - Amazon Kinesis as a target – For more information see [Prerequisites for using a Kinesis data stream as a target for AWS Database Migration Service \(p. 381\)](#).
  - Amazon Redshift as a target – You need to create the specified role only for creating a custom KMS encryption key to encrypt the target data or for specifying a custom S3 bucket to hold intermediate task storage. For more information, see [Creating and using AWS KMS keys to encrypt Amazon Redshift target data \(p. 319\)](#) or [Amazon S3 bucket settings \(p. 324\)](#).
  - Amazon S3 as a source or as a target – For more information, see [Prerequisites when using Amazon S3 as a source for AWS DMS \(p. 258\)](#) or [Prerequisites for using Amazon S3 as a target \(p. 334\)](#).

For example, to read data from an S3 source endpoint or to push data to an S3 target endpoint, you must create a service role as a prerequisite to accessing S3 for each of these endpoint operations.

- Roles with permissions required to use the AWS CLI and AWS DMS API – Two IAM roles that you need to create are `dms-vpc-role` and `dms-cloudwatch-logs-role`. If you use Amazon Redshift as a target database, you must also create and add the IAM role `dms-access-for-endpoint` to your AWS account. For more information, see [Creating the IAM roles to use with the AWS CLI and AWS DMS API \(p. 632\)](#).

## Choosing an IAM role in AWS DMS

If you use the AWS CLI or the AWS DMS API for your database migration, you must add certain IAM roles to your AWS account before you can use the features of AWS DMS. Two of these are `dms-vpc-role` and `dms-cloudwatch-logs-role`. If you use Amazon Redshift as a target database, you must also add the IAM role `dms-access-for-endpoint` to your AWS account. For more information, see [Creating the IAM roles to use with the AWS CLI and AWS DMS API \(p. 632\)](#).

# AWS Database Migration Service identity-based policy examples

By default, IAM users and roles don't have permission to create or modify AWS DMS resources. They also can't perform tasks using the AWS Management Console, AWS CLI, or AWS API. An IAM administrator must create IAM policies that grant users and roles permission to perform specific API operations on

the specified resources they need. The administrator must then attach those policies to the IAM users or groups that require those permissions.

To learn how to create an IAM identity-based policy using these example JSON policy documents, see [Creating policies on the JSON tab](#) in the *IAM User Guide*.

#### Topics

- [Policy best practices \(p. 607\)](#)
- [Using the AWS DMS console \(p. 607\)](#)
- [Allow users to view their own permissions \(p. 610\)](#)
- [Accessing one Amazon S3 bucket \(p. 611\)](#)
- [Accessing AWS DMS resources based on tags \(p. 612\)](#)

## Policy best practices

Identity-based policies determine whether someone can create, access, or delete AWS DMS resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see [AWS managed policies](#) or [AWS managed policies for job functions](#) in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see [Policies and permissions in IAM](#) in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see [IAM JSON policy elements: Condition](#) in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see [IAM Access Analyzer policy validation](#) in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or root users in your account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see [Configuring MFA-protected API access](#) in the *IAM User Guide*.

For more information about best practices in IAM, see [Security best practices in IAM](#) in the *IAM User Guide*.

## Using the AWS DMS console

The following policy gives you access to AWS DMS, including the AWS DMS console, and also specifies permissions for certain actions needed from other Amazon services such as Amazon EC2.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "dms:*",
            "Resource": "arn:aws:dms:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms>ListAliases",
                "kms>DescribeKey"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam>GetRole",
                "iam>PassRole",
                "iam>CreateRole",
                "iam>AttachRolePolicy"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2>DescribeVpcs",
                "ec2>DescribeInternetGateways",
                "ec2>DescribeAvailabilityZones",
                "ec2>DescribeSubnets",
                "ec2>DescribeSecurityGroups",
                "ec2>ModifyNetworkInterfaceAttribute",
                "ec2>CreateNetworkInterface",
                "ec2>DeleteNetworkInterface"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:Get*",
                "cloudwatch>List*"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs>DescribeLogGroups",
                "logs>DescribeLogStreams",
                "logs>FilterLogEvents",
                "logs>GetLogEvents"
            ],
            "Resource": "arn:aws:service:region:account:resourcetype/id"
        },
        {
            "Effect": "Allow",
            "Action": [
                "redshift>Describe*",
                "redshift>ModifyClusterIamRoles"
            ],
            "Resource": "arn:aws:dms:region:account:resourcetype/id"
        }
    ]
}
```

```
    ]  
}
```

A breakdown of these permissions might help you better understand why each one required for using the console is necessary.

The following section is required to allow the user to list their available AWS KMS keys and alias for display in the console. This entry is not required if you know the Amazon Resource Name (ARN) for the KMS key and you are using only the AWS Command Line Interface (AWS CLI).

```
{  
    "Effect": "Allow",  
    "Action": [  
        "kms>ListAliases",  
        "kms>DescribeKey"  
    ],  
    "Resource": "arn:aws:service:region:account:resourcetype/id"  
}
```

The following section is required for certain endpoint types that require a role ARN to be passed in with the endpoint. In addition, if the required AWS DMS roles aren't created ahead of time, the AWS DMS console has the ability to create the role. If all roles are configured ahead of time, all that is required in `iam:GetRole` and `iam:PassRole`. For more information about roles, see [Creating the IAM roles to use with the AWS CLI and AWS DMS API \(p. 632\)](#).

```
{  
    "Effect": "Allow",  
    "Action": [  
        "iam>GetRole",  
        "iam>PassRole",  
        "iam>CreateRole",  
        "iam>AttachRolePolicy"  
    ],  
    "Resource": "arn:aws:service:region:account:resourcetype/id"  
}
```

The following section is required because AWS DMS needs to create the Amazon EC2 instance and configure the network for the replication instance that is created. These resources exist in the customer's account, so the ability to perform these actions on behalf of the customer is required.

```
{  
    "Effect": "Allow",  
    "Action": [  
        "ec2>DescribeVpcs",  
        "ec2>DescribeInternetGateways",  
        "ec2>DescribeAvailabilityZones",  
        "ec2>DescribeSubnets",  
        "ec2>DescribeSecurityGroups",  
        "ec2>ModifyNetworkInterfaceAttribute",  
        "ec2>CreateNetworkInterface",  
        "ec2>DeleteNetworkInterface"  
    ],  
    "Resource": "arn:aws:service:region:account:resourcetype/id"  
}
```

The following section is required to allow the user to be able to view replication instance metrics.

```
{
```

```

    "Effect": "Allow",
    "Action": [
        "cloudwatch:Get*",
        "cloudwatch>List*"
    ],
    "Resource": "arn:aws:service:region:account:resourcetype/id"
}

```

This section is required to allow the user to view replication logs.

```

{
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams",
        "logs:FilterLogEvents",
        "logs:GetLogEvents"
    ],
    "Resource": "arn:aws:service:region:account:resourcetype/id"
}

```

The following section is required when using Amazon Redshift as a target. It allows AWS DMS to validate that the Amazon Redshift cluster is set up properly for AWS DMS.

```

{
    "Effect": "Allow",
    "Action": [
        "redshift:Describe*",
        "redshift:ModifyClusterIamRoles"
    ],
    "Resource": "arn:aws:dms:region:account:resourcetype/id"
}

```

The AWS DMS console creates several roles that are automatically attached to your AWS account when you use the AWS DMS console. If you use the AWS Command Line Interface (AWS CLI) or the AWS DMS API for your migration, you need to add these roles to your account. For more information about adding these roles, see [Creating the IAM roles to use with the AWS CLI and AWS DMS API \(p. 632\)](#).

For more information on the requirements for using this policy to access AWS DMS, see [IAM permissions needed to use AWS DMS \(p. 628\)](#).

## Allow users to view their own permissions

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

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam>ListGroupsForUser",
                "iam>ListAttachedUserPolicies",
                "iam>ListUserPolicies",
                "iam:GetUser"
            ]
        }
    ]
}

```

```

        ],
        "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
        "Sid": "NavigateInConsole",
        "Effect": "Allow",
        "Action": [
            "iam:GetGroupPolicy",
            "iam:GetPolicyVersion",
            "iam:GetPolicy",
            "iam>ListAttachedGroupPolicies",
            "iam>ListGroupPolicies",
            "iam>ListPolicyVersions",
            "iam>ListPolicies",
            "iam>ListUsers"
        ],
        "Resource": "*"
    }
]
}

```

## Accessing one Amazon S3 bucket

AWS DMS uses Amazon S3 buckets as intermediate storage for database migration. Typically, AWS DMS manages default S3 buckets for this purpose. However, in certain cases, especially when you use the AWS CLI or the AWS DMS API, AWS DMS enables you to specify your own S3 bucket instead. For example, you can specify your own S3 bucket for migrating data to an Amazon Redshift target endpoint. In this case, you need to create a role with permissions based on the AWS-managed `AmazonDMSRedshiftS3Role` policy.

The following example shows a version of the `AmazonDMSRedshiftS3Role` policy. It allows AWS DMS to grant an IAM user in your AWS account access to one of your Amazon S3 buckets. It also allows the user to add, update, and delete objects.

In addition to granting the `s3:PutObject`, `s3:GetObject`, and `s3>DeleteObject` permissions to the user, the policy also grants the `s3>ListAllMyBuckets`, `s3:GetBucketLocation`, and `s3>ListBucket` permissions. These are the additional permissions required by the console. Other permissions allow AWS DMS to manage the bucket life cycle. Also, the `s3:GetObjectAcl` action is required to be able to copy objects.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3>CreateBucket",
                "s3>ListBucket",
                "s3>DeleteBucket",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:PutObject",
                "s3>DeleteObject",
                "s3:GetObjectVersion",
                "s3:GetBucketPolicy",
                "s3:PutBucketPolicy",
                "s3:GetBucketAcl",
                "s3:PutBucketVersioning",
                "s3:GetBucketVersioning",
                "s3:PutLifecycleConfiguration",
                "s3:GetLifecycleConfiguration",
                "s3>DeleteBucketPolicy"
            ]
        }
    ]
}

```

```
        ],
      "Resource": "arn:aws:s3:::dms-*"
    }
}
```

For more information on creating a role based on this policy, see [Amazon S3 bucket settings \(p. 324\)](#).

## Accessing AWS DMS resources based on tags

You can use conditions in your identity-based policy to control access to AWS DMS resources based on tags. This example shows how you might create a policy that allows access to all AWS DMS endpoints. However, permission is granted only if the endpoint database tag `Owner` has the value of that user's user name.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "dms:*",
      "Resource": "arn:aws:dms:*:*:endpoint/*",
      "Condition": {
        "StringEquals": {"dms:endpoint-tag/Owner": "${aws:username}"}
      }
    }
  ]
}
```

You can attach this policy to the IAM users in your account. If a user named `richard-roe` attempts to access an AWS DMS endpoint, the endpoint database must be tagged `Owner=richard-roe` or `owner=richard-roe`. Otherwise, this user is denied access. The condition tag key `Owner` matches both `Owner` and `owner` because condition key names are not case-sensitive. For more information, see [IAM JSON policy elements: Condition](#) in the *IAM User Guide*.

## Resource-based policy examples for AWS KMS

AWS DMS allows you to create custom AWS KMS encryption keys to encrypt supported target endpoint data. To learn how to create and attach a key policy to the encryption key you create for supported target data encryption, see [Creating and using AWS KMS keys to encrypt Amazon Redshift target data \(p. 319\)](#) and [Creating AWS KMS keys to encrypt Amazon S3 target objects \(p. 341\)](#).

### Topics

- [A policy for a custom AWS KMS encryption key to encrypt Amazon Redshift target data \(p. 612\)](#)
- [A policy for a custom AWS KMS encryption key to encrypt Amazon S3 target data \(p. 614\)](#)

## A policy for a custom AWS KMS encryption key to encrypt Amazon Redshift target data

The following example shows the JSON for the key policy created for an AWS KMS encryption key that you create to encrypt Amazon Redshift target data.

```
{
  "Id": "key-consolepolicy-3",
```

```

    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::987654321098:root"
                ]
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow access for Key Administrators",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::987654321098:role/Admin"
                ]
            },
            "Action": [
                "kms>Create*",
                "kms:Describe*",
                "kms:Enable*",
                "kms>List*",
                "kms:Put*",
                "kms:Update*",
                "kms:Revoke*",
                "kms:Disable*",
                "kms:Get*",
                "kms>Delete*",
                "kms:TagResource",
                "kms:UntagResource",
                "kms:ScheduleKeyDeletion",
                "kms:CancelKeyDeletion"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow use of the key",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::987654321098:role/DMS-Redshift-endpoint-access-role"
                ]
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey*",
                "kms:DescribeKey"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow attachment of persistent resources",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::987654321098:role/DMS-Redshift-endpoint-access-role"
                ]
            },
            "Action": [
                "kms>CreateGrant",

```

```

        "kms>ListGrants",
        "kms>RevokeGrant"
    ],
    "Resource": "*",
    "Condition": {
        "Bool": {
            "kms>GrantIsForAWSResource": true
        }
    }
}
]
}

```

Here, you can see where the key policy references the role for accessing Amazon Redshift target endpoint data that you created before creating the key. In the example, that is DMS-Redshift-endpoint-access-role. You can also see the different key actions permitted for the different principals (users and roles). For example, any user with DMS-Redshift-endpoint-access-role can encrypt, decrypt, and re-encrypt the target data. Such a user can also generate data keys for export to encrypt the data outside of AWS KMS. They can also return detailed information about a AWS KMS key, such as the key that you just created. In addition, such a user can manage attachments to AWS resources, such as the target endpoint.

## A policy for a custom AWS KMS encryption key to encrypt Amazon S3 target data

The following example shows the JSON for the key policy created for an AWS KMS encryption key that you create to encrypt Amazon S3 target data.

```

{
    "Id": "key-consolepolicy-3",
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable IAM User Permissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::987654321098:root"
                ]
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Allow access for Key Administrators",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::987654321098:role/Admin"
                ]
            },
            "Action": [
                "kms>Create*",
                "kms>Describe*",
                "kms>Enable*",
                "kms>List*",
                "kms>Put*",
                "kms>Update*",
                "kms>Revoke*",
                "kms>Disable*",
                "kms>Get*",
                "kms>Delete*",
                "kms>GenerateDataKey*"
            ]
        }
    ]
}

```

```

        "kms:TagResource",
        "kms:UntagResource",
        "kms:ScheduleKeyDeletion",
        "kms:CancelKeyDeletion"
    ],
    "Resource": "*"
},
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::987654321098:role/DMS-S3-endpoint-access-role"
        ]
    },
    "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey"
    ],
    "Resource": "*"
},
{
    "Sid": "Allow attachment of persistent resources",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::987654321098:role/DMS-S3-endpoint-access-role"
        ]
    },
    "Action": [
        "kms>CreateGrant",
        "kms>ListGrants",
        "kms:RevokeGrant"
    ],
    "Resource": "*",
    "Condition": {
        "Bool": {
            "kms:GrantIsForAWSResource": true
        }
    }
}
]

```

Here, you can see where the key policy references the role for accessing Amazon S3 target endpoint data that you created prior to creating the key. In the example, that is DMS-S3-endpoint-access-role. You can also see the different key actions permitted for the different principals (users and roles). For example, any user with DMS-S3-endpoint-access-role can encrypt, decrypt, and re-encrypt the target data. Such a user can also generate data keys for export to encrypt the data outside of AWS KMS. They can also return detailed information about a AWS KMS key, such as the key that you just created. In addition, such a user can manage attachment to AWS resources, such as the target endpoint.

## Using secrets to access AWS Database Migration Service endpoints

For AWS DMS, a *secret* is an encrypted key that you can use to represent a set of user credentials to authenticate, through *secret authentication*, the database connection for a supported AWS DMS source or target endpoint. For an Oracle endpoint that also uses Oracle Automatic Storage Management (ASM), AWS DMS requires an additional secret that represents the user credentials to access Oracle ASM.

You can create the secret or secrets that AWS DMS requires for secret authentication using AWS Secrets Manager, a service for securely creating, storing, and retrieving credentials to access applications, services, and IT resources in the cloud and on premise. This includes support for automatic periodic rotation of the encrypted secret value without your intervention, providing an extra level of security for your credentials. Enabling secret value rotation in AWS Secrets Manager also ensures that this secret value rotation happens without any effect on any database migration that relies on the secret. For secretly authenticating an endpoint database connection, create a secret whose identity or ARN you assign to `SecretsManagerSecretId`, which you include in your endpoint settings. For secretly authenticating Oracle ASM as part of an Oracle endpoint, create a secret whose identity or ARN you assign to `SecretsManagerOracleAsmSecretId`, which you also include in your endpoint settings.

For more information on AWS Secrets Manager, see [What Is AWS Secrets Manager?](#) in the *AWS Secrets Manager User Guide*.

AWS DMS supports secret authentication for the following on-premise or AWS-managed databases on supported source and target endpoints:

- Amazon DocumentDB
- IBM Db2 LUW
- Microsoft SQL Server
- MongoDB
- MySQL
- Oracle
- PostgreSQL
- Amazon Redshift
- SAP ASE

For connection to any of these databases, you have the choice of entering one of the following sets of values, but not both, as part of your endpoint settings:

- Clear-text values to authenticate the database connection using the `UserName`, `Password`, `ServerName`, and `Port` settings. For an Oracle endpoint that also uses Oracle ASM, include additional clear-text values to authenticate ASM using the `AsmUserName`, `AsmPassword`, and `AsmServerName` settings.
- Secret authentication using values for the `SecretsManagerSecretId` and `SecretsManagerAccessRoleArn` settings. For an Oracle endpoint using Oracle ASM, include additional values for the `SecretsManagerOracleAsmSecretId` and `SecretsManagerOracleAsmAccessRoleArn` settings. The secret values for these settings can include the following for:
  - `SecretsManagerSecretId` – The full Amazon Resource Name (ARN), partial ARN, or friendly name of a secret that you have created for endpoint database access in the AWS Secrets Manager.
  - `SecretsManagerAccessRoleArn` – The ARN of a secret access role that you have created in IAM to provide AWS DMS access to this `SecretsManagerSecretId` secret on your behalf.
  - `SecretsManagerOracleAsmSecretId` – The full Amazon Resource Name (ARN), partial ARN, or friendly name of a secret that you have created for Oracle ASM access in the AWS Secrets Manager.
  - `SecretsManagerOracleAsmAccessRoleArn` – The ARN of a secret access role that you have created in IAM to provide AWS DMS access to this `SecretsManagerOracleAsmSecretId` secret on your behalf.

**Note**

You can also use a single secret access role to provide AWS DMS access to both the `SecretsManagerSecretId` secret and the `SecretsManagerOracleAsmSecretId` secret. If you create this single secret access role for both secrets, ensure that you assign the same ARN for this access role to both `SecretsManagerAccessRoleArn` and

`SecretsManagerAccessRoleArn`. For example, if your secret access role for both secrets has its ARN assigned to the variable, `ARN2xsecrets`, you can set these ARN settings as follows:

```
SecretsManagerAccessRoleArn = ARN2xsecrets;
SecretsManagerOracleAsmAccessRoleArn = ARN2xsecrets;
```

For more information on creating these values, see [Using the AWS Management Console to create a secret and secret access role \(p. 618\)](#).

After you have created and specified the required secret and secret access-role endpoint settings for your endpoints, update the permissions on the user accounts that will run the `CreateEndpoint` or `ModifyEndpoint` API request with these secret details. Ensure that these account permissions include the `IAM:GetRole` permission on the secret access role and the `SecretsManager:DescribeSecret` permission on the secret. AWS DMS requires these permissions to validate both the access role and its secret.

### To provide and verify required user permissions

1. Sign in to the AWS Management Console and open the AWS Identity and Access Management console at <https://console.aws.amazon.com/iam/>.
2. Choose **Users**, then select the **User ID** used for making `CreateEndpoint` and `ModifyEndpoint` API calls.
3. From the **Permissions** tab, choose **JSON**.
4. Make sure the user has the permissions shown following.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "SECRET_ACCESS_ROLE_ARN"
    },
    {
      "Effect": "Allow",
      "Action": "secretsmanager:DescribeSecret",
      "Resource": "SECRET_ARN"
    }
  ]
}
```

5. If the user doesn't have those permission, add the permissions.
6. If you're using an IAM Role for making DMS API calls, repeat the steps above for the respective role.
7. Open a terminal and use the AWS CLI to validate that permissions are given correctly by assuming the Role or User used above.
  - a. Validate user's permission on the `SecretAccessRole` using the `iam get-role` command.

```
aws iam get-role --role-name ROLE_NAME
```

Replace `ROLE_NAME` with the name of `SecretsManagerAccessRole`.

If the command returns an error message, make sure the permissions were given correctly.

- b. Validate user's permission on the secret using the Secrets Manager `describe-secret` command.

```
aws secretsmanager describe-secret --secret-id SECRET_NAME OR SECRET_ARN --region=REGION_NAME
```

User can be the friendly name, partial ARN or the full ARN. For more information, see [describe-secret](#).

If the command returns an error message, make sure the permissions were given correctly.

## Using the AWS Management Console to create a secret and secret access role

You can use the AWS Management Console to create a secret for endpoint authentication and to create the policy and role to allow AWS DMS to access the secret on your behalf.

### To create a secret using the AWS Management Console that AWS DMS can use to authenticate a database for source and target endpoint connections

1. Sign in to the AWS Management Console and open the AWS Secrets Manager console at <https://console.aws.amazon.com/secretsmanager/>.
2. Choose **Store a new secret**.
3. Under **Select secret type** on the **Store a new secret** page, choose **Other type of secrets**, then choose **Plaintext**.

#### Note

This is the only place that you need to enter clear text credentials to connect to your endpoint database from this point forward.

4. In the **Plaintext** field:

- For a secret whose identity you assign to `SecretManagerSecretId`, enter the following JSON structure.

```
{  
    "username": db_username,  
    "password": db_user_password,  
    "port": db_port_number,  
    "host": db_server_name  
}
```

#### Note

This is the minimum list of JSON members required to authenticate the endpoint database. You can add any additional JSON endpoint settings as JSON members in all lower case that you want. However, AWS DMS ignores any additional JSON members for endpoint authentication.

Here, *db\_username* is the name of the user accessing the database, *db\_user\_password* is the password of the database user, *db\_port\_number* is the port number to access the database, and *db\_server\_name* is the database server name (address) on the web, as in the following example.

```
{  
    "username": "admin",  
    "password": "some_password",  
}
```

```
    "port": "8190",
    "host": "oracle101.abcdefgij.us-east-1.rds.amazonaws.com"
}
```

- For a secret whose identity you assign to `SecretsManagerOracleAsmSecretId`, enter the following JSON structure.

```
{
  "asm_user": asm_username,
  "asm_password": asm_user_password,
  "asm_server": asm_server_name
}
```

**Note**

This is the minimum list of JSON members required to authenticate Oracle ASM for an Oracle endpoint. It is also the complete list that you can specify based on the available Oracle ASM endpoint settings.

Here, `asm_username` is the name of the user accessing Oracle ASM, `asm_user_password` is the password of the Oracle ASM user, and `asm_server_name` is the Oracle ASM server name (address) on the web, including the port, as in the following example.

```
{
  "asm_user": "oracle_asm_user",
  "asm_password": "oracle_asm_password",
  "asm_server": "oracle101.abcdefgij.us-east-1.rds.amazonaws.com:8190/+ASM"
```

5. Select an AWS KMS encryption key to encrypt the secret. You can accept the default encryption key created for your service by AWS Secrets Manager or select a AWS KMS key that you create.
6. Specify a name to reference this secret and an optional description. This is the friendly name that you use as the value for `SecretsManagerSecretId` or `SecretsManagerOracleAsmSecretId`.
7. If you want to enable automatic rotation in the secret, you need to select or create an AWS Lambda function with permission to rotate the credentials for the secret as described. However, before setting automatic rotation to use your Lambda function, ensure that the configuration settings for the function add the following four characters to the value of the EXCLUDE\_CHARACTERS environment variable.

```
; . : +
```

AWS DMS doesn't allow these characters in passwords used for endpoint credentials. Configuring your Lambda function to exclude them prevents AWS Secrets Manager from generating these characters as part of its rotated password values. After you set automatic rotation to use your Lambda function, AWS Secrets Manager immediately rotates the secret to validate your secret configuration.

**Note**

Depending on your database engine configuration, your database might not fetch the rotated credentials. In this case, you need to manually restart the task to refresh the credentials.

8. Review and store your secret in AWS Secrets Manager. You can then look up each secret by its friendly name in AWS Secrets Manager, then retrieve the secret ARN as the value for `SecretsManagerSecretId` or `SecretsManagerOracleAsmSecretId` as appropriate to authenticate access to your endpoint database connection and Oracle ASM (if used).

**To create the secret access policy and role to set your `SecretsManagerAccessRoleArn` or `SecretsManagerOracleAsmAccessRoleArn`, which allows AWS DMS to access AWS Secrets Manager to access your appropriate secret**

1. Sign in to the AWS Management Console and open the AWS Identity and Access Management (IAM) console at <https://console.aws.amazon.com/iam/>.
2. Choose **Policies**, then choose **Create policy**.
3. Choose **JSON** and enter the following policy to enable access to and decryption of your secret.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": secret_arn,
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": kms_key_arn,
        }
    ]
}
```

Here, *secret\_arn* is the ARN of your secret, which you can get from either `SecretsManagerSecretId` or `SecretsManagerOracleAsmSecretId` as appropriate, and *kms\_key\_arn* is the ARN of the AWS KMS key that you are using to encrypt your secret, as in the following example.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "arn:aws:secretsmanager:us-east-2:123456789012:secret:MySQLTestSecret-qeHamH"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-east-2:123456789012:key/761138dc-0542-4e58-947f-4a3a8458d0fd"
        }
    ]
}
```

#### Note

If you use the default encryption key created by AWS Secrets Manager, you do not have to specify the AWS KMS permissions for *kms\_key\_arn*.

If you want your policy to provide access to both secrets, simply specify an additional JSON resource object for the other *secret\_arn*.

4. Review and create the policy with a friendly name and optional description.
5. Choose **Roles**, then choose **Create role**.
6. Choose **AWS service** as the type of trusted entity.
7. Choose **DMS** from the list of services as the trusted service, then choose **Next: Permissions**.
8. Look up and attach the policy you created in step 4, then proceed through adding any tags and review your role. At this point, edit the trust relationships for the role to use your AWS DMS regional service principal as the trusted entity. This principal has the following format.

```
dms.region-name.amazonaws.com
```

Here, *region-name* is the name of your region, such as us-east-1. Thus, an AWS DMS regional service principal for this region follows.

```
dms.us-east-1.amazonaws.com
```

9. After editing the trusted entity for the role, create the role with a friendly name and optional description. You can now look up your new role by its friendly name in IAM, then retrieve the role ARN as the `SecretsManagerAccessRoleArn` or `SecretsManagerOracleAsmAccessRoleArn` value to authenticate your endpoint database connection.

### To use secrets manager with a replication instance in a private subnet

1. Create a secret manager VPC endpoint and note the DNS for the endpoint. For more information about creating a secrets manager VPC endpoint, see [Connecting to Secrets Manager through a VPC endpoint](#) in the *AWS Secrets Manager User Guide*.
2. Attach the replication instance security group to the secret manager VPC endpoint.
3. For the replication instance security group egress rules, allow all traffic for destination `0.0.0.0/0`.
4. Set the endpoint extra connection attribute, `secretsManagerEndpointOverride=secretsManager endpoint DNS` to provide the secret manager VPC endpoint DNS, as shown in the following example.

```
secretsManagerEndpointOverride=vpce-1234a5678b9012c-12345678.secretsmanager.eu-west-1.vpce.amazonaws.com
```

## Troubleshooting AWS Database Migration Service identity and access

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

### Topics

- [I am not authorized to perform an action in AWS DMS \(p. 622\)](#)
- [I am not authorized to perform iam:PassRole \(p. 622\)](#)
- [I want to view my access keys \(p. 622\)](#)
- [I'm an administrator and want to allow others to access AWS DMS \(p. 623\)](#)
- [I want to allow people outside of my AWS account to access my AWS DMS resources \(p. 623\)](#)

## I am not authorized to perform an action in AWS DMS

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about an AWS DMS endpoint but does not have `dms:DescribeEndpoint` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:  
dms:DescribeEndpoint on resource: my-postgresql-target
```

In this case, Mateo asks his administrator to update his policies to allow him to access the `my-postgresql-target` endpoint resource using the `dms:DescribeEndpoint` action.

## I am not authorized to perform iam:PassRole

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

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 DMS. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

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

## I want to view my access keys

After you create your IAM user access keys, you can view your access key ID at any time. However, you can't view your secret access key again. If you lose your secret key, you must create a new access key pair.

Access keys consist of two parts: an access key ID (for example, `AKIAIOSFODNN7EXAMPLE`) and a secret access key (for example, `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`). Like a user name and password, you must use both the access key ID and secret access key together to authenticate your requests. Manage your access keys as securely as you do your user name and password.

### Important

Do not provide your access keys to a third party, even to help [find your canonical user ID](#). By doing this, you might give someone permanent access to your account.

When you create an access key pair, you are prompted to save the access key ID and secret access key in a secure location. The secret access key is available only at the time you create it. If you lose your secret access key, you must add new access keys to your IAM user. You can have a maximum of two access keys. If you already have two, you must delete one key pair before creating a new one. To view instructions, see [Managing access keys](#) in the *IAM User Guide*.

## I'm an administrator and want to allow others to access AWS DMS

To allow others to access AWS DMS, you must create an IAM entity (user or role) for the person or application that needs access. They will use the credentials for that entity to access AWS. You must then attach a policy to the entity that grants them the correct permissions in AWS DMS.

To get started right away, see [Creating your first IAM delegated user and group](#) in the *IAM User Guide*.

## I want to allow people outside of my AWS account to access my AWS DMS 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 DMS supports these features, see [How AWS Database Migration Service works with IAM \(p. 602\)](#).
- To learn how to provide access to your resources across AWS accounts that you own, see [Providing access to an IAM user in another AWS account that you own](#) in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see [Providing access to AWS accounts owned by third parties](#) in the *IAM User Guide*.
- To learn how to provide access through identity federation, see [Providing access to externally authenticated users \(identity federation\)](#) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see [How IAM roles differ from resource-based policies](#) in the *IAM User Guide*.

# Compliance validation for AWS Database Migration Service

Third-party auditors assess the security and compliance of AWS Database Migration Service as part of multiple AWS compliance programs. These include the following programs:

- SOC
- PCI
- ISO
- FedRAMP
- DoD CC SRG
- HIPAA BAA
- MTCS
- CS
- K-ISMS
- ENS High
- OSPAR
- HITRUST CSF

For a list of AWS services in scope of specific compliance programs, see [AWS services in scope by compliance program](#). For general information, see [AWS compliance programs](#).

You can download third-party audit reports using AWS Artifact. For more information, see [Downloading reports in AWS artifact](#).

Your compliance responsibility when using AWS DMS is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- [Security and compliance quick start guides](#) – These deployment guides discuss architectural considerations and provide steps for deploying security- and compliance-focused baseline environments on AWS.
- [Architecting for HIPAA security and compliance whitepaper](#) – This whitepaper describes how companies can use AWS to create HIPAA-compliant applications.
- [AWS compliance resources](#) – This collection of workbooks and guides might apply to your industry and location.
- [AWS Config](#) – This AWS service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#) – This AWS service provides a comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

# Resilience in AWS Database Migration Service

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 Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS global infrastructure](#).

In addition to the AWS global infrastructure, AWS DMS provides high availability and failover support for a replication instance using a Multi-AZ deployment when you choose the **Multi-AZ** option.

In a Multi-AZ deployment, AWS DMS automatically provisions and maintains a standby replica of the replication instance in a different Availability Zone. The primary replication instance is synchronously replicated to the standby replica. If the primary replication instance fails or becomes unresponsive, the standby resumes any running tasks with minimal interruption. Because the primary is constantly replicating its state to the standby, Multi-AZ deployment does incur some performance overhead.

For more information on working with Multi-AZ deployments, see [Working with an AWS DMS replication instance \(p. 100\)](#).

# Infrastructure security in AWS Database Migration Service

As a managed service, AWS Database Migration Service is protected by the AWS global network security procedures that are described in the [Amazon Web Services: Overview of security processes](#) whitepaper.

You use AWS published API calls to access AWS DMS through the network. Clients must support Transport Layer Security (TLS) 1.0 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an AWS Identity and Access Management (IAM) principal. Or you can use the [AWS Security Token Service](#) (AWS STS) to generate temporary security credentials to sign requests.

You can call these API operations from any network location. AWS DMS also supports resource-based access policies, which can specify restrictions on actions and resources, for example, based on the source IP address. In addition, you can use AWS DMS policies to control access from specific Amazon VPC endpoints or specific virtual private clouds (VPCs). Effectively, this isolates network access to a given AWS DMS resource from only the specific VPC within the AWS network. For more information about using resource-based access policies with AWS DMS, including examples, see [Fine-grained access control using resource names and tags \(p. 636\)](#).

To confine your communications with AWS DMS within a single VPC, you can create a VPC interface endpoint that enables you to connect to AWS DMS through AWS PrivateLink. AWS PrivateLink helps ensure that any call to AWS DMS and its associated results remain confined to the specific VPC for which your interface endpoint is created. You can then specify the URL for this interface endpoint as an option with every AWS DMS command that you run using the AWS CLI or an SDK. Doing this helps ensure that your entire communications with AWS DMS remain confined to the VPC and are otherwise invisible to the public internet.

## To create an interface endpoint to access DMS in a single VPC

1. Sign in to the AWS Management Console and open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
2. From the navigation pane, choose **Endpoints**. This opens the **Create endpoints** page, where you can create the interface endpoint from a VPC to AWS DMS.
3. Choose **AWS services**, then search for and choose a value for **Service Name**, in this case AWS DMS in the following form.

```
com.amazonaws.region.dms
```

Here, *region* specifies the AWS Region where AWS DMS runs, for example com.amazonaws.us-west-2.dms.

4. For **VPC**, choose the VPC to create the interface endpoint from, for example vpc-12abcd34.
5. Choose a value for **Availability Zone** and for **Subnet ID**. These values should indicate a location where your chosen AWS DMS endpoint can run, for example us-west-2a (usw2-az1) and subnet-ab123cd4.
6. Choose **Enable DNS name** to create the endpoint with a DNS name. This DNS name consists of the endpoint ID (vpce-12abcd34efg567hij) hyphenated with a random string (ab12dc34). These are separated from the service name by a dot in reverse dot-separated order, with vpce added (dms.us-west-2.vpce.amazonaws.com).

An example is vpce-12abcd34efg567hij-ab12dc34.dms.us-west-2.vpce.amazonaws.com.

7. For **Security group**, choose a group to use for the endpoint.

When you set up your security group, make sure to allow outbound HTTPS calls from within it. For more information, see [Creating security groups](#) in the *Amazon VPC User Guide*.

8. Choose either **Full Access** or a custom value for **Policy**. For example, you might choose a custom policy similar to the following that restricts your endpoint's access to certain actions and resources.

```
{  
  "Statement": [  
    {  
      "Action": "dms:*",  
      "Effect": "Allow",  
      "Resource": "*",  
      "Principal": "*"  
    },  
    {  
      "Action": [  
        "dms:ModifyReplicationInstance",  
        "dms:DeleteReplicationInstance"  
      ],  
      "Effect": "Deny",  
      "Resource": "arn:aws:dms:us-west-2:<account-id>:rep:<replication-instance-id>",  
      "Principal": "*"  
    }  
  ]  
}
```

Here, the sample policy allows any AWS DMS API call, except for deleting or modifying a specific replication instance.

You can now specify a URL formed using the DNS name created in step 6 as an option. You specify this for every AWS DMS CLI command or API operation to access the service instance using the created interface endpoint. For example, you might run the DMS CLI command `DescribeEndpoints` in this VPC as shown following.

```
$ aws dms describe-endpoints --endpoint-url https://vpce-12abcd34efg567hij-ab12dc34.dms.us-west-2.vpce.amazonaws.com
```

If you enable the private DNS option, you don't have to specify the endpoint URL in the request.

For more information on creating and using VPC interface endpoints (including enabling the private DNS option), see [Interface VPC endpoints \(AWS PrivateLink\)](#) in the *Amazon VPC User Guide*.

## IAM permissions needed to use AWS DMS

You use certain IAM permissions and IAM roles to use AWS DMS. If you are signed in as an IAM user and want to use AWS DMS, your account administrator must attach the policy discussed in this section to the IAM user, group, or role that you use to run AWS DMS. For more information about IAM permissions, see the [IAM User Guide](#).

The following policy gives you access to AWS DMS, and also permissions for certain actions needed from other Amazon services such as AWS KMS, IAM, Amazon EC2, and Amazon CloudWatch. CloudWatch monitors your AWS DMS migration in real time and collects and tracks metrics that indicate the progress of your migration. You can use CloudWatch Logs to debug problems with a task.

**Note**

You can further restrict access to AWS DMS resources using tagging. For more information about restricting access to AWS DMS resources using tagging, see [Fine-grained access control using resource names and tags \(p. 636\)](#).

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Action": "dms:*",  
            "Resource": "arn:aws:dms:region:account:resourcetype/id"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "kms>ListAliases",  
                "kmsDescribeKey"  
            ],  
            "Resource": "arn:aws:service:region:account:resourcetype/id"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "iamGetRole",  
                "iamPassRole",  
                "iamCreateRole",  
                "iamAttachRolePolicy"  
            ],  
            "Resource": "arn:aws:service:region:account:resourcetype/id"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "ec2DescribeVpcs",  
                "ec2DescribeInternetGateways",  
                "ec2DescribeAvailabilityZones",  
                "ec2DescribeSubnets",  
                "ec2DescribeSecurityGroups",  
                "ec2ModifyNetworkInterfaceAttribute",  
                "ec2CreateNetworkInterface",  
                "ec2DeleteNetworkInterface"  
            ],  
            "Resource": "arn:aws:service:region:account:resourcetype/id"  
        },  
        {  
            "Effect": "Allow",  
            "Action": [  
                "cloudwatchGet*",  
                "cloudwatchList*"  
            ]  
        }  
    ]  
}
```

```

        ],
        "Resource": "arn:aws:service:region:account:resourcetype/id"
    },
    {
        "Effect": "Allow",
        "Action": [
            "logs:DescribeLogGroups",
            "logs:DescribeLogStreams",
            "logs:FilterLogEvents",
            "logs:GetLogEvents"
        ],
        "Resource": "arn:aws:service:region:account:resourcetype/id"
    },
    {
        "Effect": "Allow",
        "Action": [
            "redshift:Describe*",
            "redshift:ModifyClusterIamRoles"
        ],
        "Resource": "arn:aws:dms:region:account:resourcetype/id"
    }
]
}

```

The breakdown of these following permissions might help you better understand why each one is necessary.

The following section is required to allow the user to call AWS DMS API operations.

```
{
    "Effect": "Allow",
    "Action": "dms:*",
    "Resource": "arn:aws:dms:region:account:resourcetype/id"
}
```

The following section is required to allow the user to list their available AWS KMS keys and alias for display in the console. This entry is not required if you know the Amazon Resource Name (ARN) for the KMS key and you are using only the AWS Command Line Interface (AWS CLI).

```
{
    "Effect": "Allow",
    "Action": [
        "kms>ListAliases",
        "kms>DescribeKey"
    ],
    "Resource": "arn:aws:service:region:account:resourcetype/id"
}
```

The following section is required for certain endpoint types that require an IAM role ARN to be passed in with the endpoint. In addition, if the required AWS DMS roles aren't created ahead of time, the AWS DMS console can create the role. If all roles are configured ahead of time, all that is required is `iam:GetRole` and `iam:PassRole`. For more information about roles, see [Creating the IAM roles to use with the AWS CLI and AWS DMS API \(p. 632\)](#).

```
{
    "Effect": "Allow",
    "Action": [
        "iam:GetRole",
        "iam:PassRole",

```

```

        "iam:CreateRole",
        "iam:AttachRolePolicy"
    ],
    "Resource": "arn:aws:service:region:account:resourcetype/id"
}

```

The following section is required because AWS DMS needs to create the Amazon EC2 instance and configure the network for the replication instance that is created. These resources exist in the customer's account, so the ability to perform these actions on behalf of the customer is required.

```

{
    "Effect": "Allow",
    "Action": [
        "ec2:DescribeVpcs",
        "ec2:DescribeInternetGateways",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:ModifyNetworkInterfaceAttribute",
        "ec2:CreateNetworkInterface",
        "ec2:DeleteNetworkInterface"
    ],
    "Resource": "arn:aws:service:region:account:resourcetype/id"
}

```

The following section is required to allow the user to be able to view replication instance metrics.

```

{
    "Effect": "Allow",
    "Action": [
        "cloudwatch:Get*",
        "cloudwatch>List*"
    ],
    "Resource": "arn:aws:service:region:account:resourcetype/id"
}

```

This section is required to allow the user to view replication logs.

```

{
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams",
        "logs:FilterLogEvents",
        "logs:GetLogEvents"
    ],
    "Resource": "arn:aws:service:region:account:resourcetype/id"
}

```

The following section is required when using Amazon Redshift as a target. It allows AWS DMS to validate that the Amazon Redshift cluster is set up properly for AWS DMS.

```

{
    "Effect": "Allow",
    "Action": [
        "redshift:Describe*",
        "redshift:ModifyClusterIamRoles"
    ],
    "Resource": "arn:aws:dms:region:account:resourcetype/id"
}

```

The AWS DMS console creates several roles that are automatically attached to your AWS account when you use the AWS DMS console. If you use the AWS Command Line Interface (AWS CLI) or the AWS DMS API for your migration, you need to add these roles to your account. For more information about adding these roles, see [Creating the IAM roles to use with the AWS CLI and AWS DMS API \(p. 632\)](#).

# Creating the IAM roles to use with the AWS CLI and AWS DMS API

If you use the AWS CLI or the AWS DMS API for your database migration, you must add three IAM roles to your AWS account before you can use the features of AWS DMS. Two of these are `dms-vpc-role` and `dms-cloudwatch-logs-role`. If you use Amazon Redshift as a target database, you must also add the IAM role `dms-access-for-endpoint` to your AWS account.

Updates to managed policies are automatic. If you are using a custom policy with the IAM roles, be sure to periodically check for updates to the managed policy in this documentation. You can view the details of the managed policy by using a combination of the `get-policy` and `get-policy-version` commands.

For example, the following `get-policy` command retrieves information about the specified IAM role.

```
aws iam get-policy --policy-arn arn:aws:iam::aws:policy/service-role/  
AmazonDMSVPCManagementRole
```

The information returned from the command is as follows.

```
{  
    "Policy": {  
        "PolicyName": "AmazonDMSVPCManagementRole",  
        "Description": "Provides access to manage VPC settings for AWS managed customer  
configurations",  
        "CreateDate": "2015-11-18T16:33:19Z",  
        "AttachmentCount": 1,  
        "IsAttachable": true,  
        "PolicyId": "ANPAJHKIGMBQI4AEFFSY0",  
        "DefaultVersionId": "v3",  
        "Path": "/service-role/",  
        "Arn": "arn:aws:iam::aws:policy/service-role/AmazonDMSVPCManagementRole",  
        "UpdateDate": "2016-05-23T16:29:57Z"  
    }  
}
```

The following `get-policy-version` command retrieves IAM policy information.

```
aws iam get-policy-version --policy-arn arn:aws:iam::aws:policy/service-role/  
AmazonDMSVPCManagementRole --version-id v3
```

The information returned from the command is as follows.

```
{  
    "PolicyVersion": {  
        "CreateDate": "2016-05-23T16:29:57Z",  
        "VersionId": "v3",  
        "Document": {  
            "Version": "2012-10-17",  
            "Statement": [  
                {  
                    "Action": "sts:AssumeRole",  
                    "Effect": "Allow",  
                    "Principal": "*"  
                }  
            ]  
        }  
    }  
}
```

```
{
    "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:DescribeAvailabilityZones",
        "ec2:DescribeInternetGateways",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSubnets",
        "ec2:DescribeVpcs",
        "ec2:DeleteNetworkInterface",
        "ec2:ModifyNetworkInterfaceAttribute"
    ],
    "Resource": "arn:aws:service:region:account:resourcetype/id",
    "Effect": "Allow"
}
]
},
"IsDefaultVersion": true
}
}
```

You can use the same commands to get information about AmazonDMSCloudWatchLogsRole and the AmazonDMSRedshiftS3Role managed policy.

**Note**

If you use the AWS DMS console for your database migration, these roles are added to your AWS account automatically.

The following procedures create the dms-vpc-role, dms-cloudwatch-logs-role, and dms-access-for-endpoint IAM roles.

#### To create the dms-vpc-role IAM role for use with the AWS CLI or AWS DMS API

1. Create a JSON file with the following IAM policy. Name the JSON file dmsAssumeRolePolicyDocument.json.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "dms.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Create the role using the AWS CLI using the following command.

```
aws iam create-role --role-name dms-vpc-role --assume-role-policy-document file://dmsAssumeRolePolicyDocument.json
```

2. Attach the AmazonDMSVPCManagementRole policy to dms-vpc-role using the following command.

```
aws iam attach-role-policy --role-name dms-vpc-role --policy-arn arn:aws:iam::aws:policy/service-role/AmazonDMSVPCManagementRole
```

### To create the dms-cloudwatch-logs-role IAM role for use with the AWS CLI or AWS DMS API

1. Create a JSON file with the following IAM policy. Name the JSON file dmsAssumeRolePolicyDocument2.json.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "dms.amazonaws.com"  
            },  
            "Action": "sts:AssumeRole"  
        }  
    ]  
}
```

Create the role using the AWS CLI using the following command.

```
aws iam create-role --role-name dms-cloudwatch-logs-role --assume-role-policy-document file://dmsAssumeRolePolicyDocument2.json
```

2. Attach the AmazonDMSCloudWatchLogsRole policy to dms-cloudwatch-logs-role using the following command.

```
aws iam attach-role-policy --role-name dms-cloudwatch-logs-role --policy-arn arn:aws:iam::aws:policy/service-role/AmazonDMSCloudWatchLogsRole
```

If you use Amazon Redshift as your target database, you must create the IAM role dms-access-for-endpoint to provide access to Amazon S3.

### To create the dms-access-for-endpoint IAM role for use with Amazon Redshift as a target database

1. Create a JSON file with the following IAM policy. Name the JSON file dmsAssumeRolePolicyDocument3.json.

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "1",  
            "Effect": "Allow",  
            "Principal": {  
                "Service": "dms.amazonaws.com"  
            },  
            "Action": "s3:GetBucketLocation"  
        }  
    ]  
}
```

```
        "Action": "sts:AssumeRole"
    },
{
    "Sid": "2",
    "Effect": "Allow",
    "Principal": {
        "Service": "redshift.amazonaws.com"
    },
    "Action": "sts:AssumeRole"
}
]
```

2. Create the role using the AWS CLI using the following command.

```
aws iam create-role --role-name dms-access-for-endpoint --assume-role-policy-document file://dmsAssumeRolePolicyDocument3.json
```

3. Attach the AmazonDMSRedshiftS3Role policy to dms-access-for-endpoint role using the following command.

```
aws iam attach-role-policy --role-name dms-access-for-endpoint \
--policy-arn arn:aws:iam::aws:policy/service-role/AmazonDMSRedshiftS3Role
```

You should now have the IAM policies in place to use the AWS CLI or AWS DMS API.

# Fine-grained access control using resource names and tags

You can use resource names and resource tags based on Amazon Resource Names (ARNs) to manage access to AWS DMS resources. You do this by defining permitted action or including conditional statements in IAM policies.

## Using resource names to control access

You can create an IAM user account and assign a policy based on the AWS DMS resource's ARN.

The following policy denies access to the AWS DMS replication instance with the ARN *arn:aws:dms:us-east-1:152683116:rep:DOH67ZTOXGLIXMIHKITV*:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Action": [  
                "dms:*"  
            ],  
            "Effect": "Deny",  
            "Resource": "arn:aws:dms:us-east-1:152683116:rep:DOH67ZTOXGLIXMIHKITV"  
        }  
    ]  
}
```

For example, the following commands fail when the policy is in effect.

```
$ aws dms delete-replication-instance  
  --replication-instance-arn "arn:aws:dms:us-east-1:152683116:rep:DOH67ZTOXGLIXMIHKITV"  
  
A client error (AccessDeniedException) occurred when calling the DeleteReplicationInstance  
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:  
dms:DeleteReplicationInstance on resource: arn:aws:dms:us-  
east-1:152683116:rep:DOH67ZTOXGLIXMIHKITV  
  
$ aws dms modify-replication-instance  
  --replication-instance-arn "arn:aws:dms:us-east-1:152683116:rep:DOH67ZTOXGLIXMIHKITV"  
  
A client error (AccessDeniedException) occurred when calling the ModifyReplicationInstance  
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:  
dms:ModifyReplicationInstance on resource: arn:aws:dms:us-  
east-1:152683116:rep:DOH67ZTOXGLIXMIHKITV
```

You can also specify IAM policies that limit access to AWS DMS endpoints and replication tasks.

The following policy limits access to an AWS DMS endpoint using the endpoint's ARN.

```
{  
    "Version": "2012-10-17",
```

```

"Statement": [
    {
        "Action": [
            "dms:*"
        ],
        "Effect": "Deny",
        "Resource": "arn:aws:dms:us-east-1:152683116:endpoint:D6E37YBXTNHOA6XRQSZCUGX"
    }
]
}

```

For example, the following commands fail when the policy using the endpoint's ARN is in effect.

```

$ aws dms delete-endpoint
--endpoint-arn "arn:aws:dms:us-east-1:152683116:endpoint:D6E37YBXTNHOA6XRQSZCUGX"

A client error (AccessDeniedException) occurred when calling the DeleteEndpoint operation:
User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:DeleteEndpoint
on resource: arn:aws:dms:us-east-1:152683116:endpoint:D6E37YBXTNHOA6XRQSZCUGX

$ aws dms modify-endpoint
--endpoint-arn "arn:aws:dms:us-east-1:152683116:endpoint:D6E37YBXTNHOA6XRQSZCUGX"

A client error (AccessDeniedException) occurred when calling the ModifyEndpoint operation:
User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:ModifyEndpoint
on resource: arn:aws:dms:us-east-1:152683116:endpoint:D6E37YBXTNHOA6XRQSZCUGX

```

The following policy limits access to an AWS DMS task using the task's ARN.

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "dms:*"
            ],
            "Effect": "Deny",
            "Resource": "arn:aws:dms:us-east-1:152683116:task:U03YR4N47DXH3ATT4YMWOIT"
        }
    ]
}

```

For example, the following commands fail when the policy using the task's ARN is in effect.

```

$ aws dms delete-replication-task
--replication-task-arn "arn:aws:dms:us-east-1:152683116:task:U03YR4N47DXH3ATT4YMWOIT"

A client error (AccessDeniedException) occurred when calling the DeleteReplicationTask
operation:
User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:DeleteReplicationTask
on resource: arn:aws:dms:us-east-1:152683116:task:U03YR4N47DXH3ATT4YMWOIT

```

## Using tags to control access

AWS DMS defines a set of common key-value pairs that are available for use in customer defined policies without any additional tagging requirements. For more information about tagging AWS DMS resources, see [Tagging resources in AWS Database Migration Service \(p. 591\)](#).

The following lists the standard tags available for use with AWS DMS:

- `aws:CurrentTime` – Represents the request date and time, allowing the restriction of access based on temporal criteria.
- `aws:EpochTime` – This tag is similar to the `aws:CurrentTime` tag preceding, except that the current time is represented as the number of seconds elapsed since the Unix epoch.
- `aws:MultiFactorAuthPresent` – This is a Boolean tag that indicates whether or not the request was signed via multi-factor authentication.
- `aws:MultiFactorAuthAge` – Provides access to the age of the multi-factor authentication token (in seconds).
- `aws:principaltype` – Provides access to the type of principal (user, account, federated user, etc.) for the current request.
- `aws:SourceIp` – Represents the source ip address for the user issuing the request.
- `aws:UserAgent` – Provides information about the client application requesting a resource.
- `aws:userid` – Provides access to the ID of the user issuing the request.
- `aws:username` – Provides access to the name of the user issuing the request.
- `dms:InstanceClass` – Provides access to the compute size of the replication instance host(s).
- `dms:StorageSize` – Provides access to the storage volume size (in GB).

You can also define your own tags. Customer-defined tags are simple key-value pairs that are persisted in the AWS tagging service. You can add these to AWS DMS resources, including replication instances, endpoints, and tasks. These tags are matched by using IAM "Conditional" statements in policies, and are referenced using a specific conditional tag. The tag keys are prefixed with "dms", the resource type, and the "tag" prefix. The following shows the tag format.

```
dms:{resource type}-tag/{tag key}={tag value}
```

For example, suppose that you want to define a policy that only allows an API call to succeed for a replication instance that contains the tag "stage=production". The following conditional statement matches a resource with the given tag.

```
"Condition":  
{  
    "streq":  
        {  
            "dms:rep-tag/stage":"production"  
        }  
}
```

You add the following tag to a replication instance that matches this policy condition.

```
stage production
```

In addition to tags already assigned to AWS DMS resources, policies can also be written to limit the tag keys and values that can be applied to a given resource. In this case, the tag prefix is "req".

For example, the following policy statement limits the tags that a user can assign to a given resource to a specific list of allowed values.

```

"Condition":
{
    "streq":
        {
            "dms:rep-tag/stage": [ "production", "development", "testing" ]
        }
}

```

The following policy examples limit access to an AWS DMS resource based on resource tags.

The following policy limits access to a replication instance where the tag value is "Desktop" and the tag key is "Env":

```

{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "dms:/*"
            ],
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "dms:rep-tag/Env": [
                        "Desktop"
                    ]
                }
            }
        }
    ]
}

```

The following commands succeed or fail based on the IAM policy that restricts access when the tag value is "Desktop" and the tag key is "Env".

```

$ aws dms list-tags-for-resource
--resource-name arn:aws:dms:us-east-1:152683116:rep:46DHOU7J0JY0JXWD0ZNFEN
--endpoint-url http://localhost:8000
{
    "TagList": [
        {
            "Value": "Desktop",
            "Key": "Env"
        }
    ]
}

$ aws dms delete-replication-instance
--replication-instance-arn "arn:aws:dms:us-east-1:152683116:rep:46DHOU7J0JY0JXWD0ZNFEN"
A client error (AccessDeniedException) occurred when calling the DeleteReplicationInstance operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform: dms>DeleteReplicationInstance on resource: arn:aws:dms:us-east-1:152683116:rep:46DHOU7J0JY0JXWD0ZNFEN

$ aws dms modify-replication-instance
--replication-instance-arn "arn:aws:dms:us-east-1:152683116:rep:46DHOU7J0JY0JXWD0ZNFEN"

A client error (AccessDeniedException) occurred when calling the ModifyReplicationInstance

```

```
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:  
dms:ModifyReplicationInstance on resource: arn:aws:dms:us-  
east-1:152683116:rep:46DHOU7J0JY0JXWD0ZNFEN
```

```
$ aws dms add-tags-to-resource  
--resource-name arn:aws:dms:us-east-1:152683116:rep:46DHOU7J0JY0JXWD0ZNFEN  
--tags Key=CostCenter,Value=1234
```

A client error (AccessDeniedException) occurred when calling the AddTagsToResource operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform: dms:AddTagsToResource on resource: arn:aws:dms:us-east-1:152683116:rep:46DHOU7J0JY0JXWD0ZNFEN

```
$ aws dms remove-tags-from-resource  
--resource-name arn:aws:dms:us-east-1:152683116:rep:46DHOU7J0JY0JXWD0ZNFEN  
--tag-keys Env
```

A client error (AccessDeniedException) occurred when calling the RemoveTagsFromResource operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform: dms:RemoveTagsFromResource on resource: arn:aws:dms:us-east-1:152683116:rep:46DHOU7J0JY0JXWD0ZNFEN

The following policy limits access to an AWS DMS endpoint where the tag value is "Desktop" and the tag key is "Env".

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "dms:*
            ],
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "dms:endpoint-tag/Env": [
                        "Desktop"
                    ]
                }
            }
        }
    ]
}
```

The following commands succeed or fail based on the IAM policy that restricts access when the tag value is "Desktop" and the tag key is "Env".

```
$ aws dms list-tags-for-resource  
--resource-name arn:aws:dms:us-east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I
{
    "TagList": [
        {
            "Value": "Desktop",
            "Key": "Env"
        }
    ]
}
```

```
$ aws dms delete-endpoint
  --endpoint-arn "arn:aws:dms:us-east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I"

A client error (AccessDeniedException) occurred when calling the DeleteEndpoint
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:DeleteEndpoint on resource: arn:aws:dms:us-
east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I

$ aws dms modify-endpoint
  --endpoint-arn "arn:aws:dms:us-east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I"

A client error (AccessDeniedException) occurred when calling the ModifyEndpoint
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:ModifyEndpoint on resource: arn:aws:dms:us-
east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I

$ aws dms add-tags-to-resource
  --resource-name arn:aws:dms:us-east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I
  --tags Key=CostCenter,Value=1234

A client error (AccessDeniedException) occurred when calling the AddTagsToResource
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:AddTagsToResource on resource: arn:aws:dms:us-
east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I

$ aws dms remove-tags-from-resource
  --resource-name arn:aws:dms:us-east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I
  --tag-keys Env

A client error (AccessDeniedException) occurred when calling the RemoveTagsFromResource
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:RemoveTagsFromResource on resource: arn:aws:dms:us-
east-1:152683116:endpoint:J2YCZPNGOLFY52344IZWA6I
```

The following policy limits access to a replication task where the tag value is "Desktop" and the tag key is "Env".

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "dms:*"
            ],
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "dms:task-tag/Env": [
                        "Desktop"
                    ]
                }
            }
        ]
    }
}
```

The following commands succeed or fail based on the IAM policy that restricts access when the tag value is "Desktop" and the tag key is "Env".

```
$ aws dms list-tags-for-resource
--resource-name arn:aws:dms:us-east-1:152683116:task:RB7N24J2XBUPS3RFABZTG3
{
    "TagList": [
        {
            "Value": "Desktop",
            "Key": "Env"
        }
    ]
}

$ aws dms delete-replication-task
--replication-task-arn "arn:aws:dms:us-east-1:152683116:task:RB7N24J2XBUPS3RFABZTG3"

A client error (AccessDeniedException) occurred when calling the DeleteReplicationTask
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:DeleteReplicationTask on resource: arn:aws:dms:us-
east-1:152683116:task:RB7N24J2XBUPS3RFABZTG3

$ aws dms add-tags-to-resource
--resource-name arn:aws:dms:us-east-1:152683116:task:RB7N24J2XBUPS3RFABZTG3
--tags Key=CostCenter,Value=1234

A client error (AccessDeniedException) occurred when calling the AddTagsToResource
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:AddTagsToResource on resource: arn:aws:dms:us-
east-1:152683116:task:RB7N24J2XBUPS3RFABZTG3

$ aws dms remove-tags-from-resource
--resource-name arn:aws:dms:us-east-1:152683116:task:RB7N24J2XBUPS3RFABZTG3
--tag-keys Env

A client error (AccessDeniedException) occurred when calling the RemoveTagsFromResource
operation: User: arn:aws:iam::152683116:user/dmstestusr is not authorized to perform:
dms:RemoveTagsFromResource on resource: arn:aws:dms:us-
east-1:152683116:task:RB7N24J2XBUPS3RFABZTG3
```

# Setting an encryption key and specifying AWS KMS permissions

AWS DMS encrypts the storage used by a replication instance and the endpoint connection information. To encrypt the storage used by a replication instance, AWS DMS uses an AWS Key Management Service (AWS KMS) key that is unique to your AWS account. You can view and manage this key with AWS KMS. You can use the default KMS key in your account (aws/dms) or you can create a custom KMS key. If you have an existing KMS key, you can also use that key for encryption.

## Note

Any custom or existing AWS KMS key that you use as an encryption key must be a symmetric key. AWS DMS does not support the use of asymmetric encryption keys. For more information on symmetric and asymmetric encryption keys, see <https://docs.aws.amazon.com/kms/latest/developerguide/symmetric-asymmetric.html> in the *AWS Key Management Service Developer Guide*.

The default KMS key (aws/dms) is created when you first launch a replication instance, if you haven't selected a custom KMS key from the **Advanced** section of the **Create Replication Instance** page. If you use the default KMS key, the only permissions you need to grant to the IAM user account you are using for migration are kms:ListAliases and kms:DescribeKey. For more information about using the default KMS key, see [IAM permissions needed to use AWS DMS \(p. 628\)](#).

To use a custom KMS key, assign permissions for the custom KMS key using one of the following options:

- Add the IAM user account used for the migration as a key administrator or key user for the AWS KMS custom key. Doing this ensures that necessary AWS KMS permissions are granted to the IAM user account. This action is in addition to the IAM permissions that you grant to the IAM user account to use AWS DMS. For more information about granting permissions to a key user, see [Allows key users to use the KMS key](#) in the *AWS Key Management Service Developer Guide*.
- If you don't want to add the IAM user account as a key administrator or key user for your custom KMS key, then add the following additional permissions to the IAM permissions that you must grant to the IAM user account to use AWS DMS.

```
{  
    "Effect": "Allow",  
    "Action": [  
        "kms>ListAliases",  
        "kms>DescribeKey",  
        "kms>CreateGrant",  
        "kms>Encrypt",  
        "kms>ReEncrypt"  
    ],  
    "Resource": "*"  
},
```

AWS DMS also works with KMS key aliases. For more information about creating your own AWS KMS keys and giving users access to a KMS key, see the [AWS KMS Developer Guide](#).

If you don't specify a KMS key identifier, then AWS DMS uses your default encryption key. AWS KMS creates the default encryption key for AWS DMS for your AWS account. Your AWS account has a different default encryption key for each AWS Region.

To manage the AWS KMS keys used for encrypting your AWS DMS resources, use the AWS Key Management Service. AWS KMS combines secure, highly available hardware and software to provide

a key management system scaled for the cloud. Using AWS KMS, you can create encryption keys and define the policies that control how these keys can be used.

### You can find AWS KMS in the AWS Management Console

1. Sign in to the AWS Management Console and open the AWS Key Management Service (AWS KMS) console at <https://console.aws.amazon.com/kms>.
2. To change the AWS Region, use the Region selector in the upper-right corner of the page.
3. Choose one of the following options to work with AWS KMS keys:
  - To view the keys in your account that AWS creates and manages for you, in the navigation pane, choose **AWS managed keys**.
  - To view the keys in your account that you create and manage, in the navigation pane choose **Customer managed keys**.

AWS KMS supports AWS CloudTrail, so you can audit key usage to verify that keys are being used appropriately. Your AWS KMS keys can be used in combination with AWS DMS and supported AWS services such as Amazon RDS, Amazon S3, Amazon Redshift, and Amazon EBS.

You can also create custom AWS KMS keys specifically to encrypt target data for the following AWS DMS endpoints:

- Amazon Redshift – For more information, see [Creating and using AWS KMS keys to encrypt Amazon Redshift target data \(p. 319\)](#).
- Amazon S3 – For more information, see [Creating AWS KMS keys to encrypt Amazon S3 target objects \(p. 341\)](#).

After you have created your AWS DMS resources with a KMS key, you can't change the encryption key for those resources. Make sure to determine your encryption key requirements before you create your AWS DMS resources.

# Network security for AWS Database Migration Service

The security requirements for the network you create when using AWS Database Migration Service depend on how you configure the network. The general rules for network security for AWS DMS are as follows:

- The replication instance must have access to the source and target endpoints. The security group for the replication instance must have network ACLs or rules that allow egress from the instance out on the database port to the database endpoints.
- Database endpoints must include network ACLs and security group rules that allow incoming access from the replication instance. You can achieve this using the replication instance's security group, the private IP address, the public IP address, or the NAT gateway's public address, depending on your configuration.
- If your network uses a VPN tunnel, the Amazon EC2 instance acting as the NAT gateway must use a security group that has rules that allow the replication instance to send traffic through it.

By default, the VPC security group used by the AWS DMS replication instance has rules that allow egress to 0.0.0.0/0 on all ports. If you modify this security group or use your own security group, egress must, at a minimum, be permitted to the source and target endpoints on the respective database ports.

The network configurations that you can use for database migration each require specific security considerations:

- [Configuration with all database migration components in one VPC \(p. 112\)](#) – The security group used by the endpoints must allow ingress on the database port from the replication instance. Ensure that the security group used by the replication instance has ingress to the endpoints, or you can create a rule in the security group used by the endpoints that allows the private IP address of the replication instance access.
- [Configuration with multiple VPCs \(p. 113\)](#) – The security group used by the replication instance must have a rule for the VPC range and the DB port on the database.
- [Configuration for a network to a VPC using AWS Direct Connect or a VPN \(p. 113\)](#) – a VPN tunnel allowing traffic to tunnel from the VPC into an on-premises VPN. In this configuration, the VPC includes a routing rule that sends traffic destined for a specific IP address or range to a host that can bridge traffic from the VPC into the on-premises VPN. If this case, the NAT host includes its own Security Group settings that must allow traffic from the Replication Instance's private IP address or security group into the NAT instance.
- [Configuration for a network to a VPC using the internet \(p. 114\)](#) – The VPC security group must include routing rules that send traffic not destined for the VPC to the Internet gateway. In this configuration, the connection to the endpoint appears to come from the public IP address on the replication instance.
- [Configuration with an RDS DB instance not in a VPC to a DB instance in a VPC using ClassicLink \(p. 114\)](#)
  - When the source or target Amazon RDS DB instance is not in a VPC and does not share a security group with the VPC where the replication instance is located, you can setup a proxy server and use ClassicLink to connect the source and target databases.
- **Source endpoint is outside the VPC used by the replication instance and uses a NAT gateway** – You can configure a network address translation (NAT) gateway using a single Elastic IP address bound to a single Elastic network interface. This Elastic network interface then receives a NAT identifier (nat-#####). If the VPC includes a default route to that NAT gateway instead of the internet gateway, the replication instance instead appears to contact the database endpoint using the public IP address of the internet gateway. In this case, the ingress to the database endpoint outside the VPC needs to allow ingress from the NAT address instead of the replication instance's public IP address.

- **VPC endpoints for non-RDBMS engines** – AWS DMS doesn't support VPC endpoints for non-RDBMS engines.

# Using SSL with AWS Database Migration Service

You can encrypt connections for source and target endpoints by using Secure Sockets Layer (SSL). To do so, you can use the AWS DMS Management Console or AWS DMS API to assign a certificate to an endpoint. You can also use the AWS DMS console to manage your certificates.

Not all databases use SSL in the same way. Amazon Aurora MySQL-Compatible Edition uses the server name, the endpoint of the primary instance in the cluster, as the endpoint for SSL. An Amazon Redshift endpoint already uses an SSL connection and does not require an SSL connection set up by AWS DMS. An Oracle endpoint requires additional steps; for more information, see [SSL support for an Oracle endpoint \(p. 154\)](#).

## Topics

- [Limitations on using SSL with AWS DMS \(p. 648\)](#)
- [Managing certificates \(p. 648\)](#)
- [Enabling SSL for a MySQL-compatible, PostgreSQL, or SQL Server endpoint \(p. 649\)](#)

To assign a certificate to an endpoint, you provide the root certificate or the chain of intermediate CA certificates leading up to the root (as a certificate bundle), that was used to sign the server SSL certificate that is deployed on your endpoint. Certificates are accepted as PEM formatted X509 files, only. When you import a certificate, you receive an Amazon Resource Name (ARN) that you can use to specify that certificate for an endpoint. If you use Amazon RDS, you can download the root CA and certificate bundle provided in the `rds-combined-ca-bundle.pem` file hosted by Amazon RDS. For more information about downloading this file, see [Using SSL/TLS to encrypt a connection to a DB instance](#) in the *Amazon RDS User Guide*.

You can choose from several SSL modes to use for your SSL certificate verification.

- **none** – The connection is not encrypted. This option is not secure, but requires less overhead.
- **require** – The connection is encrypted using SSL (TLS) but no CA verification is made. This option is more secure, and requires more overhead.
- **verify-ca** – The connection is encrypted. This option is more secure, and requires more overhead. This option verifies the server certificate.
- **verify-full** – The connection is encrypted. This option is more secure, and requires more overhead. This option verifies the server certificate and verifies that the server hostname matches the hostname attribute for the certificate.

Not all SSL modes work with all database endpoints. The following table shows which SSL modes are supported for each database engine.

DB engine	none	require	verify-ca	verify-full
MySQL/MariaDB/ Amazon Aurora MySQL	Default	Not supported	Supported	Supported
Microsoft SQL Server	Default	Supported	Not Supported	Supported
PostgreSQL	Default	Supported	Supported	Supported
Amazon Redshift	Default	SSL not enabled	SSL not enabled	SSL not enabled
Oracle	Default	Not supported	Supported	Not Supported
SAP ASE	Default	SSL not enabled	SSL not enabled	Supported

DB engine	none	require	verify-ca	verify-full
MongoDB	Default	Supported	Not Supported	Supported
Db2 LUW	Default	Not Supported	Supported	Not Supported
Db2 for z/OS	Default	Not Supported	Supported	Not Supported

#### Note

The SSL Mode option on the DMS console or API doesn't apply to some data streaming and NoSQL services like Kinesis, and DynamoDB. They are secure by default, so DMS shows the SSL mode setting is equal to none (**SSL Mode=None**). You don't need to provide any additional configuration for your endpoint to make use of SSL. For example, when using Kinesis as a target endpoint, it is secure by default. All API calls to Kinesis use SSL, so there is no need for an additional SSL option in the DMS endpoint. You can securely put data and retrieve data through SSL endpoints using the HTTPS protocol, which DMS uses by default when connecting to a Kinesis Data Stream.

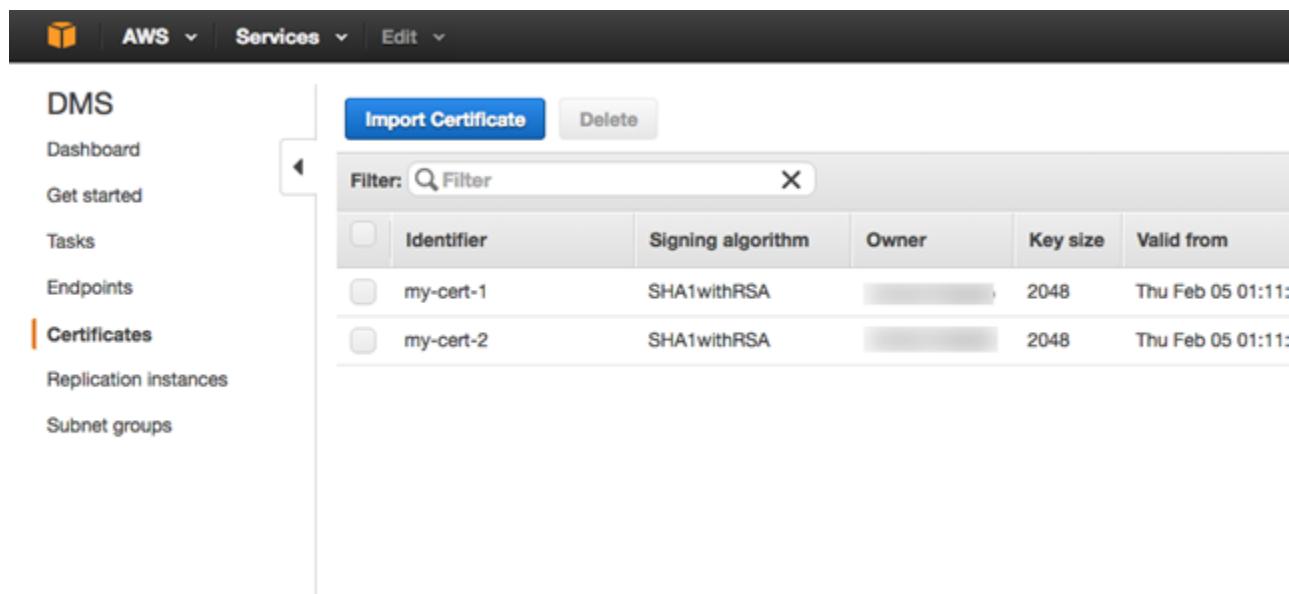
## Limitations on using SSL with AWS DMS

Following are limitations on using SSL with AWS DMS:

- SSL connections to Amazon Redshift target endpoints aren't supported. AWS DMS uses an Amazon S3 bucket to transfer data to the Amazon Redshift database. This transmission is encrypted by Amazon Redshift by default.
- SQL timeouts can occur when performing change data capture (CDC) tasks with SSL-enabled Oracle endpoints. If you have an issue where CDC counters don't reflect the expected numbers, set the `MinimumTransactionSize` parameter from the `ChangeProcessingTuning` section of the task settings to a lower value. You can start with a value as low as 100. For more information about the `MinimumTransactionSize` parameter, see [Change processing tuning settings \(p. 465\)](#).
- You can import certificates only in the .pem and .sso (Oracle wallet) formats.
- In some cases, your server SSL certificate might be signed by an intermediate certificate authority (CA). If so, make sure that the entire certificate chain leading from the intermediate CA up to the root CA is imported as a single .pem file.
- If you are using self-signed certificates on your server, choose **require** as your SSL mode. The **require** SSL mode implicitly trusts the server's SSL certificate and doesn't try to validate that the certificate was signed by a CA.

## Managing certificates

You can use the DMS console to view and manage your SSL certificates. You can also import your certificates using the DMS console.



## Enabling SSL for a MySQL-compatible, PostgreSQL, or SQL Server endpoint

You can add an SSL connection to a newly created endpoint or to an existing endpoint.

### To create an AWS DMS endpoint with SSL

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.

If you're signed in as an AWS Identity and Access Management (IAM) user, make sure that you have the appropriate permissions to access AWS DMS. For more information about the permissions required for database migration, see [IAM permissions needed to use AWS DMS \(p. 628\)](#).

2. In the navigation pane, choose **Certificates**.
3. Choose **Import Certificate**.
4. Upload the certificate you want to use for encrypting the connection to an endpoint.

#### Note

You can also upload a certificate using the AWS DMS console when you create or modify an endpoint by selecting **Add new CA certificate** on the **Create database endpoint** page. For Aurora Serverless as target, get the certificate mentioned in [Using TLS/SSL with Aurora Serverless](#).

5. Create an endpoint as described in [Step 2: Specify source and target endpoints \(p. 26\)](#)

### To modify an existing AWS DMS endpoint to use SSL

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.

If you're signed in as an IAM user, make sure that you have the appropriate permissions to access AWS DMS. For more information about the permissions required for database migration, see [IAM permissions needed to use AWS DMS \(p. 628\)](#).

2. In the navigation pane, choose **Certificates**.

3. Choose **Import Certificate**.
4. Upload the certificate you want to use for encrypting the connection to an endpoint.

**Note**

You can also upload a certificate using the AWS DMS console when you create or modify an endpoint by selecting **Add new CA certificate** on the **Create database endpoint** page.

5. In the navigation pane, choose **Endpoints**, select the endpoint you want to modify, and choose **Modify**.
6. Choose a value for **SSL mode**.

If you choose **verify-ca** or **verify-full** mode, specify the certificate that you want to use for **CA certificate**, as shown following.

### Create database endpoint

A database endpoint is used by the replication server to connect to a database. The database specified in the endpoint can be on-prem. Details should be specified in the form below. It is recommended that you test your endpoint connections here to avoid errors during pr

The screenshot shows the 'Create database endpoint' configuration page. The 'Endpoint type\*' field is set to 'Source'. The 'Endpoint identifier\*' field contains 'e.g. ProdEndpoint'. The 'Source engine\*' dropdown is set to 'sqlserver'. The 'Server name\*' field is empty. The 'Port\*' field is empty. The 'SSL mode\*' dropdown is set to 'verify-ca'. The 'CA certificate\*' dropdown is set to '- Select One -' and has a link 'Add new CA certificate'. The 'User name\*' and 'Password\*' fields are empty. Red circles highlight the 'SSL mode\*' and 'CA certificate\*' fields.

» Advanced

7. Choose **Modify**.
8. When the endpoint has been modified, choose the endpoint and choose **Test connection** to determine if the SSL connection is working.

After you create your source and target endpoints, create a task that uses these endpoints. For more information about creating a task, see [Step 3: Create a task and migrate data \(p. 28\)](#).

# Changing the database password

In most situations, changing the database password for your source or target endpoint is straightforward. If you need to change the database password for an endpoint that you are currently using in a migration or replication task, the process is slightly more complex. The procedure following shows how to do this.

## To change the database password for an endpoint in a migration or replication task

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.

If you're signed in as an IAM user, make sure that you have the appropriate permissions to access AWS DMS. For more information about the permissions required, see [IAM permissions needed to use AWS DMS \(p. 628\)](#).
2. In the navigation pane, choose **Tasks**.
3. Choose the task that uses the endpoint you want to change the database password for, and then choose **Stop**.
4. While the task is stopped, you can change the password of the database for the endpoint using the native tools you use to work with the database.
5. Return to the DMS Management Console and choose **Endpoints** from the navigation pane.
6. Choose the endpoint for the database you changed the password for, and then choose **Modify**.
7. Type the new password in the **Password** box, and then choose **Modify**.
8. Choose **Tasks** from the navigation pane.
9. Choose the task that you stopped previously, and choose **Start/Resume**.
10. Choose either **Start** or **Resume**, depending on how you want to continue the task, and then choose **Start task**.

# Quotas for AWS Database Migration Service

Following, you can find the resource quotas and naming constraints for AWS Database Migration Service (AWS DMS).

The maximum size of a database that AWS DMS can migrate depends on a number of factors. These include your source environment, the distribution of data in your source database, and how busy your source system is.

The best way to determine whether your particular system is a candidate for AWS DMS is to test it. Start slowly so you can get the configuration worked out, then add some complex objects. Finally, attempt a full load as a test.

## Resource quotas for AWS Database Migration Service

Each AWS account has quotas for each AWS Region on the number of AWS DMS resources that can be created. After a quota for a resource has been reached, additional calls to create that resource fail with an exception.

The following table lists the AWS DMS resources and their quotas for each AWS Region.

Resource	Default quota
API request throttling	200 request maximum per second
API request refresh rate	8 requests per second
Replication instances per user account	60
Total amount of storage for a replication instance	30,000 GB
Event subscriptions per user account	60
Replication subnet groups per user account	60
Subnets per replication subnet group	60
Endpoints per user account	1000
Endpoints per replication instance	100
Tasks per user account	600
Tasks per replication instance	200
Certificates per user account	100

For more information on the API request throttling quota and refresh rate, see [Understanding API request throttling \(p. 653\)](#).

The 30,000-GB quota for storage applies to all your AWS DMS replication instances in a given AWS Region. This storage is used to cache changes if a target can't keep up with a source, and for storing log information.

## Understanding API request throttling

AWS DMS supports a varying, but maximum API request quota of 100 API calls per second. In other words, your API requests are throttled when they exceed this rate. Also, you can be limited to fewer API calls per second, depending on how long it takes AWS DMS to refresh your quota before you make another API request. This quota applies both when you make API calls directly and when they are made on your behalf as part of using the AWS DMS Management Console.

To understand how API request throttling works, it helps to imagine that AWS DMS maintains a token bucket that tracks your API requests. In this scenario, each token in the bucket allows you to make a single API call. You can have no more than 100 tokens in the bucket at any one time. When you make an API call, AWS DMS removes one token from the bucket. If you make 100 API calls in under a second, your bucket is empty and any attempt to make another API call fails. For each second that you don't make an API call, AWS DMS adds 4 tokens to the bucket, up to the 100 token maximum. This is the AWS DMS API request refresh rate. At any point after throttling, when you have tokens added to your bucket, you can make as many additional API calls as tokens available until your calls are throttled again.

If you are using the AWS CLI to run API calls that are throttled, AWS DMS returns an error like the following:

An error occurred (ThrottlingException) when calling the `AwsDmsApiCall` operation (reached max retries: 2): Rate exceeded

Here, `AwsDmsApiCall` is the name of the AWS DMS API operation that was throttled, for example, `DescribeTableStatistics`. You can then retry or make a different call after sufficient delay to avoid throttling.

### Note

Unlike API request throttling managed by some other services, such as Amazon EC2, you can't order an increase in the API request throttling quotas managed by AWS DMS.

# Troubleshooting migration tasks in AWS Database Migration Service

Following, you can find topics about troubleshooting issues with AWS Database Migration Service (AWS DMS). These topics can help you to resolve common issues using both AWS DMS and selected endpoint databases.

If you have opened an AWS Support case, your support engineer might identify a potential issue with one of your endpoint database configurations. Your engineer might also ask you to run a support script to return diagnostic information about your database. For details about downloading, running, and uploading the diagnostic information from this type of support script, see [Working with diagnostic support scripts in AWS DMS \(p. 674\)](#).

## Topics

- [Migration tasks run slowly \(p. 655\)](#)
- [Task status bar doesn't move \(p. 655\)](#)
- [Task completes but nothing was migrated \(p. 655\)](#)
- [Foreign keys and secondary indexes are missing \(p. 655\)](#)
- [AWS DMS does not create CloudWatch logs \(p. 656\)](#)
- [Issues occur with connecting to Amazon RDS \(p. 656\)](#)
- [Networking issues occur \(p. 656\)](#)
- [CDC is stuck after full load \(p. 657\)](#)
- [Primary key violation errors occur when you restart a task \(p. 657\)](#)
- [Initial load of a schema fails \(p. 657\)](#)
- [Tasks fail with an unknown error \(p. 658\)](#)
- [Task restart loads tables from the beginning \(p. 658\)](#)
- [Number of tables per task causes issues \(p. 658\)](#)
- [Tasks fail when a primary key is created on a LOB column \(p. 658\)](#)
- [Duplicate records occur on a target table without a primary key \(p. 658\)](#)
- [Source endpoints fall in the reserved IP range \(p. 658\)](#)
- [Timestamps are garbled in Amazon Athena queries \(p. 659\)](#)
- [Troubleshooting issues with Oracle \(p. 659\)](#)
- [Troubleshooting issues with MySQL \(p. 664\)](#)
- [Troubleshooting issues with PostgreSQL \(p. 668\)](#)
- [Troubleshooting issues with Microsoft SQL Server \(p. 670\)](#)
- [Troubleshooting issues with Amazon Redshift \(p. 672\)](#)
- [Troubleshooting issues with Amazon Aurora MySQL \(p. 673\)](#)
- [Troubleshooting issues with SAP ASE \(p. 674\)](#)
- [Troubleshooting issues with IBM Db2 \(p. 674\)](#)
- [Working with diagnostic support scripts in AWS DMS \(p. 674\)](#)
- [Working with the AWS DMS diagnostic support AMI \(p. 683\)](#)

## Migration tasks run slowly

Several issues can cause a migration task to run slowly, or cause subsequent tasks to run slower than the initial task.

The most common reason for a migration task running slowly is that there are inadequate resources allocated to the AWS DMS replication instance. To make sure that your instance has enough resources for the tasks you are running on it, check your replication instance's use of CPU, memory, swap files, and IOPS. For example, multiple tasks with Amazon Redshift as an endpoint are I/O intensive. You can increase IOPS for your replication instance or split your tasks across multiple replication instances for a more efficient migration.

For more information about determining the size of your replication instance, see [Choosing the best size for a replication instance \(p. 107\)](#).

You can increase the speed of an initial migration load by doing the following:

- If your target is an Amazon RDS DB instance, make sure that Multi-AZ isn't enabled for the target DB instance.
- Turn off any automatic backups or logging on the target database during the load, and turn back on those features after your migration is complete.
- If the feature is available on your target, use provisioned IOPS.
- If your migration data contains LOBs, make sure that the task is optimized for LOB migration. For more information on optimizing for LOBs, see [Target metadata task settings \(p. 452\)](#).

## Task status bar doesn't move

The task status bar gives an estimation of the task's progress. The quality of this estimate depends on the quality of the source database's table statistics; the better the table statistics, the more accurate the estimation.

For a task with only one table that has no estimated rows statistic, AWS DMS can't provide any kind of percentage complete estimate. In this case, use the task state and the indication of rows loaded to confirm that the task is running and making progress.

## Task completes but nothing was migrated

Do the following if nothing was migrated after your task has completed.

- Check if the user that created the endpoint has read access to the table you intend to migrate.
- Check if the object you want to migrate is a table. If it is a view, update table mappings and specify the object-locator as "view" or "all". For more information, see [Specifying table selection and transformations rules from the console \(p. 491\)](#).

## Foreign keys and secondary indexes are missing

AWS DMS creates tables, primary keys, and in some cases unique indexes, but it doesn't create any other objects that aren't required to efficiently migrate the data from the source. For example, it doesn't create secondary indexes, non-primary key constraints, or data defaults.

To migrate secondary objects from your database, use the database's native tools if you are migrating to the same database engine as your source database. Use the AWS Schema Conversion Tool (AWS SCT)

if you are migrating to a different database engine than that used by your source database to migrate secondary objects.

## AWS DMS does not create CloudWatch logs

If your replication task doesn't create CloudWatch logs, make sure that your account has the `dms-cloudwatch-logs-role` role. If this role is not present, do the following to create it:

1. Sign in to the AWS Management Console and open the IAM console at <https://console.aws.amazon.com/iam/>.
2. Choose the **Roles** tab. Choose **Create role**.
3. In the **Select type of trusted entity** section, choose **AWS service**.
4. In the **Choose a use case** section, choose **DMS**.
5. Choose **Next: Permissions**.
6. Enter **AmazonDMSCloudWatchLogsRole** in the search field, and check the box next to **AmazonDMSCloudWatchLogsRole**. This grants AWS DMS permissions to access CloudWatch.
7. Choose **Next: Tags**.
8. Choose **Next: Review**.
9. Enter **dms-cloudwatch-logs-role** for **Role name**. This name is case sensitive.
10. Choose **Create role**.

## Issues occur with connecting to Amazon RDS

There can be several reasons why you can't connect to an Amazon RDS DB instance that you set as a source or target. Some items to check follow:

- Check that the user name and password combination is correct.
- Check that the endpoint value shown in the Amazon RDS console for the instance is the same as the endpoint identifier you used to create the AWS DMS endpoint.
- Check that the port value shown in the Amazon RDS console for the instance is the same as the port assigned to the AWS DMS endpoint.
- Check that the security group assigned to the Amazon RDS DB instance allows connections from the AWS DMS replication instance.
- If the AWS DMS replication instance and the Amazon RDS DB instance aren't in the same virtual private cloud (VPC), check that the DB instance is publicly accessible.

## Error message: Incorrect thread connection string: Incorrect thread value 0

This error can often occur when you are testing the connection to an endpoint. This error indicates that there is an error in the connection string. An example is a space after the host IP address. Another is a bad character copied into the connection string.

## Networking issues occur

The most common networking issue involves the VPC security group used by the AWS DMS replication instance. By default, this security group has rules that allow egress to 0.0.0.0/0 on all ports. In many

cases, you modify this security group or use your own security group. If so, at a minimum, make sure to give egress to the source and target endpoints on their respective database ports.

Other configuration-related issues can include the following:

- **Replication instance and both source and target endpoints in the same VPC** – The security group used by the endpoints must allow ingress on the database port from the replication instance. Make sure that the security group used by the replication instance has ingress to the endpoints. Or you can create a rule in the security group used by the endpoints that allows the private IP address of the replication instance access.
- **Source endpoint is outside the VPC used by the replication instance (using an internet gateway)**
  - The VPC security group must include routing rules that send traffic that isn't for the VPC to the internet gateway. In this configuration, the connection to the endpoint appears to come from the public IP address on the replication instance.
- **Source endpoint is outside the VPC used by the replication instance (using a NAT gateway)** – You can configure a network address translation (NAT) gateway using a single elastic IP address bound to a single elastic network interface. This NAT gateway receives a NAT identifier (nat-#####).

In some cases, the VPC includes a default route to that NAT gateway instead of the internet gateway. In such cases, the replication instance instead appears to contact the database endpoint using the public IP address of the NAT gateway. Here, the ingress to the database endpoint outside the VPC needs to allow ingress from the NAT address instead of the replication instance's public IP address.

For information about using your own on-premises name server, see [Using your own on-premises name server \(p. 90\)](#).

## CDC is stuck after full load

Slow or stuck replication changes can occur after a full load migration when several AWS DMS settings conflict with each other.

For example, suppose that the **Target table preparation mode** parameter is set to **Do nothing** or **Truncate**. In this case, you have instructed AWS DMS to do no setup on the target tables, including creating primary and unique indexes. If you haven't created primary or unique keys on the target tables, AWS DMS does a full table scan for each update. This approach can affect performance significantly.

## Primary key violation errors occur when you restart a task

This error can occur when data remains in the target database from a previous migration task. If the **Target table preparation mode** option is set to **Do nothing**, AWS DMS doesn't do any preparation on the target table, including cleaning up data inserted from a previous task.

To restart your task and avoid these errors, remove rows inserted into the target tables from the previous running of the task.

## Initial load of a schema fails

In some cases, the initial load of your schemas might fail with an error of `Operation:getSchemaListDetails:errType=, status=0, errMessage=, errDetails=`.

In such cases, the user account used by AWS DMS to connect to the source endpoint doesn't have the necessary permissions.

## Tasks fail with an unknown error

The cause of unknown types of error can be varied. However, often we find that the issue involves insufficient resources allocated to the AWS DMS replication instance.

To make sure that your replication instance has enough resources to perform the migration, check your instance's use of CPU, memory, swap files, and IOPS. For more information on monitoring, see [AWS Database Migration Service metrics \(p. 562\)](#).

## Task restart loads tables from the beginning

AWS DMS restarts table loading from the beginning when it hasn't finished the initial load of a table. When a task is restarted, AWS DMS reloads tables from the beginning when the initial load didn't complete.

## Number of tables per task causes issues

There is no set limit on the number of tables per replication task. However, we recommend limiting the number of tables in a task to less than 60,000, as a rule of thumb. Resource use can often be a bottleneck when a single task uses more than 60,000 tables.

## Tasks fail when a primary key is created on a LOB column

In FULL LOB or LIMITED LOB mode, AWS DMS doesn't support replication of primary keys that are LOB data types.

DMS initially migrates a row with a LOB column as null, then later updates the LOB column. So, when the primary key is created on a LOB column, the initial insert fails since the primary key can't be null. As a workaround, add another column as primary key and remove the primary key from the LOB column.

## Duplicate records occur on a target table without a primary key

Running a full load and CDC task can create duplicate records on target tables that don't have a primary key or unique index. To avoid duplicating records on target tables during full load and CDC tasks, make sure that target tables have a primary key or unique index.

## Source endpoints fall in the reserved IP range

If an AWS DMS source database uses an IP address within the reserved IP range of 192.168.0.0/24, the source endpoint connection test fails. The following steps provide a possible workaround:

1. Find one Amazon EC2 instance that isn't in the reserved range that can communicate to the source database at 192.168.0.0/24.
2. Install a socat proxy and run it. The following shows an example.

```
yum install socat

socat -d -d -l /tmp/local2 tcp4-listen:database_port,bind=0.0.0.0,reuseaddr,fork
    tcp4:source_database_ip_address:database_port
&
```

Use the EC2 instance IP address and the database port given preceding for the AWS DMS endpoint. Make sure that the endpoint has the security group that allows AWS DMS to talk to it at the database port.

## Timestamps are garbled in Amazon Athena queries

If timestamps are garbled in Athena queries, use the AWS Management Console or the [ModifyEndpoint](#) action to set the `parquetTimestampInMillisecond` value for your Amazon S3 endpoint to `true`. For more information, see [S3Settings](#).

## Troubleshooting issues with Oracle

Following, you can learn about troubleshooting issues specific to using AWS DMS with Oracle databases.

### Topics

- [Pulling data from views \(p. 659\)](#)
- [Migrating LOBs from Oracle 12c \(p. 660\)](#)
- [Switching between Oracle LogMiner and Binary Reader \(p. 660\)](#)
- [Error: Oracle CDC stopped 122301 oracle CDC maximum retry counter exceeded. \(p. 660\)](#)
- [Automatically add supplemental logging to an Oracle source endpoint \(p. 661\)](#)
- [LOB changes aren't being captured \(p. 661\)](#)
- [Error: ORA-12899: Value too large for column column-name \(p. 661\)](#)
- [NUMBER data type being misinterpreted \(p. 661\)](#)
- [Records missing during full load \(p. 662\)](#)
- [Table Error \(p. 662\)](#)
- [Error: Cannot retrieve Oracle archived Redo log destination ids \(p. 662\)](#)
- [Evaluating read performance of Oracle redo or archive logs \(p. 662\)](#)

## Pulling data from views

You can pull data once from a view; you can't use it for ongoing replication. To be able to extract data from views, you must add the following code to **Extra connection attributes** in the **Advanced** section of the Oracle source endpoint page. When you extract data from a view, the view is shown as a table on the target schema.

```
exposeViews=true
```

## Migrating LOBs from Oracle 12c

AWS DMS can use two methods to capture changes to an Oracle database, Binary Reader and Oracle LogMiner. By default, AWS DMS uses Oracle LogMiner to capture changes. However, on Oracle 12c, Oracle LogMiner doesn't support LOB columns. To capture changes to LOB columns on Oracle 12c, use Binary Reader.

## Switching between Oracle LogMiner and Binary Reader

AWS DMS can use two methods to capture changes to a source Oracle database, Binary Reader and Oracle LogMiner. Oracle LogMiner is the default. To switch to using Binary Reader for capturing changes, do the following:

### To use binary reader for capturing changes

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Endpoints**.
3. Choose the Oracle source endpoint that you want to use Binary Reader.
4. Choose **Modify**.
5. Choose **Advanced**, and then add the following code for **Extra connection attributes**.

```
useLogminerReader=N
```

6. Use an Oracle developer tool such as SQL-Plus to grant the following additional privilege to the AWS DMS user account used to connect to the Oracle endpoint.

```
SELECT ON V_$TRANSPORTABLE_PLATFORM
```

## Error: Oracle CDC stopped 122301 oracle CDC maximum retry counter exceeded.

This error occurs when the needed Oracle archive logs have been removed from your server before AWS DMS was able to use them to capture changes. Increase your log retention policies on your database server. For an Amazon RDS database, run the following procedure to increase log retention. For example, the following code increases log retention on an Amazon RDS DB instance to 24 hours.

```
exec rdsadmin.rdsadmin_util.set_configuration('archivelog retention hours',24);
```

## Automatically add supplemental logging to an Oracle source endpoint

By default, AWS DMS has supplemental logging turned off. To automatically turn on supplemental logging for a source Oracle endpoint, do the following:

### To add supplemental logging to a source oracle endpoint

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Endpoints**.
3. Choose the Oracle source endpoint that you want to add supplemental logging to.
4. Choose **Modify**.
5. Choose **Advanced**, and then add the following code to the **Extra connection attributes** text box:

```
addSupplementalLogging=Y
```

6. Choose **Modify**.

## LOB changes aren't being captured

Currently, a table must have a primary key for AWS DMS to capture LOB changes. If a table that contains LOBs doesn't have a primary key, there are several actions you can take to capture LOB changes:

- Add a primary key to the table. This can be as simple as adding an ID column and populating it with a sequence using a trigger.
- Create a materialized view of the table that includes a system-generated ID as the primary key and migrate the materialized view rather than the table.
- Create a logical standby, add a primary key to the table, and migrate from the logical standby.

## Error: ORA-12899: Value too large for column *column-name*

The error "ORA-12899: value too large for column *column-name*" is often caused by a couple of issues.

In one of these issues, there's a mismatch in the character sets used by the source and target databases.

In another of these issues, national language support (NLS) settings differ between the two databases. A common cause of this error is when the source database NLS\_LENGTH\_SEMANTICS parameter is set to CHAR and the target database NLS\_LENGTH\_SEMANTICS parameter is set to BYTE.

## NUMBER data type being misinterpreted

The Oracle NUMBER data type is converted into various AWS DMS data types, depending on the precision and scale of NUMBER. These conversions are documented here [Source data types for Oracle \(p. 172\)](#). The way the NUMBER type is converted can also be affected by using extra connection attributes for the source Oracle endpoint. These extra connection attributes are documented in [Extra connection attributes when using Oracle as a source for AWS DMS \(p. 165\)](#).

## Records missing during full load

When performing a full load, AWS DMS looks for open transactions at the database level and waits for the transaction to be committed. For example, based on the task setting `TransactionConsistencyTimeout=600`, AWS DMS waits for 10 minutes even if the open transaction is on a table not included in table mapping. But if the open transaction is on a table included in table mapping, and the transaction isn't committed in time, missing records in the target table result.

You can modify the `TransactionConsistencyTimeout` task setting and increase wait time if you know that open transactions will take longer to commit.

Also, note the default value of the `FailOnTransactionConsistencyBreached` task setting is `false`. This means AWS DMS continues to apply other transactions but open transactions are missed. If you want the task to fail when open transactions aren't closed in time, you can set `FailOnTransactionConsistencyBreached` to `true`.

## Table Error

`Table Error` appears in table statistics during replication if a `WHERE` clause doesn't reference a primary key column, and supplemental logging isn't used for all columns.

To fix this issue, turn on supplemental logging for all columns of the referenced table. For more information, see [Setting up supplemental logging \(p. 142\)](#).

## Error: Cannot retrieve Oracle archived Redo log destination ids

This error occurs when your Oracle source doesn't have any archive logs generated or `V$ARCHIVED_LOG` is empty. You can resolve the error by switching logs manually.

For an Amazon RDS database, run the following procedure to switch log files. The `switch_logfile` procedure doesn't have parameters.

```
exec rdsadmin.rdsadmin_util.switch_logfile;
```

For a self-managed Oracle source database, use the following command to force a log switch.

```
ALTER SYSTEM SWITCH LOGFILE ;
```

## Evaluating read performance of Oracle redo or archive logs

If you experience performance issues with your Oracle source, you can evaluate the read performance of your Oracle redo or archive logs to find ways to improve performance. To test the redo or archive log read performance, use the [AWS DMS diagnostic Amazon machine image \(p. 683\)](#) (AMI).

You can use the AWS DMS diagnostic AMI to do the following:

- Use the bFile method to evaluate redo log file performance.
- Use the LogMiner method to evaluate redo log file performance.

- Use the PL/SQL (`dbms_lob.read`) method to evaluate redo log file performance.
- Use Single-thread to evaluate read performance on ASMFile.
- Use Multi-threads to evaluate read performance on ASMFile.
- Use Direct OS Readfile() Windows or Pread64 Linux function to evaluate the redo log file.

Then you can take remedial steps based upon the results.

### To test read performance on an Oracle redo or archive log file

1. Create an AWS DMS diagnostic AMI Amazon EC2 instance and connect to it.

For more information see, [Working with the AWS DMS diagnostic AMI \(p. 683\)](#).

2. Run the `awsreplperf` command.

```
$ awsreplperf
```

The command displays the AWS DMS Oracle Read Performance Utility options.

```
0. Quit
1. Read using Bfile
2. Read using LogMiner
3. Read file PL/SQL (dms_lob.read)
4. Read ASMFile Single Thread
5. Read ASMFile Multi Thread
6. Readfile() function
```

3. Select an option from the list.
4. Enter the following database connection and archive log information.

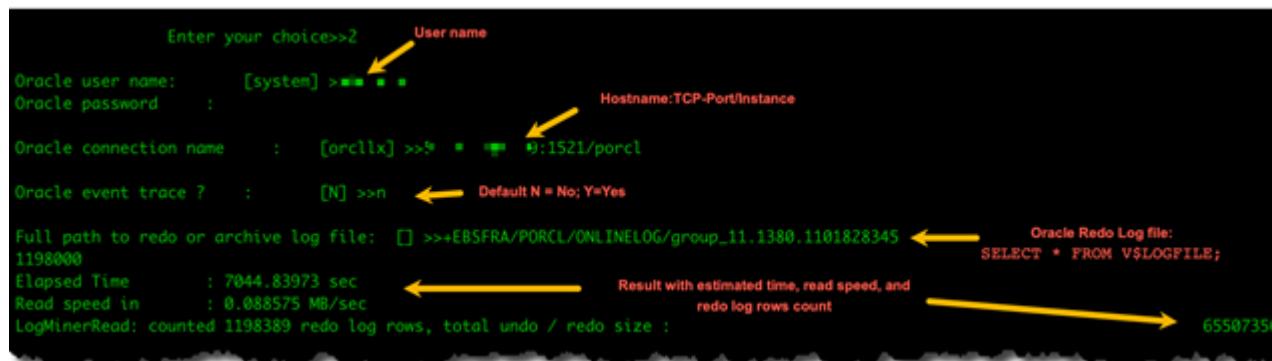
```
Oracle user name [system]:
Oracle password:

Oracle connection name [orcllx]:
Connection format hostname:port/instance

Oracle event trace? [N]:
Default N = No or Y = Yes

Path to redo or archive log file []:
```

5. Examine the output displayed for relevant read performance information. For example, the following shows output that can result from selecting option number 2, **Read using LogMiner**.



The screenshot shows a terminal window with the following text:

```
Enter your choice>>2
Username
Oracle user name:      [system] >sys * *
Hostname:TCP-Port/Instance
Oracle password :      :

Oracle connection name :  [orcllx] >>* * * 1521/orcl
Default N = No; Y=Yes

Oracle event trace ? :  [N] >>n
Full path to redo or archive log file:  <-->+>EBSFRA/PORCL/ONLINELOG/group_11.1380.1101828345
Elapsed Time       : 7044.83973 sec
Read speed in     : 0.088575 MB/sec
LogMinerRead: counted 1198389 redo log rows, total undo / redo size : 6550735
```

Annotations with yellow arrows point to specific fields and output lines:

- An arrow points to the 'Username' prompt with the label 'Username'.
- An arrow points to the 'Hostname:TCP-Port/Instance' prompt with the label 'Hostname:TCP-Port/Instance'.
- An arrow points to the 'Default N = No; Y=Yes' prompt with the label 'Default N = No; Y=Yes'.
- An arrow points to the 'Full path to redo or archive log file:' field with the label 'Full path to redo or archive log file:'.
- An arrow points to the 'Elapsed Time' line with the label 'Result with estimated time, read speed, and redo log rows count'.
- An arrow points to the 'LogMinerRead' output line with the label '6550735'.

6. To exit the utility, enter **0** (zero).

## Next steps

- When results show that read speed is below an acceptable threshold, run the [Oracle diagnostic support script \(p. 675\)](#) on the endpoint, review Wait Time, Load Profile, and IO Profile sections. Then adjust any abnormal configuration that might improve read performance. For example, if your redo log files are up to 2 GB, try increasing LOG\_BUFFER to 200 MB to help improve performance.
- Review [AWS DMS Best Practices \(p. 85\)](#) to make sure your DMS replication instance, task, and endpoints are configured optimally.

# Troubleshooting issues with MySQL

Following, you can learn about troubleshooting issues specific to using AWS DMS with MySQL databases.

## Topics

- [CDC task failing for Amazon RDS DB instance endpoint because binary logging disabled \(p. 664\)](#)
- [Connections to a target MySQL instance are disconnected during a task \(p. 664\)](#)
- [Adding autocommit to a MySQL-compatible endpoint \(p. 665\)](#)
- [Disable foreign keys on a target MySQL-compatible endpoint \(p. 665\)](#)
- [Characters replaced with question mark \(p. 666\)](#)
- ["Bad event" log entries \(p. 666\)](#)
- [Change data capture with MySQL 5.5 \(p. 666\)](#)
- [Increasing binary log retention for Amazon RDS DB instances \(p. 666\)](#)
- [Log message: Some changes from the source database had no impact when applied to the target database. \(p. 666\)](#)
- [Error: Identifier too long \(p. 666\)](#)
- [Error: Unsupported character set causes field data conversion to fail \(p. 667\)](#)
- [Error: Codepage 1252 to UTF8 \[120112\] a field data conversion failed \(p. 667\)](#)

## CDC task failing for Amazon RDS DB instance endpoint because binary logging disabled

This issue occurs with Amazon RDS DB instances because automated backups are disabled. Enable automatic backups by setting the backup retention period to a non-zero value.

## Connections to a target MySQL instance are disconnected during a task

If you have a task with LOBs that is getting disconnected from a MySQL target, you might see the following type of errors in the task log.

```
[TARGET_LOAD ]E: RetCode: SQL_ERROR SqlState: 08S01 NativeError: 2013 Message: [MySQL][ODBC 5.3(w) Driver][mysqld-5.7.16-log]Lost connection to MySQL server during query [122502] ODBC general error.
```

```
[TARGET_LOAD ]E: RetCode: SQL_ERROR SqlState: HY000 NativeError: 2006 Message: [MySQL][ODBC 5.3(w) Driver]MySQL server has gone away
```

[122502] ODBC general error.

In this case, you might need to adjust some of your task settings.

To solve the issue where a task is being disconnected from a MySQL target, do the following:

- Check that you have your database variable `max_allowed_packet` set large enough to hold your largest LOB.
- Check that you have the following variables set to have a large timeout value. We suggest you use a value of at least 5 minutes for each of these variables.
  - `net_read_timeout`
  - `net_write_timeout`
  - `wait_timeout`

For information about setting MySQL system variables, see [Server System Variables](#) in the [MySQL documentation](#).

## Adding autocommit to a MySQL-compatible endpoint

### To add autocommit to a target MySQL-compatible endpoint

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Endpoints**.
3. Choose the MySQL-compatible target endpoint that you want to add autocommit to.
4. Choose **Modify**.
5. Choose **Advanced**, and then add the following code to the **Extra connection attributes** text box:

```
Initstmt= SET AUTOCOMMIT=1
```

6. Choose **Modify**.

## Disable foreign keys on a target MySQL-compatible endpoint

You can disable foreign key checks on MySQL by adding the following to the **Extra Connection Attributes** in the **Advanced** section of the target MySQL, Amazon Aurora MySQL-Compatible Edition, or MariaDB endpoint.

### To disable foreign keys on a target MySQL-compatible endpoint

1. Sign in to the AWS Management Console and open the AWS DMS console at <https://console.aws.amazon.com/dms/v2/>.
2. Choose **Endpoints**.
3. Choose the MySQL, Aurora MySQL, or MariaDB target endpoint that you want to disable foreign keys.
4. Choose **Modify**.
5. Choose **Advanced**, and then add the following code to the **Extra connection attributes** text box:

```
Initstmt=SET FOREIGN_KEY_CHECKS=0
```

6. Choose **Modify**.

## Characters replaced with question mark

The most common situation that causes this issue is when the source endpoint characters have been encoded by a character set that AWS DMS doesn't support.

### "Bad event" log entries

"Bad event" entries in the migration logs usually indicate that an unsupported data definition language (DDL) operation was attempted on the source database endpoint. Unsupported DDL operations cause an event that the replication instance can't skip, so a bad event is logged.

To fix this issue, restart the task from the beginning. Doing this reloads the tables and starts capturing changes at a point after the unsupported DDL operation was issued.

### Change data capture with MySQL 5.5

AWS DMS change data capture (CDC) for Amazon RDS MySQL-compatible databases requires full image row-based binary logging, which isn't supported in MySQL version 5.5 or lower. To use AWS DMS CDC, you must upgrade your Amazon RDS DB instance to MySQL version 5.6.

### Increasing binary log retention for Amazon RDS DB instances

AWS DMS requires the retention of binary log files for change data capture. To increase log retention on an Amazon RDS DB instance, use the following procedure. The following example increases the binary log retention to 24 hours.

```
call mysql.rds_set_configuration('binlog retention hours', 24);
```

### Log message: Some changes from the source database had no impact when applied to the target database.

When AWS DMS updates a MySQL database column's value to its existing value, a message of `zero rows affected` is returned from MySQL. This behavior is unlike other database engines such as Oracle and SQL Server. These engines update one row, even when the replacing value is the same as the current one.

### Error: Identifier too long

The following error occurs when an identifier is too long:

```
TARGET_LOAD E: RetCode: SQL_ERROR SqlState: HY000 NativeError:  
1059 Message: MySQLhttp://ODBC 5.3(w) Driverhttp://mysqld-5.6.10Identifier  
name 'name' is too long 122502 ODBC general error. (ar_odbc_stmt.c:4054)
```

In some cases, you set AWS DMS to create the tables and primary keys in the target database. In these cases, DMS currently doesn't use the same names for the primary keys that were used in the source database. Instead, DMS creates the primary key name based on the table name. When the table name is long, the autogenerated identifier created can be longer than the allowed limits for MySQL.

To solve this issue, the current approach is to first precreate the tables and primary keys in the target database. Then use a task with the task setting **Target table preparation mode** set to **Do nothing** or **Truncate** to populate the target tables.

## Error: Unsupported character set causes field data conversion to fail

The following error occurs when an unsupported character set causes a field data conversion to fail:

```
"[SOURCE_CAPTURE ]E: Column 'column-name' uses an unsupported character set [120112]  
A field data conversion failed. (mysql_endpoint_capture.c:2154)
```

Check your database's parameters related to connections. The following command can be used to set these parameters.

```
SHOW VARIABLES LIKE '%char%';
```

## Error: Codepage 1252 to UTF8 [120112] a field data conversion failed

The following error can occur during a migration if you have non codepage-1252 characters in the source MySQL database.

```
[SOURCE_CAPTURE ]E: Error converting column 'column_xyz' in table  
'table_xyz' with codepage 1252 to UTF8 [120112] A field data conversion failed.  
(mysql_endpoint_capture.c:2248)
```

As a workaround, you can use the `CharsetMapping` extra connection attribute with your source MySQL endpoint to specify character set mapping. You might need to restart the AWS DMS migration task from the beginning if you add this extra connection attribute.

For example, the following extra connection attribute could be used for a MySQL source endpoint where the source character set is `utf8` or `latin1`. `65001` is the `UTF8` code page identifier.

```
CharsetMapping=utf8,65001
CharsetMapping=latin1,65001
```

## Troubleshooting issues with PostgreSQL

Following, you can learn about troubleshooting issues specific to using AWS DMS with PostgreSQL databases.

### Topics

- [JSON data types being truncated \(p. 668\)](#)
- [Columns of a user-defined data type not being migrated correctly \(p. 669\)](#)
- [Error: No schema has been selected to create in \(p. 669\)](#)
- [Deletes and updates to a table aren't being replicated using CDC \(p. 669\)](#)
- [Truncate statements aren't being propagated \(p. 669\)](#)
- [Preventing PostgreSQL from capturing DDL \(p. 669\)](#)
- [Selecting the schema where database objects for capturing DDL are created \(p. 669\)](#)
- [Oracle tables missing after migrating to PostgreSQL \(p. 670\)](#)
- [ReplicationSlotDiskUsage increases and restart\\_lsn stops moving forward during long transactions, such as ETL workloads \(p. 670\)](#)
- [Task using view as a source has no rows copied \(p. 670\)](#)

## JSON data types being truncated

AWS DMS treats the JSON data type in PostgreSQL as an LOB data type column. This means that the LOB size limitation when you use limited LOB mode applies to JSON data.

For example, suppose that limited LOB mode is set to 4,096 KB. In this case, any JSON data larger than 4,096 KB is truncated at the 4,096 KB limit and fails the validation test in PostgreSQL.

The following log information shows JSON that was truncated due to the limited LOB mode setting and failed validation.

```
03:00:49
2017-09-19T03:00:49 [TARGET_APPLY ]E: Failed to execute statement:
  'UPDATE "public"."delivery_options_quotes" SET "id"=? , "enabled"=? ,
  "new_cart_id"=? , "order_id"=? , "user_id"=? , "zone_id"=? , "quotes"=? ,
  "start_at"=? , "end_at"=? , "last_quoted_at"=? , "created_at"=? ,
  "updated_at"=? WHERE "id"=? ' [1022502] (ar_odbc_stmt
2017-09-19T03:00:49 [TARGET_APPLY ]E: Failed to execute statement:
  'UPDATE "public"."delivery_options_quotes" SET "id"=? , "enabled"=? ,
  "new_cart_id"=? , "order_id"=? , "user_id"=? , "zone_id"=? , "quotes"=? ,
  "start_at"=? , "end_at"=? , "last_quoted_at"=? , "created_at"=? ,
  "updated_at"=? WHERE "id"=? ' [1022502] (ar_odbc_stmt.c:2415)
#
03:00:49
2017-09-19T03:00:49 [TARGET_APPLY ]E: RetCode: SQL_ERRORR SqlState:
  22P02 NativeError: 1 Message: ERROR: invalid input syntax for type json;;
  Error while executing the query [1022502] (ar_odbc_stmt.c:2421)
2017-09-19T03:00:49 [TARGET_APPLY ]E: RetCode: SQL_ERRORR SqlState:
  22P02 NativeError: 1 Message: ERROR: invalid input syntax for type json;;
  Error while executing the query [1022502] (ar_odbc_stmt.c:2421)
```

## Columns of a user-defined data type not being migrated correctly

When replicating from a PostgreSQL source, AWS DMS creates the target table with the same data types for all columns, apart from columns with user-defined data types. In such cases, the data type is created as "character varying" in the target.

## Error: No schema has been selected to create in

In some case, you might see the error "SQL\_ERROR SqlState: 3F000 NativeError: 7 Message: ERROR: no schema has been selected to create in".

This error can occur when your JSON table mapping contains a wildcard value for the schema but the source database doesn't support that value.

## Deletes and updates to a table aren't being replicated using CDC

Delete and update operations during change data capture (CDC) are ignored if the source table doesn't have a primary key. AWS DMS supports change data capture (CDC) for PostgreSQL tables with primary keys.

If a table doesn't have a primary key, the write-ahead (WAL) logs don't include a before image of the database row. In this case, AWS DMS can't update the table. For delete operations to be replicated, create a primary key on the source table.

## Truncate statements aren't being propagated

When using change data capture (CDC), TRUNCATE operations aren't supported by AWS DMS.

## Preventing PostgreSQL from capturing DDL

You can prevent a PostgreSQL target endpoint from capturing DDL statements by adding the following **Extra Connection Attribute** statement. The **Extra Connection Attribute** parameter is available in the **Advanced** tab of the source endpoint.

```
captureDDLs=N
```

## Selecting the schema where database objects for capturing DDL are created

You can control what schema the database objects related to capturing DDL are created in. Add the following **Extra Connection Attribute** statement. The **Extra Connection Attribute** parameter is available in the **Advanced** tab of the source endpoint.

```
ddlArtifactsSchema=xyzddlschema
```

## Oracle tables missing after migrating to PostgreSQL

In this case, your tables and data are generally still accessible.

Oracle defaults to uppercase table names, and PostgreSQL defaults to lowercase table names. When you perform a migration from Oracle to PostgreSQL, we suggest that you supply certain transformation rules under your task's table-mapping section. These are transformation rules to convert the case of your table names.

If you migrated your tables without using transformation rules to convert the case of your table names, enclose your table names in quotation marks when referencing them.

## ReplicationSlotDiskUsage increases and restart\_lsn stops moving forward during long transactions, such as ETL workloads

When logical replication is enabled, the maximum number of changes kept in memory per transaction is 4MB. After that, changes are spilled to disk. As a result ReplicationSlotDiskUsage increases, and restart\_lsn doesn't advance until the transaction is completed/aborted and the rollback finishes. Since it is a long transaction, it can take a long time to rollback.

So, avoid long running transactions when logical replication is enabled. Instead, try to break the transaction into several smaller transactions.

## Task using view as a source has no rows copied

To migrate a view, set table-type to all or view. For more information, see [Specifying table selection and transformations rules from the console \(p. 491\)](#).

Sources that support views include the following.

- Oracle
- Microsoft SQL Server
- MySQL
- PostgreSQL
- IBM Db2 LUW
- SAP Adaptive Server Enterprise (ASE)

## Troubleshooting issues with Microsoft SQL Server

Following, you can learn about troubleshooting issues specific to using AWS DMS with Microsoft SQL Server databases.

### Topics

- [Errors capturing changes for SQL server database \(p. 671\)](#)
- [Missing identity columns \(p. 671\)](#)
- [Error: SQL Server doesn't support publications \(p. 671\)](#)

- [Changes don't appear in your target \(p. 671\)](#)
- [Non-uniform table mapped across partitions \(p. 671\)](#)

## Errors capturing changes for SQL server database

Errors during change data capture (CDC) can often indicate that one of the prerequisites wasn't met. For example, the most common overlooked prerequisite is a full database backup. The task log indicates this omission with the following error:

```
SOURCE_CAPTURE E: No FULL database backup found (under the 'FULL' recovery model).  
To enable all changes to be captured, you must perform a full database backup.  
120438 Changes may be missed. (sqlserver_log_queries.c:2623)
```

Review the prerequisites listed for using SQL Server as a source in [Using a Microsoft SQL Server database as a source for AWS DMS \(p. 175\)](#).

## Missing identity columns

AWS DMS doesn't support identity columns when you create a target schema. You must add them after the initial load has completed.

## Error: SQL Server doesn't support publications

The following error is generated when you use SQL Server Express as a source endpoint:

```
RetCode: SQL_ERROR SqlState: HY000 NativeError: 21106  
Message: This edition of SQL Server does not support publications.
```

AWS DMS currently doesn't support SQL Server Express as a source or target.

## Changes don't appear in your target

AWS DMS requires that a source SQL Server database be in either 'FULL' or 'BULK LOGGED' data recovery model in order to consistently capture changes. The 'SIMPLE' model isn't supported.

The SIMPLE recovery model logs the minimal information needed to allow users to recover their database. All inactive log entries are automatically truncated when a checkpoint occurs.

All operations are still logged. However, as soon as a checkpoint occurs the log is automatically truncated. This truncation means that the log becomes available for reuse and older log entries can be overwritten. When log entries are overwritten, changes can't be captured. This issue is why AWS DMS doesn't support the SIMPLE data recovery model. For information on other required prerequisites for using SQL Server as a source, see [Using a Microsoft SQL Server database as a source for AWS DMS \(p. 175\)](#).

## Non-uniform table mapped across partitions

During change data capture (CDC), migration of a table with a specialized structure is suspended when AWS DMS can't properly perform CDC on the table. Messages like these are issued:

```
[SOURCE_CAPTURE ]W: Table is not uniformly mapped across partitions. Therefore - it is excluded from CDC (sqlserver_log_metadata.c:1415)
[SOURCE_CAPTURE ]I: Table has been mapped and registered for CDC.
(sqlserver_log_metadata.c:835)
```

When running CDC on SQL Server tables, AWS DMS parses the SQL Server tlogs. On each tlog record, AWS DMS parses hexadecimal values containing data for columns that were inserted, updated, or deleted during a change.

To parse the hexadecimal record, AWS DMS reads the table metadata from the SQL Server system tables. Those system tables identify what the specially structured table columns are and reveal some of their internal properties, such as "xoffset" and "null bit position".

AWS DMS expects that metadata to be the same for all raw partitions of the table. But in some cases, specially structured tables don't have the same metadata on all of their partitions. In these cases, AWS DMS can suspend CDC on that table to avoid parsing changes incorrectly and providing the target with incorrect data. Workarounds include the following:

- If the table has a clustered index, perform an index rebuild.
- If the table doesn't have a clustered index, add a clustered index to the table (you can drop it later if you want).

## Troubleshooting issues with Amazon Redshift

Following, you can learn about troubleshooting issues specific to using AWS DMS with Amazon Redshift databases.

### Topics

- [Loading in to an Amazon Redshift cluster in a different AWS Region \(p. 672\)](#)
- [Error: Relation "awsdms\\_apply\\_exceptions" already exists \(p. 672\)](#)
- [Errors with tables whose name begins with "awsdms\\_changes" \(p. 673\)](#)
- [Seeing tables in clusters with names like dms.awsdms\\_changes00000000XXXX \(p. 673\)](#)
- [Permissions required to work with Amazon Redshift \(p. 673\)](#)

## Loading in to an Amazon Redshift cluster in a different AWS Region

You can't load into an Amazon Redshift cluster in a different AWS Region than your AWS DMS replication instance. DMS requires that your replication instance and your Amazon Redshift cluster be in the same Region.

## Error: Relation "awsdms\_apply\_exceptions" already exists

The error "Relation 'awsdms\_apply\_exceptions' already exists" often occurs when a Redshift endpoint is specified as a PostgreSQL endpoint. To fix this issue, modify the endpoint and change the **Target engine** to "redshift."

## Errors with tables whose name begins with "awsdms\_changes"

Table error messages with names that begin with "awsdms\_changes" can occur when two tasks trying to load data into the same Amazon Redshift cluster run concurrently. Due to the way temporary tables are named, concurrent tasks can conflict when updating the same table.

## Seeing tables in clusters with names like dms.awsdms\_changes000000000XXXX

AWS DMS creates temporary tables when data is being loaded from files stored in Amazon S3. The names of these temporary tables each have the prefix dms . awsdms\_changes. These tables are required so AWS DMS can store data when it is first loaded and before it is placed in its final target table.

## Permissions required to work with Amazon Redshift

To use AWS DMS with Amazon Redshift, the user account that you use to access Amazon Redshift must have the following permissions:

- CRUD (Choose, Insert, Update, Delete)
- Bulk load
- Create, alter, drop (if required by the task's definition)

To see the prerequisites required for using Amazon Redshift as a target, see [Using an Amazon Redshift database as a target for AWS Database Migration Service \(p. 315\)](#).

## Troubleshooting issues with Amazon Aurora MySQL

Following, you can learn about troubleshooting issues specific to using AWS DMS with Amazon Aurora MySQL databases.

### Topics

- [Error: CHARACTER SET UTF8 fields terminated by ',' enclosed by "" lines terminated by '\n' \(p. 673\)](#)

## Error: CHARACTER SET UTF8 fields terminated by ',', enclosed by "" lines terminated by '\n'

If you are using Amazon Aurora MySQL as a target, you might see an error like the following in the logs. This type of error usually indicates that you have ANSI\_QUOTES as part of the SQL\_MODE parameter. Having ANSI\_QUOTES as part of the SQL\_MODE parameter causes double quotation marks to be handled like quotation marks and can create issues when you run a task.

To fix this error, remove ANSI\_QUOTES from the SQL\_MODE parameter.

```
2016-11-02T14:23:48 [TARGET_LOAD ]E: Load data sql statement. load data local infile
```

```
"/rdsdbdata/data/tasks/7X04FJHCVON7TYTLQ6RX3CQHDU/data_files/4/LOAD000001DF.csv" into
table
`VOSPUSER`.`SANDBOX_SRC_FILE` CHARACTER SET UTF8 fields terminated by ','
enclosed by ''' lines terminated by '\n'(`SANDBOX_SRC_FILE_ID`, `SANDBOX_ID`,
`FILENAME`, `LOCAL_PATH`, `LINES_OF_CODE`, `INSERT_TS`, `MODIFIED_TS`, `MODIFIED_BY`,
`RECORD_VER`, `REF_GUID`, `PLATFORM_GENERATED`, `ANALYSIS_TYPE`, `SANITIZED`, `DYN_TYPE`,
`CRAWL_STATUS`, `ORIG_EXEC_UNIT_VER_ID`); (provider_syntax_manager.c:2561)
```

## Troubleshooting issues with SAP ASE

Following, you can learn about troubleshooting issues specific to using AWS DMS with SAP ASE databases.

### Error: LOB columns have NULL values when source has a composite unique index with NULL values

When using SAP ASE as a source with tables configured with a composite unique index that allows NULL values, LOB values might not migrate during ongoing replication. This behavior is usually the result of ANSI\_NULL set to 1 by default on the DMS replication instance client.

To ensure that LOB fields migrate correctly, include the Extra Connection Attribute (ECA) 'AnsiNull=0' to the AWS DMS source endpoint for the task.

## Troubleshooting issues with IBM Db2

Following, you can learn about troubleshooting issues specific to using AWS DMS with IBM Db2 databases.

### Error: Resume from timestamp is not supported Task

For ongoing replication (CDC), if you plan to start replication from a specific timestamp, set the connection attribute StartFromContext to the required timestamp. For more information, see [Extra connection attributes when using Db2 LUW \(p. 263\)](#). Setting StartFromContext to the required timestamp prevents the following issue:

```
Last Error Resume from timestamp is not supported Task error notification received from
subtask 0, thread 0 [reptask/replicationtask.c:2822] [1020455] 'Start from timestamp' was
blocked to prevent Replicate from
scanning the log (to find the timestamp). When using IBM DB2 for LUW, 'Start from
timestamp' is only supported if an actual
change was captured by this Replicate task earlier to the specified timestamp.
```

## Working with diagnostic support scripts in AWS DMS

If you encounter an issue when working with AWS DMS, your support engineer might need more information about either your source or target database. We want to make sure that AWS Support gets as much of the required information as possible in the shortest possible time. Therefore, we developed scripts to query this information for several of the major relational database engines.

If a support script is available for your database, you can download it using the link in the corresponding script topic described following. After verifying and reviewing the script (described following), you can run it according to the procedure described in the script topic. When the script run is complete, you can upload its output to your AWS Support case (again, described following).

Before running the script, you can detect any errors that might have been introduced when downloading or storing the support script. To do this, compare the checksum for the script file with a value provided by AWS. AWS uses the SHA256 algorithm for the checksum.

### To verify the support script file using a checksum

1. Open the latest checksum file provided to verify these support scripts at <https://d2pwp9zz55emqw.cloudfront.net/sha256Check.txt>. For example, the file might have content like the following.

```
MYSQL  dfafdf0d511477c699f96c64693ad0b1547d47e74d5c5f2f2025b790b1422e3c8
ORACLE  6c41ebcfcc99518cfa8a10cb2ce8943b153b2cc7049117183d0b5de3d551bc312
POSTGRES  6ccd274863d14f6f3146fbdbbba43f2d8d4c6a4c25380d7b41c71883aa4f9790
SQL_SERVER  971a6f2c46aec8d083d2b3b6549b1e9990af3a15fe4b922e319f4fdd358debe7
```

2. Run the SHA256 validation command for your operating system in the directory that contains the support file. For example, on the macOS operating system you can run the following command on an Oracle support script described later in this topic.

```
shasum -a 256 awsdms_support_collector_oracle.sql
```

3. Compare the results of the command with the value shown in the latest sha256Check.txt file that you opened. The two values should match. If they don't, contact your support engineer about the mismatch and how you can obtain a clean support script file.

If you have a clean support script file, before running the script make sure to read and understand the SQL from both a performance and security perspective. If you aren't comfortable running any of the SQL in this script, you can comment out or remove the problem SQL. You can also consult with your support engineer about any acceptable workarounds.

Upon successful completion and unless otherwise noted, the script returns output in a readable HTML format. The script is designed to exclude from this HTML any data or security details that might compromise your business. It also makes no modifications to your database or its environment. However, if you find any information in the HTML that you are uncomfortable sharing, feel free to remove the problem information before uploading the HTML. When the HTML is acceptable, upload it using the **Attachments** in the **Case details** of your support case.

Each of the following topics describes the scripts available for a supported AWS DMS database and how to run them. Your support engineer will direct you to a specific script documented following.

#### Topics

- [Oracle diagnostic support scripts \(p. 675\)](#)
- [SQL Server diagnostic support scripts \(p. 678\)](#)
- [Diagnostic support scripts for MySQL-compatible databases \(p. 680\)](#)
- [PostgreSQL diagnostic support scripts \(p. 681\)](#)

## Oracle diagnostic support scripts

Following, you can find the diagnostic support scripts available to analyze an on-premises or Amazon RDS for Oracle database in your AWS DMS migration configuration. These scripts work with either a

source or target endpoint. The scripts are all written to run in the SQL\*Plus command-line utility. For more information on using this utility, see [A Using SQL Command Line](#) in the Oracle documentation.

Before running the script, ensure that the user account that you use has the necessary permissions to access your Oracle database. The permissions settings shown assume a user created as follows.

```
CREATE USER script_user IDENTIFIED BY password;
```

For an on-premises database, set the minimum permissions as shown following for *script\_user*.

```
GRANT CREATE SESSION TO script_user;
GRANT SELECT on V$DATABASE to script_user;
GRANT SELECT on V$VERSION to script_user;
GRANT SELECT on GV$SGA to script_user;
GRANT SELECT on GV$INSTANCE to script_user;
GRANT SELECT on GV$DATAGUARD_CONFIG to script_user;
GRANT SELECT on GV$LOG to script_user;
GRANT SELECT on DBA_TABLESPACES to script_user;
GRANT SELECT on DBA_DATA_FILES to script_user;
GRANT SELECT on DBA_SEGMENTS to script_user;
GRANT SELECT on DBA_LOBS to script_user;
GRANT SELECT on V$ARCHIVED_LOG to script_user;
GRANT SELECT on DBA_TAB_MODIFICATIONS to script_user;
GRANT SELECT on DBA_TABLES to script_user;
GRANT SELECT on DBA_TAB_PARTITIONS to script_user;
GRANT SELECT on DBA_MVIEWS to script_user;
GRANT SELECT on DBA_OBJECTS to script_user;
GRANT SELECT on DBA_TAB_COLUMNS to script_user;
GRANT SELECT on DBA_LOG_GROUPS to script_user;
GRANT SELECT on DBA_LOG_GROUP_COLUMNS to script_user;
GRANT SELECT on V$ARCHIVE_DEST to script_user;
GRANT SELECT on DBA_SYS_PRIVS to script_user;
GRANT SELECT on DBA_TAB_PRIVS to script_user;
GRANT SELECT on DBA_TYPES to script_user;
GRANT SELECT on DBA_CONSTRAINTS to script_user;
GRANT SELECT on V$TRANSACTION to script_user;
GRANT SELECT on GV$ASM_DISK_STAT to script_user;
GRANT SELECT on GV$SESSION to script_user;
GRANT SELECT on GV$SQL to script_user;
GRANT SELECT on DBA_ENCRYPTED_COLUMNS to script_user;
GRANT SELECT on DBA_PDBS to script_user;

GRANT EXECUTE on dbms_utility to script_user;
```

For an Amazon RDS database, set the minimum permissions as shown following.

```
GRANT CREATE SESSION TO script_user;
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$DATABASE', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$VERSION', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('GV_$SGA', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('GV_$INSTANCE', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('GV_DATAGUARD_CONFIG', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('GV$_LOG', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TABLESPACES', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_DATA_FILES', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_SEGMENTS', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_LOBS', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V$_ARCHIVED_LOG', 'script_user', 'SELECT');
exec
  rdsadmin.rdsadmin_util.grant_sys_object('DBA_TAB_MODIFICATIONS', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TABLES', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TAB_PARTITIONS', 'script_user', 'SELECT');
```

```
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_MVIEWS', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_OBJECTS', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TAB_COLUMNS', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_LOG_GROUPS', 'script_user', 'SELECT');
exec
rdsadmin.rdsadmin_util.grant_sys_object('DBA_LOG_GROUP_COLUMNS', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$ARCHIVE_DEST', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_SYS_PRIVS', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TAB_PRIVS', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_TYPES', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_CONSTRAINTS', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('V_$TRANSACTION', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('GV_$ASM_DISK_STAT', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('GV_$SESSION', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('GV_SQL', 'script_user', 'SELECT');
exec
rdsadmin.rdsadmin_util.grant_sys_object('DBA_ENCRYPTED_COLUMNS', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBA_PDBS', 'script_user', 'SELECT');
exec rdsadmin.rdsadmin_util.grant_sys_object('DBMSUTILITY', 'script_user', 'EXECUTE');
```

Following, you can find descriptions how to download, review, and run each SQL\*Plus support script available for Oracle. You can also find how to review and upload the output to your AWS Support case.

### Topics

- [awsdms\\_support\\_collector\\_oracle.sql script \(p. 677\)](#)

## awsdms\_support\_collector\_oracle.sql script

Download the [awsdms\\_support\\_collector\\_oracle.sql](#) script.

This script collects information about your Oracle database configuration. Remember to verify the checksum on the script, and if the checksum verifies, review the SQL code in the script to comment out any of the code that you are uncomfortable running. After you are satisfied with the integrity and content of the script, you can run it.

### To run the script and upload the results to your support case

1. Run the script from your database environment using the following SQL\*Plus command line.

```
SQL> @awsdms_support_collector_oracle.sql
```

```
<result>
```

The script displays a brief description and a prompt to either continue or abort the run. Press [Enter] to continue.

```
</result>
```

2. At the following prompt, enter the name of only one of the schemas that you want to migrate.
3. At the following prompt, enter the name of the user (*script\_user*) that you have defined to connect to the database.
4. At the following prompt, enter the number of days of data you want to examine, or accept the default. The script then collects the specified data from your database.

```
<result>
```

After the script completes, it displays the name of the output HTML file, for example `dms_support_oracle-2020-06-22-13-20-39-ORCL.html`. The script saves this file in your working directory.

</result>

5. Review this HTML file and remove any information that you are uncomfortable sharing. When the HTML is acceptable for you to share, upload the file to your AWS Support case. For more information on uploading this file, see [Working with diagnostic support scripts in AWS DMS \(p. 674\)](#).

## SQL Server diagnostic support scripts

Following, you can find a description of the diagnostic support scripts available to analyze an on-premises or Amazon RDS for SQL Server database in your AWS DMS migration configuration. These scripts work with either a source or target endpoint. For an on-premises database, run these scripts in the sqlcmd command-line utility. For more information on using this utility, see [sqlcmd - Use the utility](#) in the Microsoft documentation.

For an Amazon RDS database, you can't connect using the sqlcmd command-line utility. Instead, run these scripts using any client tool that connects to Amazon RDS SQL Server.

Before running the script, ensure that the user account that you use has the necessary permissions to access your SQL Server database. For both an on-premises and an Amazon RDS database, you can use the same permissions you use to access your SQL Server database without the SysAdmin role.

### To set up the minimum permissions to run for an on-premises SQL Server database

1. Create a new SQL Server account with password authentication using SQL Server Management Studio (SSMS), for example *on-prem-user*.
2. In the **User Mappings** section of SSMS, choose the **MSDB** and **MASTER** databases (which gives public permission), and assign the DB\_OWNER role to the database where you want to run the script.
3. Open the context (right-click) menu for the new account, and choose **Security** to explicitly grant the Connect SQL privilege.
4. Run the grant commands following.

```
GRANT VIEW SERVER STATE TO on-prem-user;
USE MSDB;
GRANT SELECT ON MSDB.DBO.BACKUPSET TO on-prem-user;
GRANT SELECT ON MSDB.DBO.BACKUPMEDIAFAMILY TO on-prem-user;
GRANT SELECT ON MSDB.DBO.BACKUPFILE TO on-prem-user;
```

### To run with the minimum permissions for an Amazon RDS SQL Server database

1. Create a new SQL Server account with password authentication using SQL Server Management Studio (SSMS), for example *rds-user*.
2. In the **User Mappings** section of SSMS, choose the **MSDB** database (which gives public permission), and assign the DB\_OWNER role to the database where you want to run the script.
3. Open the context (right-click) menu for the new account, and choose **Security** to explicitly grant the Connect SQL privilege.
4. Run the grant commands following.

```
GRANT VIEW SERVER STATE TO rds-user;
USE MSDB;
GRANT SELECT ON MSDB.DBO.BACKUPSET TO rds-user;
GRANT SELECT ON MSDB.DBO.BACKUPMEDIAFAMILY TO rds-user;
GRANT SELECT ON MSDB.DBO.BACKUPFILE TO rds-user;
```

The following topics describe how to download, review, and run each support script available for SQL Server. They also describe how to review and upload the script output to your AWS Support case.

**Topics**

- [awsdms\\_support\\_collector\\_sql\\_server.sql script \(p. 679\)](#)

## awsdms\_support\_collector\_sql\_server.sql script

Download the [awsdms\\_support\\_collector\\_sql\\_server.sql](#) script.

**Note**

Run this SQL Server diagnostic support script on SQL Server 2014 and later versions only.

This script collects information about your SQL Server database configuration. Remember to verify the checksum on the script, and if the checksum verifies, review the SQL code in the script to comment out any of the code that you are uncomfortable running. After you are satisfied with the integrity and content of the script, you can run it.

### To run the script for an on-premises SQL Server database

1. Run the script using the following sqlcmd command line.

```
sqlcmd -Uon-prem-user -Ppassword -SDMS-SQL17AG-N1 -y 0  
-iC:\Users\admin\awsdms_support_collector_sql_server.sql -oC:\Users\admin  
\DMS_Support_Report_SQLServer.html -dsqlserverdb01
```

The specified sqlcmd command parameters include the following:

- -U – Database user name.
  - -P – Database user password.
  - -S – SQL Server database server name.
  - -y – Maximum width of columns output from the sqlcmd utility. A value of 0 specifies columns of unlimited width.
  - -i – Path of the support script to run, in this case `awsdms_support_collector_sql_server.sql`.
  - -o – Path of the output HTML file, with a file name that you specify, containing the collected database configuration information.
  - -d – SQL Server database name.
2. After the script completes, review the output HTML file and remove any information that you are uncomfortable sharing. When the HTML is acceptable for you to share, upload the file to your AWS Support case. For more information on uploading this file, see [Working with diagnostic support scripts in AWS DMS \(p. 674\)](#).

With Amazon RDS for SQL Server, you can't connect using the sqlcmd command line utility, so use the following procedure.

### To run the script for an RDS SQL Server database

1. Run the script using any client tool that allows you to connect to RDS SQL Server as the Master user and save the output as an HTML file.
2. Review the output HTML file and remove any information that you are uncomfortable sharing. When the HTML is acceptable for you to share, upload the file to your AWS Support case. For more information on uploading this file, see [Working with diagnostic support scripts in AWS DMS \(p. 674\)](#).

# Diagnostic support scripts for MySQL-compatible databases

Following, you can find the diagnostic support scripts available to analyze an on-premises or Amazon RDS for MySQL-compatible database in your AWS DMS migration configuration. These scripts work with either a source or target endpoint. The scripts are all written to run on the MySQL SQL command line.

For information about installing the MySQL client, see [Installing MySQL Shell](#) in the MySQL documentation. For information about using the MySQL client, see [Using MySQL Shell Commands](#) in the MySQL documentation.

Before running a script, ensure that the user account that you use has the necessary permissions to access your MySQL-compatible database. Use the following procedure to create a user account and provide the minimum permissions needed to run this script.

## To set up a user account with the minimum permissions to run these scripts

1. Create the user to run the scripts.

```
create user 'username'@'hostname' identified by password;
```

2. Grant the select command on databases to analyze them.

```
grant select on database-name.* to username;  
grant replication client on *.* to username;
```

- 3.

```
grant execute on procedure mysql.rds_show_configuration to username;
```

The following topics describe how to download, review, and run each support script available for a MySQL-compatible database. They also describe how to review and upload the script output to your AWS Support case.

### Topics

- [awsdms\\_support\\_collector\\_MySQL.sql script \(p. 680\)](#)

## awsdms\_support\_collector\_MySQL.sql script

Download the [awsdms\\_support\\_collector\\_MySQL.sql](#) script.

This script collects information about your MySQL-compatible database configuration. Remember to verify the checksum on the script, and if the checksum verifies, review the SQL code in the script to comment out any of the code that you are uncomfortable running. After you are satisfied with the integrity and content of the script, you can run it.

Run the script after connecting to your database environment using the command line.

## To run this script and upload the results to your support case

1. Connect to your database using the following mysql command.

```
mysql -h hostname -P port -u username database-name
```

2. Run the script using the following mysql source command.

```
mysql> source awsdms_support_collector_MySQL_compatible_DB.sql
```

Review the generated report and remove any information that you are uncomfortable sharing. When the content is acceptable for you to share, upload the file to your AWS Support case. For more information on uploading this file, see [Working with diagnostic support scripts in AWS DMS \(p. 674\)](#).

#### Note

- If you already have a user account with required privileges described in [Diagnostic support scripts for MySQL-compatible databases \(p. 680\)](#), you can use the existing user account as well to run the script.
- Remember to connect to your database before running the script.
- The script generates its output in text format.
- Keeping security best practices in mind, if you create a new user account only to execute this MySQL diagnostic support script, we recommend that you delete this user account after successful execution of the script.

## PostgreSQL diagnostic support scripts

Following, you can find the diagnostic support scripts available to analyze any PostgreSQL RDBMS (on-premises, Amazon RDS, or Aurora PostgreSQL) in your AWS DMS migration configuration. These scripts work with either a source or target endpoint. The scripts are all written to run in the psql command-line utility.

Before running these scripts, ensure that the user account that you use has the following necessary permissions to access any PostgreSQL RDBMS:

- PostgreSQL 10.x or later – A user account with execute permission on the pg\_catalog.pg\_ls\_waldir function.
- PostgreSQL 9.x or earlier – A user account with default permissions.

We recommend using an existing account with the appropriate permissions to run these scripts.

If you need to create a new user account or grant permissions to an existing account to run these scripts, you can execute the following SQL commands for any PostgreSQL RDBMS based on the PostgreSQL version.

#### To grant account permissions to run these scripts for a PostgreSQL 10.x or later database

- Do one of the following:
  - For a new user account, run the following.

```
CREATE USER script_user WITH PASSWORD 'password';  
GRANT EXECUTE ON FUNCTION pg_catalog.pg_ls_waldir TO script_user;
```

- For an existing user account, run the following.

```
GRANT EXECUTE ON FUNCTION pg_catalog.pg_ls_waldir TO script_user;
```

## To grant account permissions to run these scripts for a PostgreSQL 9.x or earlier database

- Do one of the following:
  - For a new user account, run the following with default permissions.

```
CREATE USER script_user WITH PASSWORD password;
```

- For an existing user account, use the existing permissions.

### Note

These scripts do not support certain functionality related to finding WAL size for PostgreSQL 9.x and earlier databases. For more information, work with AWS Support.

The following topics describe how to download, review, and run each support script available for PostgreSQL. They also describe how to review and upload the script output to your AWS Support case.

### Topics

- [awsdms\\_support\\_collector\\_postgres.sql script \(p. 682\)](#)

## awsdms\_support\_collector\_postgres.sql script

Download the [awsdms\\_support\\_collector\\_postgres.sql](#) script.

This script collects information about your PostgreSQL database configuration. Remember to verify the checksum on the script. If the checksum verifies, review the SQL code in the script to comment out any of the code that you are uncomfortable running. After you are satisfied with the integrity and content of the script, you can run it.

### Note

You can run this script with psql client version 10 or later.

You can use the following procedures to run this script either from your database environment or from the command line. In either case, you can then upload your file to AWS Support later.

### To run this script and upload the results to your support case

1. Do one of the following:

- Run the script from your database environment using the following psql command line.

```
dbname=# \i awsdms_support_collector_postgres.sql
```

At the following prompt, enter the name of only one of the schemas that you want to migrate.

At the following prompt, enter the name of the user (*script\_user*) that you have defined to connect to the database.

- Run the following script directly from the command line. This option avoids any prompts prior to script execution.

```
psql -h database-hostname -p port -U script_user -d database-name -f awsdms_support_collector_postgres.sql
```

2. Review the output HTML file and remove any information that you are uncomfortable sharing. When the HTML is acceptable for you to share, upload the file to your AWS Support case. For more information on uploading this file, see [Working with diagnostic support scripts in AWS DMS \(p. 674\)](#).

# Working with the AWS DMS diagnostic support AMI

If you encounter a network-related issue when working with AWS DMS, your support engineer might need more information about your network configuration. We want to make sure that AWS Support gets as much of the required information as possible in the shortest possible time. Therefore, we developed a prebuilt Amazon EC2 AMI with diagnostic tools to test your AWS DMS networking environment.

The diagnostic tests installed on the Amazon machine image (AMI) include the following:

- Virtual Private Cloud (VPC)
- Network packet loss
- Network latency
- Maximum Transmission Unit (MTU) size

## Topics

- [Launch a new AWS DMS diagnostic Amazon EC2 instance \(p. 683\)](#)
- [Create an IAM role \(p. 683\)](#)
- [Run Diagnostic Tests \(p. 684\)](#)
- [Next Steps \(p. 687\)](#)
- [AMI IDs by region \(p. 687\)](#)

## Launch a new AWS DMS diagnostic Amazon EC2 instance

In this section, you launch a new Amazon EC2 instance. For information about how to launch an Amazon EC2 instance, see [Get started with Amazon EC2 Linux instances](#) tutorial in the [Amazon EC2 User Guide](#).

Launch an Amazon EC2 instance with the following settings:

- For **Application and OS Images (Amazon Machine Image)**, search for the **DMS-DIAG-AMI** AMI. If you are logged on to the console, you can search for the AMI with [this query](#). For the AMI ID of the AWS Diagnostic AMI in your region, see [AMI IDs by region \(p. 687\)](#) following.
- For **Instance type**, we recommend you choose **t2.micro**.
- For **Network Settings**, choose the same VPC that your replication instance uses.

After the instance is active, connect to the instance. For information about connecting to an Amazon EC2 Linux instance, see [Connect to your Linux instance](#).

## Create an IAM role

If you want to run the diagnostic tests on your replication instance using the minimum required permissions, create an IAM role that uses the following permissions policy:

```
{  
    "Version": "2012-10-17",  
    "Statement": [  
        {  
            "Sid": "VisualEditor0",  
            "Effect": "Allow",  
            "Action": "sts:AssumeRole",  
            "Resource": "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"  
        }  
    ]  
}
```

```
        "Action": [
            "dms:DescribeEndpoints",
            "dms:DescribeTableStatistics",
            "dms:DescribeReplicationInstances",
            "dms:DescribeReplicationTasks",
            "secretsmanager:GetSecretValue"
        ],
        "Resource": "*"
    }
]
```

Attach the role to a new IAM user. For information about creating IAM roles, policies, and users, see the following sections in the [IAM User Guide](#):

- [Getting Started with IAM](#)
- [Creating IAM Roles](#)
- [Creating IAM Policies](#)

## Run Diagnostic Tests

After you have created an Amazon EC2 instance and connected to it, do the following to run diagnostic tests on your replication instance.

1. Configure the AWS CLI:

```
$ aws configure
```

Provide the access credentials for the AWS user account you want to use to run the diagnostic tests. Provide the Region for your VPC and replication instance.

2. Display the available AWS DMS tasks in your Region. Replace the sample Region with your Region.

```
$ dms-report -r us-east-1 -l
```

This command displays the status of your tasks.

Task Name	Sample call to display task detail
AWS Region: us-east-2	
Total TaskARNs: 7	
[Num] [TaskName]	[Status]
0        dms:replication-task-ag-test-2-from-primary	failed
>>>> Sample call: dms-report -r us-east-2 -t arn:aws:dms:us-east-2:██████████:task:CD1D9AC0077E27BEC0400000000000000	
1        dms:replication-task-ag-test-2-from-secondary	failed
>>>> Sample call: dms-report -r us-east-2 -t arn:aws:dms:us-east-2:██████████:task:0.CD1D9AC0077E27BEC0400000000000004	
2        dms:replication-sync-pri	failed
>>>> Sample call: dms-report -r us-east-2 -t arn:aws:dms:us-east-2:██████████:task:X11223344556677889900000000000000	
3        dms:replication-test-cdc-12	failed
>>>> Sample call: dms-report -r us-east-2 -t arn:aws:dms:us-east-2:██████████:task:7.D0123456789012345678901234567890A	
4        dms:replication-test-sec-fullcdc	failed
>>>> Sample call: dms-report -r us-east-2 -t arn:aws:dms:us-east-2:██████████:task:K123456789012345678901234567890A	
5        dms:replication-logical-task	running
>>>> Sample call: dms-report -r us-east-2 -t arn:aws:dms:us-east-2:██████████:task:Q123456789012345678901234567890A	
6        mrp-poc-10-10	failed
>>>> Sample call: dms-report -r us-east-2 -t arn:aws:dms:us-east-2:██████████:task:mrp-poc-10-10	

3. Display task endpoints and settings. Replace <DMS-Task-ARN> with your task Amazon Resource Name (ARN).

```
$ dms-report -t <DMS-Task-ARN>
```

This command displays the endpoints and settings of your task.

```

#####
# AWS DMS Diagnostic
# Date: 07-13-2022
#
# aws region: us-east-2
#
#####
===== DMS Task Check =====

[DMS Replication ID]  ...
[DMS Replication Task] arn:aws:dms:us-east-2:...:task:0...
[Replication Instance] arn:aws:dms:us-east-2:...:rep:...

[RepInstanceClass] dms.t3.small
[RepAvailabilityZone] us-east-2c
[RepEngineVersion] 3.4.6
[RepPrivateIP] 172.30.2.230
[RepPublicIP] 0.0.0.0
[RepCreateDate] 2021-12-15T21:37:19.904000+00:00
[RepVPCId] vpc-08ba020355d8a952e

----- SOURCE -----
[Endpoints] arn:aws:dms:us-east-2:...:endpoint:4...
[ID] ...
[Server] ...
[TCP-Port] 5432
[IP-Test] 172.30.1.204 <> IP and TCP Port are GOOD
[Network-Check] Host
[Engine] aurora-postgresql
[DB] testdb
[Status] active,          TCP host port check

----- TARGET -----
[Endpoints] arn:aws:dms:us-east-2:...:endpoint:IX...
[ID] ...
[Server] ...
[TCP-Port] 5432
[IP-Test] 172.30.2.18 <> IP and TCP Port are Good
[Network-Check] Host
[Engine] aurora-postgresql
[DB] testdb
[Status] active,          Target Endpoint Info

----- Full Load -----
[Table] [Duration] [Rows] [lastUpdate] [Status]
public.test_orders 00:00:01 2 2022-06-21T08:48:12.172000+00:00 completed
public.test_details 00:00:01 8 2022-06-21T08:48:12.172000+00:00 completed
public.test_products 00:00:00 2 2022-06-21T08:48:12.172000+00:00 completed
public.delete_log 00:00:00 2 2022-06-21T08:48:12.172000+00:00 completed
public.address 00:00:01 1999 2022-06-21T08:48:12.172000+00:00 completed
public.huge_table 00:00:00 6 2022-06-21T08:48:12.172000+00:00 completed

```

4. Run diagnostic tests. Replace <DMS-Task-ARN> with your task ARN.

```
$ dms-report -t <DMS-Task-ARN> -n y
```

This command displays diagnostic data about your replication instance's VPC, network packet transmission, network latency, and network Maximum Transmission Unit (MTU) size.

```
#####
#
#      AWS DMS Diagnostic
#      Date: 07-13-2022
#
#
#      aws region: us-east-2
#
#
#####
===== DMS DIAG Info =====
Public IP: 3.235.100.12
Private IP: 172.30.0.240
Instance ID: i-04829b2beb8214602
Instance MAC: 02:58:04:b5:52:28
Instance Type: t2.micro
Instance Sec Group: Dms-sec-group
Instance AWS Region: us-east-2
Instance VPC Id: vpc-08ba020355d8a952e

===== Network Packet Check =====
1.) Check DMS EC2 MetaData service
>>>>Result: 10 packets transmitted, 10 packets received, 0% packet loss
    Looks good with no issue. <<<<

2.) Check Source endpoint (dm...-postgres-dev-instance-1.cucdvzaur7nk.us-east-2.rds.amazonaws.com:5432)
>>>>Result: 10 packets transmitted, 10 packets received, 0% packet loss
    Looks good with no issue. <<<<

3.) Check Target endpoint (rds...-postgres-dev-instance-1.cucdvzaur7nk.us-east-2.rds.amazonaws.com:5432)
>>>>Result: 10 packets transmitted, 10 packets received, 0% packet loss
    Looks good with no issue. <<<<
===== End network packet check =====

===== Network Latency Check =====
1.) Check DMS MetaData Service
>>>>Result: round-trip min/avg/max = 0.4/0.4/0.5 ms
    Looks good with no issue. <<<<

2.) Check Source endpoint (dm...-postgres-dev-instance-1.cucdvzaur7nk.us-east-2.rds.amazonaws.com:5432)
>>>>Result: round-trip min/avg/max = 1.0/1.1/1.2 ms
    Looks good with no issue. <<<<

3.) Check Target endpoint (rds...-postgres-dev-instance-1.cucdvzaur7nk.us-east-2.rds.amazonaws.com:5432)
>>>>Result: round-trip min/avg/max = 1.4/1.4/1.5 ms
    Looks good with no issue. <<<<
===== End network latency check =====

===== Network MTU Check =====
1.) Check DMS MetaData Service
>>>>Result: MTU setting looks good. Local MTU (9001) matches remote MTU (9001) <<<<

2.) Check Source endpoint (dm...-postgres-dev-instance-1.cucdvzaur7nk.us-east-2.rds.amazonaws.com:5432)
>>>>Result: MTU setting looks good. Local MTU (9001) matches remote MTU (9001) <<<<

3.) Check Target endpoint (rds...-postgres-dev-instance-1.cucdvzaur7nk.us-east-2.rds.amazonaws.com:5432)
>>>>Result: MTU setting looks good. Local MTU (9001) matches remote MTU (9001) <<<<
===== End network MTU check =====
```

Perform AMI Diag EC2 VPC Check

Perform Network Packet Test

Returns Test Results and Recommendation

Perform Network Latency Test

Perform Network Maximum Transmission Unit (MTU) Check

## Next Steps

The following sections describe troubleshooting information based on the results of the network diagnostic tests:

### VPC tests

This test verifies that the diagnostic Amazon EC2 instance is in the same VPC as the replication instance. If the diagnostic Amazon EC2 instance is not in the same VPC as your replication instance, terminate it and create it again in the correct VPC. You can't change the VPC of an Amazon EC2 instance after you create it.

### Network packet loss tests

This test sends 10 packets to the following endpoints and checks for packet loss:

- The AWS DMS Amazon EC2 metadata service on port 80
- The source endpoint
- The target endpoint

All packets should arrive successfully. If any packets are lost, consult with a network engineer to determine the problem and find a solution.

### Network latency tests

This test sends 10 packets to the same endpoints as the previous test, and checks for packet latency. All packets should have a latency of less than 100 milliseconds. If any packets have a latency greater than 100 milliseconds, consult with a network engineer to determine the problem and find a solution.

### Maximum Transmission Unit (MTU) size tests

This test detects the MTU size by using the Traceroute tool on the same endpoints as the previous test. All of the packets in the test should have the same MTU size. If any packets have a different MTU size, consult with a system specialist to determine the problem and find a solution.

## AMI IDs by region

The following table shows the AMI ID for the AMS Diagnostic AMI in your region:

Region	Image ID
US East (Ohio)	ami-0f9c6801e8cab9399
US West (Oregon)	ami-01ceeab889d966e16
US East (N. Virginia)	ami-0617285c8257e8f6e
US West (N. California)	ami-0983b989d0a31ed55
Canada (Central)	ami-0ba9380b30eee288e
Europe (Ireland)	ami-05959fe4fcc448f01
Europe (London)	ami-0de91dfa18b365c04

Region	Image ID
Europe (Paris)	ami-0844ecd26f2cc9073
Europe (Frankfurt)	ami-09bc69e6da9f3adf4
Europe (Stockholm)	ami-01da340b09bf5ab78
Europe (Milan)	ami-0dcfc51c17cf7727
Europe (Spain)	ami-0461c0b2e391c1280
Asia Pacific (Singapore)	ami-04150ffe00fe300d6
Asia Pacific (Sydney)	ami-049c1d1a9d72bafce
Asia Pacific (Jakarta)	ami-02189a4b8d3b2ee23
Asia Pacific (Hong Kong)	ami-0543ae06eff17cb07
Asia Pacific (Tokyo)	ami-07b91e8e25d7b7890
Asia Pacific (Seoul)	ami-0657ad8cb9b5cf71c
Asia Pacific (Osaka)	ami-01ccf3e974933afdc
Asia Pacific (Mumbai)	ami-0d9fa24bbb8225e60
Middle East (UAE)	ami-0e3dcf51dfe58929b
Middle East (Bahrain)	ami-0faad6f64101607f5
South America (São Paulo)	ami-097dd866777097d96

# Migrating large data stores using AWS Database Migration Service and AWS Snowball Edge

Larger data migrations can include many terabytes of information. This process can be cumbersome due to network bandwidth limits or just the sheer amount of data. AWS Database Migration Service (AWS DMS) can use [AWS Snowball Edge](#) and Amazon S3 to migrate large databases more quickly than by other methods.

AWS Snowball Edge is an AWS service that provides an Edge device that you can use to transfer data to the cloud at faster-than-network speeds. An Edge device is an AWS-owned appliance with large amounts of on-board storage. It uses 256-bit encryption and an industry-standard Trusted Platform Module (TPM) to ensure both security and full chain of custody for your data. AWS Snowball Edge offers many additional features; for more information, see [What is an AWS Snowball Edge?](#) in the *AWS Snowball Edge Developer Guide*.

Amazon S3 is an AWS storage and retrieval service. To store an object in Amazon S3, you upload the file you want to store to a bucket. When you upload a file, you can set permissions for the object and also for any metadata. For more information, see the [S3 documentation](#).

When you're using an Edge device, the data migration process has the following stages:

1. You use the AWS Schema Conversion Tool (AWS SCT) to extract the data locally and move it to an Edge device.
2. You ship the Edge device or devices back to AWS.
3. After AWS receives your shipment, the Edge device automatically loads its data into an Amazon S3 bucket.
4. AWS DMS takes the files and migrates the data to the target data store. If you are using change data capture (CDC), those updates are written to the Amazon S3 bucket and then applied to the target data store.

In the following sections, you can learn about using an Edge device to migrate relational databases with AWS SCT and AWS DMS. You can also use an Edge device and AWS SCT to migrate on-premises data warehouses to the AWS Cloud. For more information about data warehouse migrations, see [Migrating data from an on-premises data warehouse to Amazon Redshift](#) in the *AWS Schema Conversion Tool User Guide*.

## Topics

- [Overview of migrating large data stores using AWS DMS and AWS Snowball Edge \(p. 690\)](#)
- [Prerequisites for migrating large data stores using AWS DMS and AWS Snowball Edge \(p. 691\)](#)
- [Migration checklist \(p. 691\)](#)
- [Step-by-step procedures for migrating data using AWS DMS with AWS Snowball Edge \(p. 693\)](#)
- [Limitations when working with AWS Snowball Edge and AWS DMS \(p. 708\)](#)

## Note

You can't use an AWS Snowcone device to migrate data with AWS DMS.

# Overview of migrating large data stores using AWS DMS and AWS Snowball Edge

The process of using AWS DMS and AWS Snowball Edge incorporates both on-premises applications and AWS-managed services. We use the terms *local* and *remote* to distinguish these components.

Local components include the following:

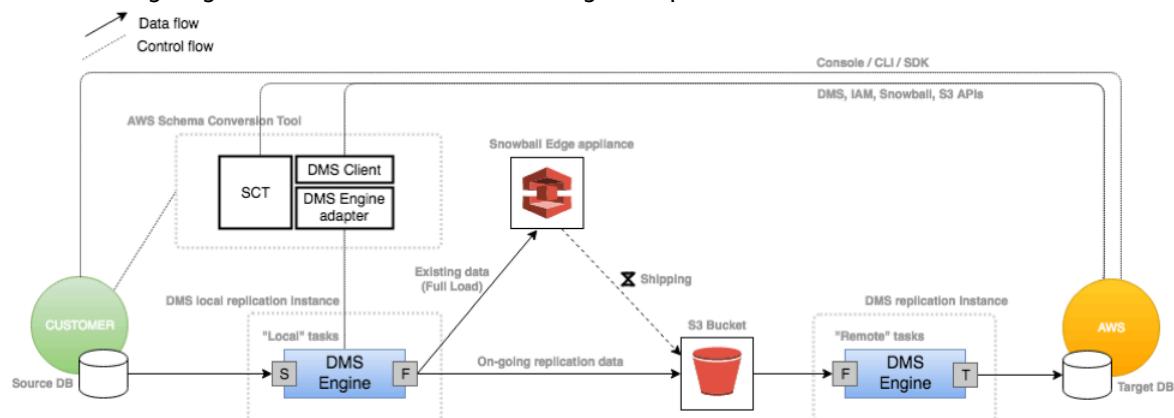
- AWS SCT
- AWS DMS Agent (a local version of AWS DMS that works on-premises)
- AWS Snowball Edge devices

Remote components include the following:

- Amazon S3
- AWS DMS

In the following sections, you can find a step-by-step guide to configuring, installing, and managing an AWS DMS migration using an Edge device or devices.

The following diagram shows an overview of the migration process.



The migration involves a local task, where you move data to an Edge device using the DMS Agent. After an Edge device is loaded, you return it to AWS. If you have multiple Edge devices, you can return them at the same time, or in sequence. When AWS receives an Edge device, a remote task using AWS DMS loads the data to the target data store on AWS.

To migrate from a local data store to an AWS data store, you follow these steps:

1. Use the [AWS Snow Family Management Console](#) to create a new job for importing data to S3 with a **Snowball Edge Storage Optimized** device. The job involves requesting the device be sent to your address.
2. Set up AWS SCT on a local machine that can access AWS. Install the Snowball Edge client tool on a different local machine.

For more information about AWS SCT, see [Installing AWS SCT](#) in the *AWS Schema Conversion Tool User Guide*. For more information about the Snowball Edge client tool, see [Downloading and Installing the Snowball Edge client](#) in the *AWS Snowball Edge Developer Guide*.

3. When the Edge device arrives, power it on, connect to it, then unlock it with the client tool. For step-by-step information, see [Getting started with AWS Snowball edge](#) in the *AWS Snowball Edge Developer Guide*.
4. Install the Open Database Connectivity (ODBC) drivers for your data sources. Put these on the machine with the Edge client tool.
5. Install and configure the AWS DMS Agent host on the machine with the Edge client tool. For more information, see [Step 5: Install the AWS DMS Agent \(p. 699\)](#).

The AWS DMS Agent must have connectivity to the source database, AWS SCT, AWS, and AWS Snowball Edge. The AWS DMS Agent is supported on the following Linux platforms only:

- Red Hat Enterprise Linux versions 6.2 through 6.8, 7.0, and 7.1 (64-bit)
- SUSE Linux version 12 (64-bit)

Although the AWS DMS Agent comes in the AWS SCT installation package, it's better that they aren't located in the same place. We recommend that you install the DMS Agent on a different machine—not the machine you installed AWS SCT on.

6. Create a new project in AWS SCT as described in [Step-by-step procedures for migrating data using AWS DMS with AWS Snowball Edge \(p. 693\)](#).
  7. Configure AWS SCT to use the AWS Snowball Edge device as described in [Step-by-step procedures for migrating data using AWS DMS with AWS Snowball Edge \(p. 693\)](#).
  8. Register the AWS DMS Agent with AWS SCT as described in [Step-by-step procedures for migrating data using AWS DMS with AWS Snowball Edge \(p. 693\)](#).
  9. Create a local and AWS DMS task in SCT as described in [Step-by-step procedures for migrating data using AWS DMS with AWS Snowball Edge \(p. 693\)](#).
10. Run and monitor the task in SCT as described in [Step-by-step procedures for migrating data using AWS DMS with AWS Snowball Edge \(p. 693\)](#).

## Prerequisites for migrating large data stores using AWS DMS and AWS Snowball Edge

Before you start the migration process, you need the following prerequisites:

- You are familiar with the basic operation of AWS SCT.
- You have or can create the S3 bucket or buckets to use for the migration.
- You have an AWS DMS replication instance in the same AWS Region as the S3 bucket.
- You are comfortable with using the AWS Command Line Interface (AWS CLI).
- You are familiar with the [AWS Snowball Edge developer guide](#).

## Migration checklist

To make things easier during migration, you can use the following checklist to create a list of the items that you need during the migration.

DMS Migration Checklist

This checklist is for my schemas named:

The database engine that my schemas reside on is:

AWS Region for the migration:

Name of migration job that you created in the AWS Snowball Management Console:

S3 bucket (and folder) for this job:

IAM role that has access to the S3 Bucket and the target database on AWS:

Path to the installation directory of AWS SCT (needed for a future step):

Name/IP of Machine #1 (SCT):

Name/IP of Machine #2 (Connectivity):

IP address of your Snowball Edge:

Port for the Snowball Edge:

Unlock code for the Snowball Edge device:

Path to the manifest file:

Output of the command snowballEdge get-secret-access-key:

    AWS access key ID:

    AWS secret access Key:

Confirm ODBC drivers is installed on Machine #2 (Connectivity):

Confirm DMS Agent is installed on Machine #2 (Connectivity):

Confirm DMS Agent is running two processes:

DMS Agent password:

DMS Agent port number:

Confirm that your firewall allows connectivity:

Name of SCT project:

Confirm that DMS Agent is registered with SCT:

New agent or service profile name that you provided:

Confirm local and DMS task exists:

Task name that you provided:

Confirm:

DMS Agent connects to the following:

- The source database
- The staging S3 bucket
- The Edge device

DMS task connects to the following:

- The staging S3 bucket
- The target database on AWS

-----

Confirm the following:

- Stopped Edge client
- Powered off Edge device
- Returned Edge device to AWS

## Step-by-step procedures for migrating data using AWS DMS with AWS Snowball Edge

In the following sections, you can find detailed information on the migration steps.

### Topics

- Step 1: Create a AWS Snowball Edge job (p. 693)
- Step 2: Download and install the AWS Schema Conversion Tool (AWS SCT) (p. 693)
- Step 3: Unlock the AWS Snowball Edge device (p. 694)
- Step 4: Configure the AWS DMS agent host with ODBC drivers (p. 696)
- Step 5: Install the AWS DMS Agent (p. 699)
- Step 6: Create a new AWS SCT project (p. 701)
- Step 7: Configure AWS SCT to use AWS Snowball Edge (p. 701)
- Step 8: Register the AWS DMS Agent in AWS SCT (p. 705)
- Step 9: Create a local and AWS DMS task (p. 705)
- Step 10: Run and monitor the task in SCT (p. 706)

## Step 1: Create a AWS Snowball Edge job

Follow the steps outlined in the section [Getting started with an AWS Snowball Edge device](#) in the *AWS Snowball Edge Developer Guide*. Open the AWS Snow Family Management Console, and create a new job for **Import into Amazon S3**.

Be sure to request a AWS Snowball Edge device (**AWS Snowball Edge Storage Optimized**), because regular Snowball devices are not supported for AWS DMS. Follow the screen prompts for remaining settings. You have a chance to review your settings before you create the job.

## Step 2: Download and install the AWS Schema Conversion Tool (AWS SCT)

You need two local machines to run this process, in addition to the Edge device.

Download the AWS Schema Conversion Tool application and install it on a local machine that can access AWS. For instructions, which include information on compatible operating systems, see [Installing, verifying, and updating AWS SCT](#) in the *AWS Schema Conversion Tool User Guide*.

On a different machine, where you plan to install the AWS DMS Agent, download and install the AWS Snowball Edge client from [AWS Snowball Edge resources](#).

After you finish this step, you should have two machines:

- Machine #1 (SCT), with AWS SCT installed
- Machine #2 (Connectivity), with the Edge client, where you plan to install the AWS DMS Agent and the ODBC drivers for the databases you are migrating

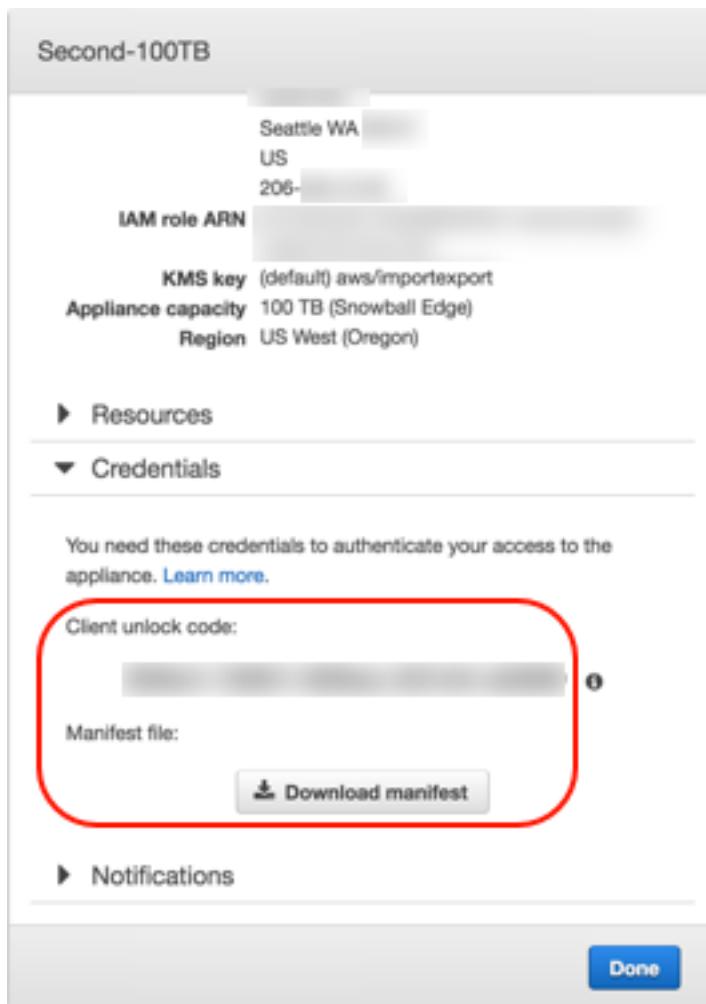
## Step 3: Unlock the AWS Snowball Edge device

When the Edge device arrives, prepare it for use.

Follow the steps outlined in the section [Getting started with an AWS Snowball Edge device](#) in the *AWS Snowball Edge Developer Guide*.

You can also check out the [AWS Snowball Edge getting started marketing page](#) for more resources.

Power the device on, [connect it to your local network](#), record the IP address of the Edge device, and [obtain the unlock code and manifest file](#) from the AWS Snowball Edge console. In the console, choose your job, choose **View job details**, and then **Credentials**. Save both the client unlock code and the manifest file.



On the Edge device screen, get the IP of the Edge device from the **Connection** tab. Then unlock the device by using the snowballEdge unlock command with the IP and the credentials information. The following example shows the sample syntax for this command.

```
snowballEdge unlock -i IP_Address -m Local_path_to_manifest_file -  
u 29_character_unlock_code
```

Following is an example command.

```
snowballEdge unlock \  
-i 192.0.2.0 \  
-m /Downloads/JID2EXAMPLE-0c40-49a7-9f53-916aEXAMPLE81-manifest.bin \  
-u 12345-abcd-e12345-ABCDE-12345
```

Finally, [retrieve the AWS Snowball Edge access key and secret key](#) from the device using the Edge client. The following shows example input and output for the command to get the access key.

#### Example input

```
snowballEdge list-access-keys \  
--endpoint https://192.0.2.0 \  
--manifest-file Path_to_manifest_file \  
--
```

```
--unlock-code 12345-abcd-e12345-ABCDE-12345
```

#### Example output

```
{  
    "AccessKeyId" : [ "AKIAIOSFODNN7EXAMPLE" ]  
}
```

The following shows example input and output for the command to get the secret key.

#### Example input

```
snowballEdge get-secret-access-key \  
    --access-key-id AKIAIOSFODNN7EXAMPLE \  
    --endpoint https://192.0.2.0 \  
    --manifest-file /Downloads/JID2EXAMPLE-0c40-49a7-9f53-916aEXAMPLE81-manifest.bin \  
    --unlock-code 12345-abcd-e12345-ABCDE-12345
```

#### Example output

```
[snowballEdge]  
aws_access_key_id = AKIAIOSFODNN7EXAMPLE  
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

When the AWS Snowball Edge is ready to use, you can interact with it directly by using the AWS CLI or S3 SDK Adapter for Snowball. This adapter also works with the Edge device.

## Step 4: Configure the AWS DMS agent host with ODBC drivers

Using Machine #2 (Connectivity) from step 2, where the Edge client is already installed, install the necessary ODBC drivers. These drivers are necessary to connect to your source database. The required driver varies by database engine. In the following sections, you can find information for each database engine.

#### Topics

- [Oracle \(p. 696\)](#)
- [Microsoft SQL Server \(p. 697\)](#)
- [ASE SAP Sybase \(p. 697\)](#)
- [MySQL \(p. 697\)](#)
- [PostgreSQL \(p. 698\)](#)
- [DB2 LUW \(p. 698\)](#)

## Oracle

Install Oracle Instant Client for Linux (x86-64) version 11.2.0.3.0 or later.

In addition, if not already included in your system, you need to create a symbolic link in the `$ORACLE_HOME\lib` directory. This link should be called `libclntsh.so`, and should point to a specific version of this file. For example, on an Oracle 12c client you use the following.

```
lrwxrwxrwx 1 oracle oracle 63 Oct 2 14:16 libclntsh.so ->/u01/app/oracle/home/lib/  
libclntsh.so.12.1
```

The LD\_LIBRARY\_PATH environment variable should be appended with the Oracle lib directory and added to the site\_arep\_login.sh script under the **bin** folder of the **agent** installation. Add this script if it doesn't exist.

```
vi /opt/amazon/aws-schema-conversion-tool-dms-agent/bin/site_arep_login.sh

export ORACLE_HOME=/usr/lib/oracle/12.2/client64;
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
```

## Microsoft SQL Server

Install the Microsoft ODBC Driver.

For ODBC 17 drivers, update the site\_arep\_login.sh script located under the bin folder of the agent installation with the following code.

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/microsoft/msodbcsql17/lib64/
```

For ODBC drivers earlier than ODBC 17, update the site\_arep\_login.sh script with the following code.

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/microsoft/msodbcsql/lib64/
```

## ASE SAP Sybase

The SAP Sybase ASE ODBC 64-bit client should be installed.

If the installation directory is /opt/sap, update the site\_arep\_login.sh script under the bin folder of the agent installation with the following.

```
export SYBASE_HOME=/opt/sap
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$SYBASE_HOME/DataAccess64/ODBC/lib:$SYBASE_HOME/
>DataAccess/ODBC/lib:$SYBASE_HOME/OCS-16_0/lib:$SYBASE_HOME/OCS-16_0/lib3p64:$SYBASE_HOME/
>OCS-16_0/lib3p
```

The /etc/odbcinst.ini file should include the following entries.

```
[Sybase]
Driver=/opt/sap/DataAccess64/ODBC/lib/libsybdrvodb.so
Description=Sybase ODBC driver
```

## MySQL

Install MySQL Connector/ODBC for Linux, version 5.2.6 or later.

Make sure that the /etc/odbcinst.ini file contains an entry for MySQL, as shown in the following example.

```
[MySQL ODBC 5.2.6 Unicode Driver]
Driver = /usr/lib64/libmyodbc5w.so
UsageCount = 1
```

The /etc/odbcinst.ini file contains information about ODBC drivers available to users and can be edited by hand. If necessary, you can use the odbcinst -j command to look for the /etc/odbcinst.ini file, as shown in the following example.

```
$ odbcinst -j  
DRIVERS.....: /etc/odbcinst.ini
```

## PostgreSQL

Install postgresql94-9.4.4-1PGDG.*OS Version*.x86\_64.rpm. This package contains the psql executable. For example, postgresql94-9.4.4-1PGDG.rhel7.x86\_64.rpm is the package required for Red Hat 7.

Install the ODBC driver postgresql94-odbc-09.03.0400-1PGDG.*OS version*.x86\_64 or above for Linux, where *OS version* is the OS of the agent machine. For example, postgresql94-odbc-09.03.0400-1PGDG.rhel7.x86\_64 is the client required for Red Hat 7.

Make sure that the /etc/odbcinst.ini file contains an entry for PostgreSQL, as shown in the following example.

```
[PostgreSQL]  
Description = PostgreSQL ODBC driver  
Driver = /usr/pgsql-9.4/lib/pgsqlodbc.so  
Setup = /usr/pgsql-9.4/lib/pgsqlodbcw.so  
Debug = 0  
CommLog = 1  
UsageCount = 2
```

## DB2 LUW

To install the IBM DB2 for LUW driver on the agent machine, do the following:

- Install the appropriate version of IBM Data Server Client on the AWS DMS Agent machine:
  - For IBM DB2 for LUW 10.5, install client version 10.5.
  - For IBM DB2 for LUW 11.1 and 11.5, install client version 11.1.
- Download and unzip the DB2 Server Software. For example, v10.5fp9\_linuxx64\_server\_t.tar.gz is the package for IBM DB2 for LUW version 10.5.
- Install the DB2 client using the db2\_install command. When prompted, choose to install **CLIENT**.
- On the AWS DMS Agent machine, create a DB2 instance by running the following commands:

```
adduser <db2_instance_name>  
/opt/ibm/db2/V10.5/instance/db2icrt <db2_instance_name>
```

*Example:*

```
adduser db2inst1  
/opt/ibm/db2/V10.5/instance/db2icrt db2inst1
```

- Add the DB2 driver location to the Linux library path. To do this, add the following lines to the site\_arep\_login.sh file:

```
export DB2_HOME=<db2_driver_path>  
export DB2INSTANCE=<db2_instance_name>  
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$DB2_HOME/lib64:$DB2_HOME/lib64/gskit_db2
```

*Example:*

```
export DB2_HOME=/opt/ibm/db2/V10.5
```

```
export DB2INSTANCE=db2inst1
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$DB2_HOME/lib64:$DB2_HOME/lib64/gskit_db2
```

- Make sure that the `/etc/odbcinst.ini` file contains an entry for IBM DB2 LUW, as shown in the following example:

```
[IBM DB2 ODBC DRIVER]
Driver = /opt/ibm/db2/V10.5/lib64/libdb2o.so
fileusage=1
dontdlclose=1
```

- Restart the AWS DMS Agent by running the following command in the `/opt/amazon/aws-schema-conversion-tool-dms-agent/bin` directory:

```
./aws-schema-conversion-tool-dms-agent restart
```

## Step 5: Install the AWS DMS Agent

Using Machine #2 (Connectivity) from step 2, where the Edge client and the ODBC drivers are already installed, install and configure the AWS DMS Agent. The AWS DMS Agent is provided as part of the [AWS SCT installation package](#), described in the [AWS Schema Conversion Tool User Guide](#).

After you finish this step, you should have two local machines prepared:

- Machine #1 (SCT) with AWS SCT installed
- Machine #2 (Connectivity) with the Edge client, the ODBC drivers, and the DMS Agent installed

### To install the AWS DMS Agent

1. In the AWS SCT installation directory, locate the RPM file called `aws-schema-conversion-tool-dms-agent-X.X.X-XX.x86_64.rpm`.

Copy it to Machine #2 (Connectivity), the AWS DMS Agent machine. SCT and the DMS Agent should be installed on separate machines. AWS DMS Agent should be located on the same machine as the Edge client and the ODBC drivers.

2. On Machine #2 (Connectivity), run the following command to install the DMS Agent. To simplify permissions, run this command as the `root` user.

```
sudo rpm -i aws-schema-conversion-tool-dms-agent-X.X.X-XX.x86_64.rpm
```

This command uses the default installation location of `/opt/amazon/aws-schema-conversion-tool-dms-agent`. To install the DMS Agent to a different location, use the following option.

```
sudo rpm --prefix installation_directory -i aws-schema-conversion-tool-dms-agent-X.X.X-XX.x86_64.rpm
```

3. To verify that the AWS DMS Agent is running, use the following command.

```
ps -ef | grep repctl
```

The output of this command should show two processes running.

To configure the AWS DMS Agent, you must provide a password and port number. You use the password later to register the AWS DMS Agent with AWS SCT, so keep it handy. Pick an unused port

number for the AWS DMS Agent to listen on for AWS SCT connections. You might have to configure your firewall to allow connectivity.

Now configure the AWS DMS Agent using the `configure.sh` script.

```
sudo /opt/amazon/aws-schema-conversion-tool-dms-agent/bin/configure.sh
```

The following prompt appears. Enter the password. When prompted, enter the password again to confirm it.

```
Configure the AWS Schema Conversion Tool DMS Agent server
Note: you will use these parameters when configuring agent in AWS Schema Conversion
Tool

Please provide password for the server
Use minimum 8 and up to 20 alphanumeric characters with at least one digit and one
capital case character

Password:
```

The output is as follows. Provide a port number.

```
chown: missing operand after 'amazon:amazon'
Try 'chown --help' for more information.
/opt/amazon/aws-schema-conversion-tool-dms-agent/bin/repctl:
    /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libcom_err.so.3: no version
    information available (required by
        /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libgssapi_krb5.so.2)
/opt/amazon/aws-schema-conversion-tool-dms-agent/bin/repctl:
    /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libcom_err.so.3: no version
    information available (required by
        /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libkrb5.so.3)
[setserverpassword command] Succeeded

Please provide port number the server will listen on (default is 3554)
Note: you will have to configure your firewall rules accordingly
Port:
```

The output is as follows, confirming that the service is started.

```
Starting service...
/opt/amazon/aws-schema-conversion-tool-dms-agent/bin/repctl:
    /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libcom_err.so.3: no version
    information available (required by
        /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libgssapi_krb5.so.2)
/opt/amazon/aws-schema-conversion-tool-dms-agent/bin/repctl:
    /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libcom_err.so.3: no version
    information available (required by
        /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libkrb5.so.3)
AWS Schema Conversion Tool DMS Agent was sent a stop signal
AWS Schema Conversion Tool DMS Agent is no longer running
[service command] Succeeded
/opt/amazon/aws-schema-conversion-tool-dms-agent/bin/repctl:
    /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libcom_err.so.3: no version
    information available (required by
        /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libgssapi_krb5.so.2)
/opt/amazon/aws-schema-conversion-tool-dms-agent/bin/repctl:
    /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libcom_err.so.3: no version
    information available (required by
        /opt/amazon/aws-schema-conversion-tool-dms-agent/lib/libkrb5.so.3)
```

AWS Schema Conversion Tool DMS Agent was started as PID 1608

We recommend that you install the [AWS Command Line Interface](#) (AWS CLI). Using the AWS CLI, you can interrogate the AWS Snowball Edge to see the data files written to the device. You use the AWS credentials retrieved from the Edge to access the Edge device. For example, you might run the following command.

```
aws s3 ls --profile SnowballEdge --endpoint https://192.0.2.0:8080 bucket-name --recursive
```

This command produces the following output.

```
2018-08-20 10:55:31 53074692 streams/load00000001000573E166ACF4C0/00000001.fcd.gz
2018-08-20 11:14:37 53059667 streams/load00000001000573E166ACF4C0/00000002.fcd.gz
2018-08-20 11:31:42 53079181 streams/load00000001000573E166ACF4C0/00000003.fcd.gz
```

To stop the AWS DMS Agent, run the following command in the /opt/amazon/aws-schema-conversion-tool-dms-agent/bin directory.

```
./aws-schema-conversion-tool-dms-agent stop
```

To start the AWS DMS Agent, run the following command in the /opt/amazon/aws-schema-conversion-tool-dms-agent/bin directory.

```
./aws-schema-conversion-tool-dms-agent start
```

## Step 6: Create a new AWS SCT project

Next, you create a new AWS SCT project that specifies the source and target databases. For more information, see [Creating an AWS Schema Conversion Tool project](#) in the *AWS Schema Conversion Tool User Guide*.

### To create a new project in AWS SCT

1. Start AWS SCT, and choose **File** then **New Project**. The **New Project** dialog box appears.
2. Enter a name for your project, which is stored locally on your computer.
3. Enter the location for your local project file.
4. Choose **OK** to create your AWS SCT project.
5. Choose **Add source** to add a new source database to your AWS SCT project. You can add multiple source databases to your AWS SCT project.
6. Choose **Add target** to add a new target platform in your AWS SCT project. You can add multiple target platforms to your AWS SCT project.
7. Choose the source database schema in the left panel.
8. In the right panel, specify the target database platform for the selected source schema.
9. Choose **Create mapping**. This button becomes active after you choose the source database schema and the target database platform.

## Step 7: Configure AWS SCT to use AWS Snowball Edge

Your AWS SCT service profile must be updated to use the AWS DMS agent, which is a local AWS DMS that works on-premises.

### To update the AWS SCT profile to work with the AWS DMS agent

1. Start AWS SCT.
2. Choose **Settings**, **Global settings**, **AWS service profiles**.
3. Choose **Add a new AWS service profile**.
4. Add the following profile information.

Parameter	Action
<b>Profile Name</b>	Enter a name for your project, which is stored locally on your computer.
<b>AWS access key</b>	Enter the AWS access key for the AWS account and AWS Region that you plan to use for the migration. The supplied credentials must have permissions to access the AWS Snowball Edge job in AWS.
<b>AWS secret key</b>	Enter the AWS secret key for the AWS account and AWS Region that you plan to use for the migration.
<b>Region</b>	Choose the AWS Region for the account you are using. Your DMS replication instance, S3 bucket, and target data store must be in this AWS Region.
<b>Amazon S3 bucket folder</b>	Enter a name for S3 bucket that you were assigned when you created the AWS Snowball Edge job.

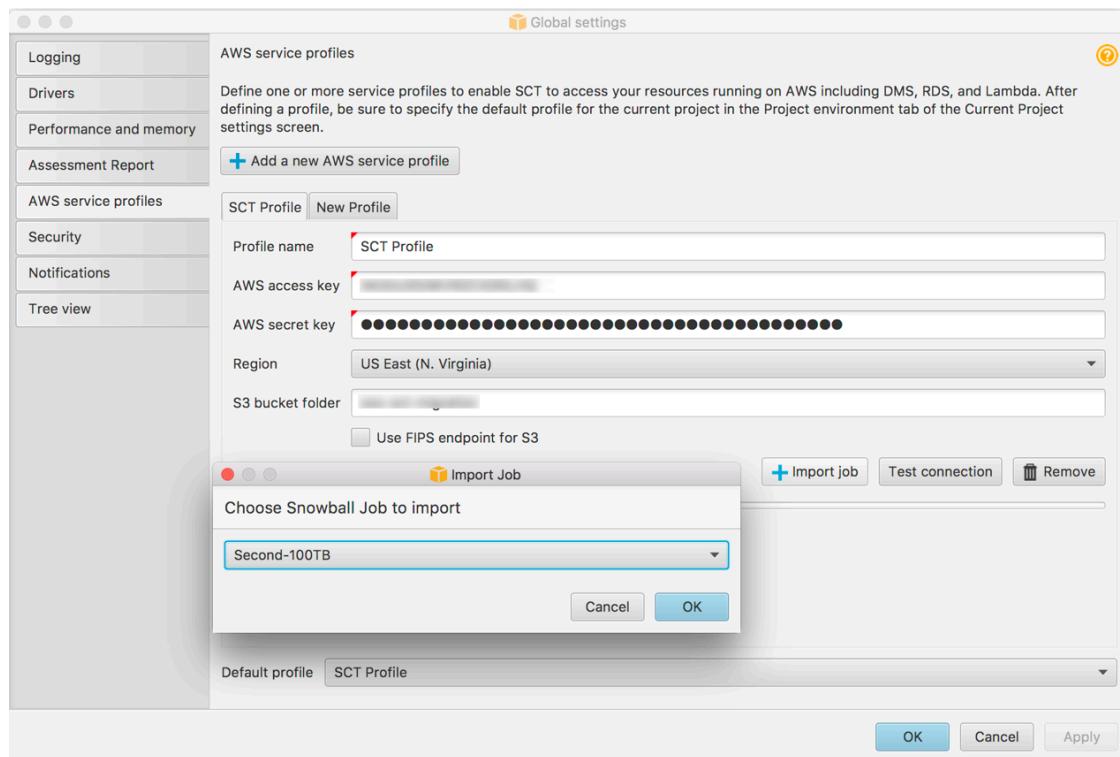
5. After you have entered the information, choose **Test connection** to verify that AWS SCT can connect to the Amazon S3 bucket.

The **OLTP Local & AWS DMS Data Migration** section in the pop-up window should show all entries with a status of **Pass**. If the test fails, the failure is probably because the account you are using is missing privileges to access the Amazon S3 bucket. The example following lists privileges.

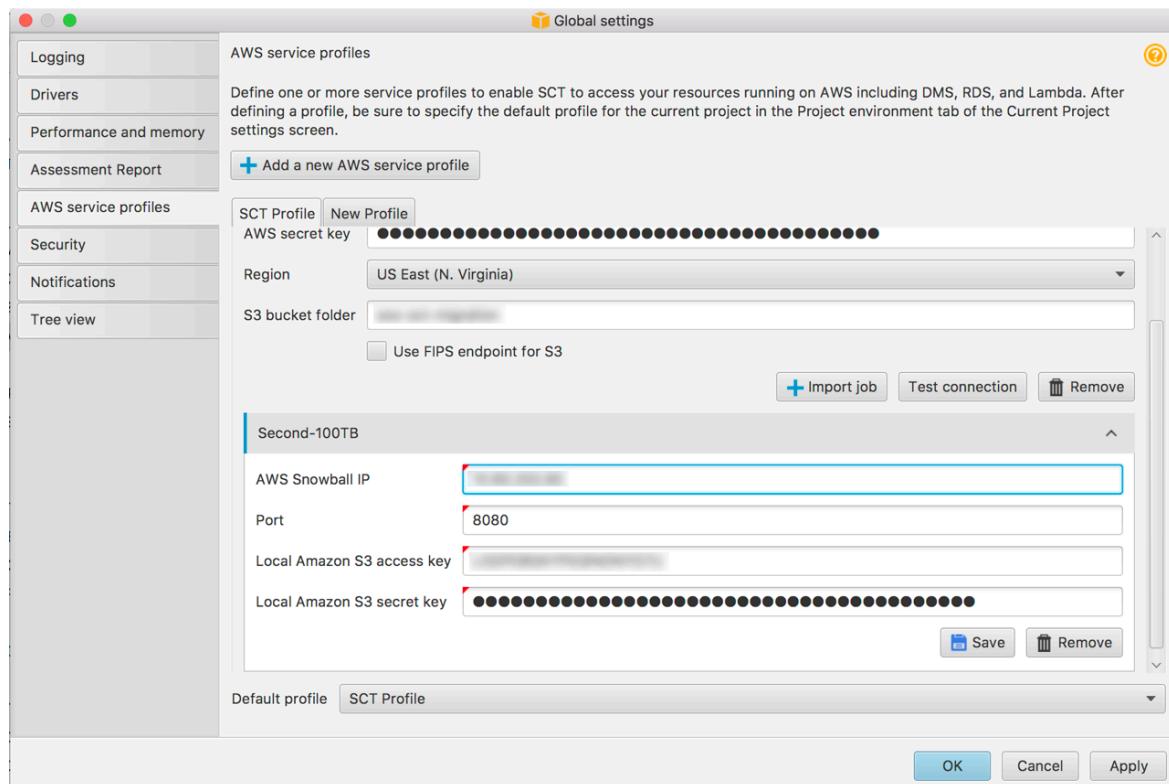
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1607372356645",
            "Action": [
                "cloudwatch:Get*",
                "cloudwatch>List*",
                "cloudwatch:PutMetricData",
                "dms:*",
                "dynamodb:BatchGetItem",
                "dynamodb:BatchWriteItem",
                "dynamodb>CreateTable",
                "dynamodb>DeleteItem",
                "dynamodb>DeleteTable",
                "dynamodb>DescribeLimits",
                "dynamodb>DescribeReservedCapacity",
                "dynamodb>DescribeReservedCapacityOfferings",
                "dynamodb>DescribeTable",
                "dynamodb>DescribeTimeToLive",
                "dynamodb>.GetItem",
                "dynamodb>ListTables",
                "dynamodb>ListTagsOfResource",
                "dynamodb>PutItem",
                "dynamodb>Query",
                "dynamodb>Scan",
                "lambda:InvokeFunction"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

```
"dynamodb:UpdateItem",
"dynamodb:UpdateTable",
"ec2>CreateNetworkInterface",
"ec2>CreateTags",
"ec2>DeleteNetworkInterface",
"ec2>DeleteTags",
"ec2>DescribeAvailabilityZones",
"ec2>DescribeInternetGateways",
"ec2>DescribeNetworkInterfaces",
"ec2>DescribeRouteTables",
"ec2>DescribeSecurityGroups",
"ec2>DescribeSubnets",
"ec2>DescribeVpcAttribute",
"ec2>DescribeVpcEndpoints",
"ec2>DescribeVpcs",
"ec2>ModifyNetworkInterfaceAttribute",
"glue:*",
"iam>AttachRolePolicy",
"iam>CreateRole",
"iam>GetRole",
"iam> GetUserStatus",
"iam> GetUserStatus",
"iam>ListRolePolicies",
"iam>PassRole",
"iam> GetUser",
"iam>PutRolePolicy",
"iam>SimulatePrincipalPolicy",
"kms>DescribeKey",
"kms>ListAliases",
"lambda>CreateFunction",
"lambda>ListFunctions",
"logs>AssociateKmsKey",
"logs>CreateLogGroup",
"logs>CreateLogStream",
"logs>DescribeLogGroups",
"logs>DescribeLogStreams",
"logs>FilterLogEvents",
"logs>GetLogEvents",
"logs>PutLogEvents",
"s3>CreateBucket",
"s3>DeleteObject",
"s3>GetBucketAcl",
"s3>GetBucketLocation",
"s3>GetObject",
"s3>ListAllMyBuckets",
"s3>ListBucket",
"s3>PutObject",
"snowball>DescribeJob",
"snowball>GetJobManifest",
"snowball>GetJobUnlockCode",
"snowball>ListJobs"
],
"Effect": "Allow",
"Resource": "*"
}
]
}
```

6. If the test passes, choose **OK** and then **OK** again to close the window and dialog box.
7. Choose **Import job**, choose the AWS Snowball Edge job from the list, and then choose **OK**.



Now configure AWS SCT to use the AWS Snowball Edge. Enter the IP address of the Snowball Edge, the listening port on the device (the default is 8080), and the AWS Snowball Edge access keys and secret keys you retrieved earlier. Choose **OK** to save your changes.



## Step 8: Register the AWS DMS Agent in AWS SCT

Next, you register the AWS DMS Agent in AWS SCT. SCT then tries to connect to the agent, showing status. When the agent is available, the status turns to active.

### To register the AWS DMS Agent

1. Start AWS SCT, choose **View**, and then choose **Database Migration View (Local & DMS)**.
2. Start AWS SCT, choose **View**, and then choose **Data migration view (other)**. Then choose **Register**.
3. For **Description**, enter a name for your data extraction agent.
4. For **Host name**, enter the IP address of the computer where you run your data extraction agent.
5. For **Port**, enter the listening port that you configured.
6. Choose **Register**.

## Step 9: Create a local and AWS DMS task

Next, you create the task that is the end-to-end migration task. This task includes two subtasks:

- The local subtask – This task migrates data from the source database to the AWS Snowball Edge appliance.
- The AWS DMS subtask – This task moves the data from the appliance into an Amazon S3 bucket and migrates it to the target database.

**Note**

We recommend that you test your migration before you use the AWS Snowball Edge device. You can do this by setting up a task to send data, such as a single table, to an Amazon S3 bucket instead of to the AWS Snowball Edge device.

**To create the end-to-end migration task**

1. Start AWS SCT, choose **View**, and then choose **Data migration view (other)**.
2. In the left panel that displays the schema from your source database, choose a schema object to migrate. Open the context (right-click) menu for the object, and then choose **Create local task**.
3. For **Task name**, enter a descriptive name for your data migration task.
4. For **Migration mode**, choose **Extract, upload, and copy**.
5. Choose **Amazon S3 settings**.
6. Select **Use Snowball**.
7. Enter folders and subfolders in your Amazon S3 bucket where the data extraction agent can store data.
8. Choose **Create** to create the task.

## Step 10: Run and monitor the task in SCT

You can start the migration task when connections to all endpoints are successful, including the following:

- AWS DMS Agent connections to these:
  - The source database
  - The staging Amazon S3 bucket
  - The Edge device
- AWS DMS Task connections to these:
  - The source database
  - The staging Amazon S3 bucket
  - The target database on AWS

If all connections are functioning properly, your SCT console resembles the following screen shot, and you are ready to begin.

Task Name	Status	Complete %	Elapsed Time	Tables Loaded	Tables Loading	Tables...
abhinavmigration		0%		0	0	0

**Task details**

**Version:**

Round Trip Latency (ms): 0

**Source Endpoint:**

- Name: [redacted]
- Status: Successful
- Connection message: To sqldemo: 'Connection successful' / To S3/Snowball: 'Connection successful'

**Target Endpoint:**

- Name: [redacted]
- Status: Successful
- Connection message: To S3: 'Connection successful' / To aurora-postgresql: 'Connection successful'

Use the following procedure to start the migration.

### To start the migration task

1. Choose the migration task, then choose **Start**.

Task Name	Status	Complete %	Elapsed Time	Tables Loaded	Tables Loading	Tables...
abhinavmigration		0%		0	0	0
abhinavmigration-Local-generated		0%		0	0	0
abhinavmigration-DMS-generated		0%		0	0	0
snowball-migration		0%		0	0	0
snowball-migration-Local-generated		0%		0	0	0
snowball-migration-DMS-generated		0%		0	0	0

**Task details**

ID: f26b735c2dbc462283d5a89f2d5a7961

Task name: abhinavmigration

2. To monitor the AWS DMS Agent, choose **Show Log**. The log details include the agent server (**Agent Log**) and local running task (**Task Log**) logs. The endpoint connectivity is done by the server. Because the local task isn't running, it has no task logs. Connection issues are listed under the **Agent Log** tab.

The screenshot shows two windows side-by-side. The top window is titled 'Log details' and contains a table of log entries:

Date	Component	Verbosity	Text
2017-12-01 09:52:00	INFRASTRUCTURE	Info	Allocated 20 ODBC connect...
2017-12-01 09:52:01	SOURCE_CAPTURE	Info	Source endpoint 'Mysql' is u...
2017-12-01 09:52:01	TARGET_LOAD	Info	Start loading table 'classicm...
2017-12-01 09:52:01	TARGET_LOAD	Info	Start loading table 'classicm...

The bottom window is titled 'AWS SCT' and shows a 'Tasks' table:

Task Name	Status	Complete %	Elapsed Time	Tables
DemoTask	Running	50%		0
DemoTask-Local	Completed	100%	0h:00m:11s	10
DemoTask-DMS	Running	0%	0h:01m:00s	0

After the source tables have been loaded onto the AWS Snowball Edge appliance, AWS SCT updates the status of the task to show it is 50 percent complete. This is because the other half of the task involves AWS DMS taking the data from Amazon S3 to the target data store.

3. Verify that the status of the migration task is 50 percent. You can use the AWS Snowball Edge console or AWS SCT to check the status of the device.
4. Follow the steps outlined in the AWS Snowball Edge documentation, starting with the section named [Stop the Snowball client, and power off the Snowball Edge](#). These steps include the following:
  - Stopping the Snowball Edge client
  - Powering off the Edge device
  - Returning the Edge device to AWS
5. Finishing the migration after the device returns to AWS involves waiting for the remote task to complete.

When the AWS Snowball Edge appliance arrives at AWS, the remote (DMS) task starts to run. If the migration type you chose was **Migrate existing data**, the status for the AWS DMS task show 100 percent complete after the data has been transferred from Amazon S3 to the target data store.

If you set the task mode to include ongoing replication, then after the full load is complete the task continues to run while AWS DMS applies ongoing changes.

## Limitations when working with AWS Snowball Edge and AWS DMS

There are some limitations you should be aware of when working with AWS Snowball Edge:

- Every AWS SCT task creates two endpoint connections on AWS DMS. If you create multiple tasks, you can reach a resource limit for the number of endpoints that you can create.
- A schema is the minimum task scope when using AWS Snowball Edge. You can't migrate individual tables or subsets of tables using AWS Snowball Edge.
- The AWS DMS Agent doesn't support HTTP/HTTPS or SOCKS proxy configurations. Connections to the source and target can fail if the AWS DMS agent host uses proxies.
- The LOB mode limits LOB file size to 32 K. LOBs larger than 32 K aren't migrated.
- In some cases, an error can occur when loading from the local database to the Edge device or when loading data from Amazon S3 to the target database. In some of these cases, the error is recoverable and the task can restart. If AWS DMS can't recover from the error, the migration stops. If this happens, contact AWS Support.
- You can't use an AWS Snowcone device to migrate data with AWS DMS. AWS Snowcone is the smallest member of the Snow Family of edge computing, edge storage, and data transfer devices.

# AWS DMS reference

In this reference section, you can find additional information you might need when using AWS Database Migration Service (AWS DMS), including data type conversion information.

AWS DMS maintains data types when you do a homogeneous database migration where both source and target use the same engine type. When you do a heterogeneous migration, where you migrate from one database engine type to a different database engine, data types are converted to an intermediate data type. To see how the data types appear on the target database, consult the data type tables for the source and target database engines.

Be aware of a few important things about data types when migrating a database:

- The FLOAT data type is inherently an approximation. When you insert a specific value in FLOAT, it might be represented differently in the database. This difference is because FLOAT isn't an exact data type, such as a decimal data type like NUMBER or NUMBER(p,s). As a result, the internal value of FLOAT stored in the database might be different than the value that you insert. Thus, the migrated value of a FLOAT might not match exactly the value in the source database.

For more information on this issue, see the following articles:

- [IEEE floating point](#) in Wikipedia
- [IEEE floating-point representation](#) on Microsoft Learn
- [Why floating-point numbers may lose precision](#) on Microsoft Learn

## Topics

- [Data types for AWS Database Migration Service \(p. 710\)](#)

## Data types for AWS Database Migration Service

AWS Database Migration Service uses built-in data types to migrate data from a source database engine type to a target database engine type. The following table shows the built-in data types and their descriptions.

AWS DMS data types	Description
STRING	A character string.
WSTRING	A double-byte character string.
BOOLEAN	A Boolean value.
BYTE	A binary data value.
DATE	A date value: year, month, day.
TIME	A time value: hour, minutes, seconds.
DATETIME	A timestamp value: year, month, day, hour, minute, second, fractional seconds. The fractional seconds have a maximum scale of 9 digits. The following format is supported: YYYY:MM:DD

AWS DMS data types	Description
	HH:MM:SS.F(9). For Amazon S3 Select and Amazon S3 Glacier Select, the DATETIME data type format is different. For more information, see the description of the timestamp primitive data type in <a href="#">Supported Data Types</a> of the <i>Amazon Simple Storage Service User Guide</i> .
INT1	A one-byte, signed integer.
INT2	A two-byte, signed integer.
INT4	A four-byte, signed integer.
INT8	An eight-byte, signed integer.
NUMERIC	An exact numeric value with a fixed precision and scale.
REAL4	A single-precision floating-point value.
REAL8	A double-precision floating-point value.
UINT1	A one-byte, unsigned integer.
UINT2	A two-byte, unsigned integer.
UINT4	A four-byte, unsigned integer.
UINT8	An eight-byte, unsigned integer.
BLOB	Binary large object.
CLOB	Character large object.
NCLOB	Native character large object.

**Note**

AWS DMS can't migrate any LOB data type to an Apache Kafka endpoint.

# AWS DMS release notes

Following, you can find release notes for current and previous versions of AWS Database Migration Service (AWS DMS).

AWS DMS uses a semantic versioning scheme to identify a service release. A version consists of a three-component number in the format of X.Y.Z where X represents an *epic* version, X.Y represents a *major* version, and Z represents a *minor* version (**Epic.Major.Minor**).

## AWS Database Migration Service 3.4.7 release notes

The following table shows the new features and enhancements introduced in AWS Database Migration Service (AWS DMS) version 3.4.7.

New feature or enhancement	Description
Support Babelfish as a target	<p>AWS DMS now supports Babelfish as a target. Using AWS DMS, you can now migrate live data from any AWS DMS supported source to a Babelfish, with minimal downtime.</p> <p>For more information, see <a href="#">Using Babelfish as a target for AWS Database Migration Service (p. 431)</a>.</p>
Support IBM Db2 z/OS databases as a source for full load only	<p>AWS DMS now supports IBM Db2 z/OS databases as a source. Using AWS DMS, you can now perform live migrations from Db2 mainframes to any AWS DMS supported target.</p> <p>For more information, see <a href="#">Using IBM Db2 for z/OS databases as a source for AWS DMS (p. 265)</a>.</p>
Support SQL Server read replica as a source	<p>AWS DMS now supports SQL Server read replica as a source. Using AWS DMS, you can now perform live migrations from SQL Server read replica to any AWS DMS supported target.</p> <p>For more information, see <a href="#">Using a Microsoft SQL Server database as a source for AWS DMS (p. 175)</a>.</p>
Support EventBridge DMS events	<p>AWS DMS supports managing event subscriptions using EventBridge for DMS events.</p> <p>For more information, see <a href="#">Working with Amazon EventBridge events and notifications in AWS Database Migration Service (p. 572)</a>.</p>
Support VPC source and target endpoints	<p>AWS DMS now supports Amazon Virtual Private Cloud (VPC) endpoints as sources and targets. AWS DMS can now connect to any AWS service with VPC endpoints when explicitly defined routes to the services are defined in their AWS DMS VPC.</p> <p><b>Note</b> Upgrades to AWS DMS versions 3.4.7 and higher require that you first configure AWS DMS to use VPC endpoints or to use</p>

New feature or enhancement	Description
	<p>public routes. This requirement applies to source and target endpoints for Amazon S3, Amazon Kinesis Data Streams, AWS Secrets Manager, Amazon DynamoDB, Amazon Redshift, and Amazon OpenSearch Service.</p> <p>For more information, see <a href="#">Configuring VPC endpoints as AWS DMS source and target endpoints (p. 437)</a>.</p>
New PostgreSQL version	PostgreSQL version 14.x is now supported as a source and as a target.
Support Aurora Serverless v2 as a target	<p>AWS DMS now supports Aurora Serverless v2 as a target. Using AWS DMS, you can now perform live migrations to Aurora Serverless v2.</p> <p>For information about supported AWS DMS targets, see <a href="#">Targets for data migration (p. 293)</a>.</p>
New IBM Db2 for LUW versions	<p>AWS DMS now supports IBM Db2 for LUW versions 11.5.6 and 11.5.7 as a source. Using AWS DMS, you can now perform live migrations from the latest versions of IBM DB2 for LUW.</p> <p>For information about AWS DMS sources, see <a href="#">Sources for data migration (p. 134)</a>.</p> <p>For information about supported AWS DMS targets, see <a href="#">Targets for data migration (p. 293)</a>.</p>

AWS DMS 3.4.7 includes the following new or changed behavior and resolved issues:

- You can now use a date format from the table definition to parse a data string into a date object when using Amazon S3 as a source.
- New table statistics counters are now available: `AppliedInserts`, `AppliedDdls`, `AppliedDeletes`, and `AppliedUpdates`.
- You can now choose the default mapping type when using OpenSearch as a target.
- The new `TrimSpaceInChar` endpoint setting for Oracle, PostgreSQL, and SQLServer sources allows you to specify whether to trim data on CHAR and NCHAR data types.
- The new `ExpectedBucketOwner` endpoint setting for Amazon S3 prevents sniping when using S3 as a source or target.
- For RDS SQL Server, Azure SQL Server, and self-managed SQL Server — DMS now provides automated setup of MS-CDC on all tables selected for a migration task that are with or without a PRIMARY KEY, or with a unique index considering the enablement priority for MS-REPLICATION on self-managed SQL Server tables with PRIMARY KEY.
- Added support for replication of Oracle Partition and sub-partition DDL Operations during Oracle homogenous migrations.
- Fixed an issue where a data validation task crashes with a composite primary key while using Oracle as a source and target.
- Fixed an issue with correctly casting a varying character type to a boolean while the target column was pre-created as a boolean when using Redshift as a target.
- Fixed an issue that was causing data truncation for `varchar` data types migrated as `varchar(255)` due to a known ODBC issue when using PostgreSQL as a target.
- Fixed an issue where Parallel Hint for the `DELETE` operation wasn't respected with `BatchApplyEnabled` set to `true` and `BatchApplyPreserveTransaction` set to `false` when using Oracle as a target.

- The new `AddTrailingPaddingCharacter` endpoint setting for an Amazon S3 adds padding on string data when using S3 as a target.
- The new `max_statement_timeout_seconds` task setting extends the default timeout of endpoint queries. This setting is currently used by MySQL endpoint metadata queries.
- When using PostgreSQL as a target, fixed an issue where a CDC task wasn't properly utilizing the error handling task settings.
- Fixed an issue where DMS was unable to correctly identify Redis mode for a Redis Enterprise instance.
- Extended the support of `includeOpForFullLoad` extra connection attribute (ECA) for the S3 target parquet format.
- Introduced a new PostgreSQL endpoint setting `migrateBooleanAsBoolean`. When this setting is set to `true` for a PostgreSQL to Redshift migration, a boolean will be migrated as `varchar(1)`. When it is set to `false`, a boolean is migrated as `varchar(15)`, which is the default behavior.
- When using SQL Server source, fixed a migration issue with `datetime` datatype. This fix addresses the issue of inserting `Null` when precision is in milliseconds.
- For PostgreSQL source with `PGLOGICAL`, fixed a migration issue when using `pglogical` and removing a field from the source table during the CDC phase, where the value after the removed field wasn't migrated to the target table.
- Fixed a SQL Server Loopback migration issue with Bidirectional replication getting repeated records.
- Added a new ECA `mapBooleanAsBoolean` for PostgreSQL as a source. Using this extra connection attribute , you can override default data type mapping of a PostgreSQL Boolean to a RedShift Boolean data type.
- Fixed a migration issue when using SQL Server as source that addresses the `ALTER DECIMAL/NUMERIC SCALE` not replicating to targets.
- Fixed connection issue with SQL Server 2005.
- As of October 17, 2022, DMS 3.4.7 now supports Generation 6 Amazon EC2 instance classes for replication instances.
- As of November 25, 2022, with DMS 3.4.7 you can convert database schemas and code objects using **DMS Schema Conversion**, and discover databases in your network environment that are good candidates for migration using **DMS Fleet Advisor**.
- As of November 25, 2022, DMS Studio is retired.
- As of January 31, 2023, DMS Schema Conversion supports Aurora MySQL and Aurora PostgreSQL as a target data provider.

#### **Issues resolved in the DMS 3.4.7 maintenance release dated 16-December-2022**

Topic	Resolution	
BatchApplyEnabled	Fixed an issue of excessive logging when <code>BatchApplyEnabled</code> is set to <code>True</code> .	
New MongoDB endpoint setting— <code>FullLoadNoCursorTimeout</code>	MongoDB endpoint setting <code>FullLoadNoCursorTimeout</code> specifies <code>NoCursorTimeout</code> for the full load cursor. <code>NoCursorTimeout</code> is a MongoDB connection setting that prevents the server from closing the cursor if idle.	
MongoDB—Filter function for single column segmentation	New filter function improves performance for migrating MongoDB databases using a single column for segmentation.	
MongoDB to Redshift	When migrating from MongoDB to Redshift, if the MongoDB collection has binary data type, fixed an issue where DMS was not creating the target table on Redshift.	

Topic	Resolution	
New MongoDB SocketTimeoutMS connection attribute	New MongoDB SocketTimeoutMS extra connection attribute configures the connection timeout for MongoDB clients in units of milliseconds. If the value is less than or equal to zero, then the MongoDB client default is used.	
Fixed issue causing an Amazon Kinesis task to crash	When migrating to Amazon Kinesis Data Streams as a target, fixed an issue handling null values if a primary key wasn't present in the table.	
Oracle NULL PK/UK data validation supported	Removed the limitation that data validation of NULL PK/UK values aren't supported.	
Oracle to Amazon S3	When migrating from Oracle to Amazon S3, fixed an issue where a few records were incorrectly migrated as NULL.	
Oracle Standby	When using Oracle Standby as a source, added the ability for DMS to handle open transactions.	
Oracle to Oracle migration with SDO_GEOOMETRY spatial data type	When migrating from Oracle to Oracle, fixed an issue where the task failed if the table had an SDO_GEOOMETRY column present in the DDL.	
Oracle as source	When using Oracle as a source, fixed an issue where DMS occasionally skips an Oracle redo log sequence number.	
Oracle as source—missing archive/online redo logs	When using Oracle as a source, fixed an issue so that the DMS task fails when archive logs are missing.	
Fixed—DMS occasionally skips Oracle standby redo log	When using Oracle as a source, fixed an issue where DMS occasionally skips an Oracle redo log sequence number.	
Fixed—Oracle to Oracle spatial datatypes not replicating during CDC	When replicating from Oracle to Oracle, fixed an issue where spatial datatypes were not replicating during CDC.	
Oracle as target	When using Oracle as a target, fixed an issue where the target apply was failing with an ORA-01747 error.	
Amazon S3—Fixed reload table data loss	When using Amazon S3 as a target, fixed an issue where a table reload operation wasn't generating CDC files.	
Fixed—SQL Server Always On context initialization in case primary server as a source	When using SQL Server Always On as a source, fixed an issue to not initialize Availability Groups (AG) if the source is primary and AlwaysOnSharedSyncedBackupsEnabled is set to true.	
Updated SQL Server endpoint setting	When a source endpoint is SQL Server Always On Availability Group and is a secondary replica, fixed an issue where the replication task fails if AlwaysOnSharedSyncedBackupsIsEnabled is set to True.	

Topic	Resolution	
PostgreSQL as source	Fixed an issue where CDC fails to migrate delete/update operations on the PostgreSQL source, which was introduced in 3.4.7 in supporting mapBooleanAsBoolean.	

## AWS Database Migration Service 3.4.6 release notes

The following table shows the new features and enhancements introduced in AWS Database Migration Service (AWS DMS) version 3.4.6.

New feature or enhancement	Description
AWS DMS Time Travel	AWS DMS introduces <a href="#">Time Travel</a> , a feature granting customers flexibility on their logging capabilities, and enhancing their troubleshooting experience. With Time Travel, you can store and encrypt AWS DMS logs using Amazon S3, and view, download and obfuscate the logs within a certain time frame.
Support Microsoft Azure SQL Managed Instance as a source	AWS DMS now supports Microsoft Azure SQL Managed Instance as a source. Using AWS DMS, you can now perform live migrations from Microsoft Azure SQL Managed Instance to any AWS DMS supported target.  For information about AWS DMS sources, see <a href="#">Sources for data migration (p. 134)</a> .  For information about supported AWS DMS targets, see <a href="#">Targets for data migration (p. 293)</a> .
Support Google Cloud SQL for MySQL as a source	AWS DMS now supports Google Cloud SQL for MySQL as a source. Using AWS DMS, you can now perform live migrations from Google Cloud SQL for MySQL to any AWS DMS supported target.  For information about AWS DMS sources, see <a href="#">Sources for data migration (p. 134)</a> .  For information about supported AWS DMS targets, see <a href="#">Targets for data migration (p. 293)</a> .
Support parallel load for partitioned data to S3	AWS DMS now supports parallel load for partitioned data to Amazon S3, improving the load times for migrating partitioned data from supported database engine source data to Amazon S3. This feature creates Amazon S3 sub-folders for each partition of the table in the database source, allowing AWS DMS to run parallel processes to populate each sub-folder.
Support multiple Apache Kafka target topics in a single task	AWS DMS now supports Apache Kafka multi-topic targets with a single task. Using AWS DMS, you can now replicate multiple schemas from a single database to different Apache Kafka target topics using the same task. This eliminates the need to create multiple separate tasks in situations where many tables from the same source database need to be migrated to different Kafka target topics.

The issues resolved in AWS DMS 3.4.6 include the following:

- Fixed an issue where columns from UPDATE statements were populated to incorrect columns if the primary key column is not the first column when using Amazon S3 as a target with CSV format.
- Fixed an issue where AWS DMS tasks might crash when using the pglogical plugin with NULL values in BYTEA columns under limited LOB mode when using PostgreSQL as a source.
- Fixed an issue where AWS DMS tasks might crash when a large number of source tables are deleted when using PostgreSQL as a source.
- Improved Amazon S3 date-based folder partitioning by introducing a new Amazon S3 setting DatePartitionTimezone to allow partitioning on non-UTC dates.
- Supported the mapping between data type TIMESTAMP WITH TIME ZONE from sources to Timestamptz when using Redshift as a target
- Improved the performance of CDC for tasks without wildcard selection rules when using MongoDB or Amazon DocumentDB as a source.
- Fixed an issue where schema names with underscore wildcard and length less than 8 were not captured by AWS DMS tasks when using Db2 LUW as a source.
- Fixed an issue where AWS DMS instances ran out of memory under large data volume when using OpenSearch Service as a target.
- Improved the performance of data validation by supporting full load validation only tasks.
- Fixed an issue where AWS DMS tasks failed to resume after forced failover when using Sybase as a source.
- Fixed an issue where AWS DMS sent warning Invalid BC timestamp was encountered in column incorrectly.

Issues resolved in the DMS 3.4.6 maintenance release include the following:

- Fixed an issue of a task crashing when bulk apply mode is enabled when using Oracle as the source and target.
- Fixed an issue so that a full load task properly uses the ExecuteTimeout endpoint setting with PostgreSQL as source.
- Fixed an issue with migrating Array data type columns when the task is set to limited LOB mode while using PostgreSQL as a source.
- Fixed an issue with migrating timestamps with time zone before 1970-01-01 when using PostgreSQL as a source.
- Fixed an issue where DMS was treating an empty string as null during replication when using SQL Server as a source and target.
- Fixed an issue to honor session read and write timeout endpoint settings when using MySQL source/target.
- Fixed an issue where a DMS CDC task was downloading full load related files when using Amazon S3 as a source.
- Fixed a log crashing issue when CdcInsertsAndUpdates and PreserveTransactions are both set to true when using Amazon S3 as a target.
- Fixed an issue where a task crashed when the ParallelApply\* feature is enabled, but some tables don't have a default primary key when using Amazon Kinesis Data Streams as a source.
- Fixed an issue where an error wasn't given for an incorrect StreamArn when using Amazon Kinesis Data Streams as a source.
- Fixed an issue where a primary key value as an empty string would cause a task to crash when using OpenSearch as a target.
- Fixed an issue where too much disk space was used by data validation.

**Issues resolved in the DMS 3.4.6 maintenance release dated 13-December-2022**

Topic	Resolution
SAP ASE odbc driver	Fixed an issue for SAP ASE as a source so that the ODBC driver can support character sets.
Sqlserver datetime primary key bug for lob lookup	Fixed an issue for SQL Server as a source where LOB lookup was not working properly, when primary key has a datetime datatype, with precision in milliseconds.
SQL Server to Redshift-'datetimeoffset' mapped to 'timestamptz'	For migrations from SQL Server to Redshift, improved mapping so that the SQL Server 'datetimeoffset' format is mapped to the Redshift 'timestamptz' format.
Data Validation - SkipLobColumns is True	Fixed an issue where DMS task crashes when SkipLobColumns is True, there is a LOB on the source, the Primary Key is on the last column, and a data difference is detected by validation.
Data Validation with MySQL as source	Fixed an issue for MySQL as a source with data validation enabled, where a DMS task crash occurs while using a table that has a composite unique key that has null values.
MySQL as source	Fixed an issue for MySQL as a source, where a table is getting suspended with Overflow error when the columns are altered to add precision.
Upgrade MySQL ODBC driver to 8.0.23	Fixed an issue for MySQL as a source, where the collation 'utf8mb4_0900_bin' was incompatible with the mysql driver used by DMS.
MySQL-support DDL changes for partitioned tables	Introduced a new MySQL endpoint setting skipTableSuspensionForPartitionDdl to allow the user to skip table suspension for partition DDL changes during CDC, so that DMS can now support DDL changes for partitioned MySQL tables.
MongoDB to Redshift migration	Fixed an issue for MongoDB to Redshift migrations, where DMS fails to create the target table on Redshift if the MongoDB collection has binary data type.
Redshift target-Time Travel Segfault in Bulk Apply	Fixed an issue for Redshift as a target, where DMS task crashes with BatchApplyEnabled set to true.
Redshift as target	Fixed an issue for Redshift as a target, where with parallel-load set to type=partitions-auto, parallel segments were writing bulk CSV files to the same table directory and interfering with each other.
Redshift as target	Fixed an issue with Redshift as a target, where during CDC the target column is of type boolean while the source is of type character varying.
Redshift as target	Improved the task log to identify a DDL change that fails to replicate to Redshift as a target.

Topic	Resolution
Data validation with PostgreSQL	Fixed an issue for validation with PostgreSQL, where the validation fails when boolean data types are present.
PostgreSQL as source	Fixed an issue for PostgreSQL as a source, so that the full load uses the ExecuteTimeout field in Extra connection attributes.
PostgreSQL as source	Fixed an issue for PostgreSQL as a source, so that a task will fail if it's reading LSNs which are greater than the requested task resume LSN for more than 60 min to indicate that this is a problem with the replication slot being used.
PostgreSQL as a source-timestamptz before 1970-01-01	Fixed an issue for PostgreSQL as a source, where timestamptz before 1970-01-01 were not migrated correctly during CDC.
PostgreSQL as source	Fixed an issue for PostgreSQL as a source, where DMS was truncating character varying datatype values during CDC.
PostgreSQL as a source-resuming stopped task	Fixed an issue for PostgreSQL as a source where resuming a previously stopped task replay misses one or more transactions during CDC.
Amazon S3 as target	Fixed an issue for S3 as a target, where the resultant CSV file header is off by one column when AddColumnName is true and TimestampColumnName is "".
Amazon S3 as source—memory usage behavior in full load phase for task	Fixed an issue for S3 as a source, where a DMS task in full load was only releasing the used memory after the entire table was loaded to the target database.
Amazon S3 as target—table reload operation	Fixed an issue for S3 as a target, where a table reload operation missed generating CDC files.

## AWS Database Migration Service 3.4.5 release notes

The following table shows the new features and enhancements introduced in AWS Database Migration Service (AWS DMS) version 3.4.5.

New feature or enhancement	Description
Support for Redis as a target	AWS DMS now supports Redis as a target. Using AWS DMS, you can now migrate live data from any AWS DMS supported source to a Redis data store, with minimal downtime. For information about AWS DMS targets, see <a href="#">Targets for data migration (p. 293)</a> .
Support for MongoDB 4.2 and 4.4 as sources	AWS DMS now supports MongoDB 4.2 and 4.4 as sources. Using AWS DMS, you can now migrate data from MongoDB 4.2 and 4.4 clusters to any AWS DMS supported target including Amazon DocumentDB (with

New feature or enhancement	Description
	MongoDB compatibility), with minimal downtime. For information about AWS DMS sources, see <a href="#">Sources for data migration (p. 134)</a> .
Support for multiple databases using MongoDB as a source	AWS DMS now supports migrating multiple databases in one task using MongoDB as a source. Using AWS DMS, you can now group multiple databases of a MongoDB cluster, and migrate them using one database migration task. You can migrate to any AWS DMS supported target, including Amazon DocumentDB (with MongoDB compatibility), with minimal downtime.
Support for automatic segmentation using MongoDB or Amazon DocumentDB (with MongoDB compatibility) as a source	AWS DMS now supports automatic segmentation using MongoDB or Amazon DocumentDB as a source. Using AWS DMS, you can configure database migration tasks to segment the collection of a MongoDB or DocumentDB cluster automatically. You can then migrate the segments in parallel to any AWS DMS supported target, including Amazon DocumentDB, with minimal downtime.
Amazon Redshift full load performance improvement	AWS DMS now supports using parallel threads when using Amazon Redshift as a target during full load. By taking advantage of the multithreaded full load task settings, you can improve the performance of your initial migration from any AWS DMS supported source to Amazon Redshift. For information about AWS DMS targets, see <a href="#">Targets for data migration (p. 293)</a> .

The issues resolved in AWS DMS 3.4.5 include the following:

- Fixed an issue where data could potentially be missing or duplicated after resuming, when using PostgreSQL as a source with high transaction concurrency.
- Fixed an issue where database migration tasks fail with error **Could not find relation id ...** when using PostgreSQL as a source, with the pglogical plugin enabled.
- Fixed an issue where VARCHAR columns are not replicated correctly when using PostgreSQL as a source and Oracle as a target.
- Fixed an issue where delete operations are not properly captured when the primary key is not the first column in the table definition, when using PostgreSQL as a source.
- Fixed an issue where database migration tasks miss LOB updates in a special metadata setting when using MySQL as a source.
- Fixed an issue where TIMESTAMP columns are treated as DATETIME in full LOB mode when using MySQL version 8 as a source.
- Fixed an issue where database migration tasks fail when parsing NULL DATETIME records when using MySQL 5.6.4 and above as a source.
- Fixed an issue where database migration tasks get stuck after encountering a **Thread is exiting** error when using Amazon Redshift as a target with parallel apply.
- Fixed an issue where data could potentially be lost, when database migration tasks disconnect with a Amazon Redshift target endpoint during batch-apply CDC.
- Improved the performance of full load by calling ACCEPTINVCHARS when using Amazon Redshift as a target.
- Fixed an issue where duplicated records are replicated when reverting from one-by-one mode to parallel apply mode using Amazon Redshift as a target.
- Fixed an issue where database migration tasks do not switch Amazon S3 object ownership to bucket owner with cannedAclForObjects=bucket\_owner\_full\_control when using Amazon S3 as a target.

- Improved AWS DMS by supporting multiple archive destinations with ECA additionalArchivedLogDestId when using Oracle as a source.
- Fixed an issue where database migration tasks fail with error OCI\_INVALID\_HANDLE while updating a LOB column in full LOB mode.
- Fixed an issue where NVARCHAR2 columns are not migrated properly during CDC when using Oracle as a source.
- Improved AWS DMS by enabling SafeguardPolicy when using RDS for SQL Server as a source.
- Fixed an issue where database migration tasks report error on rdsadmin when using a non-RDS SQL Server source.
- Fixed an issue where data validation fails with UUID as the primary key in a partition setting when using SQL Server as a source.
- Fixed an issue where full load plus CDC tasks might fail if the required LSN cannot be found in the database log when using Db2 LUW as a source.
- Improved AWS DMS by supporting custom CDC timestamps when using MongoDB as a source.
- Fixed an issue where database migration tasks get stuck when stopping, using MongoDB as a source, when the MongoDB driver errors on endSessions.
- Fixed an issue where AWS DMS fails to update non-primary fields when using DynamoDB as a target.
- Fixed an issue where data validation reports false positive mismatches on CL0B and NCL0B columns.
- Fixed an issue where data validation fails on whitespace-only records when using Oracle as a source.
- Fixed an issue where database migration tasks crash when truncating a partitioned table.
- Fixed an issue where database migration tasks fail when creating the awsdms\_apply\_exceptions control table.
- Extended support of the caching\_sha2\_password authentication plugin when using MySQL version 8.

## AWS Database Migration Service 3.4.4 release notes

The following table shows the new features and enhancements introduced in AWS DMS version 3.4.4.

New feature or enhancement	Description
Support TLS encryption and TLS or SASL authentication using Kafka as a target	AWS DMS now supports TLS encryption and TLS or SASL authentication using Amazon MSK and on-premises Kafka cluster as a target. For more information on using encryption and authentication for Kafka endpoints, see <a href="#">Connecting to Kafka using Transport Layer Security (TLS) (p. 390)</a> .

The issues resolved in AWS DMS 3.4.4 include the following:

- Improved AWS DMS logging on task failures when using Oracle endpoints.
- Improved AWS DMS task execution continues processing when Oracle source endpoints switch roles after Oracle Data Guard fail over.
- Improved error handling treats ORA—12561 as a recoverable error when using Oracle endpoints.
- Fixed an issue where EMPTY\_BLOB() and EMPTY\_CL0B() columns are migrated as null when using Oracle as a source.

- Fixed an issue where AWS DMS tasks fail to update records after add column DDL changes when using SQL Server as a source.
- Improved PostgreSQL as a source migration by supporting the `TIMESTAMP WITH TIME ZONE` data type.
- Fixed an issue where the `afterConnectScript` setting does not work during a full load when using PostgreSQL as a target.
- Introduced a new `mapUnboundedNumericAsString` setting to better handle the `NUMERIC` date type without precision and scale when using PostgreSQL endpoints.
- Fixed an issue where AWS DMS tasks fail with “0 rows affected” after stopping and resuming the task when using PostgreSQL as a source.
- Fixed an issue where AWS DMS fails to migrate the `TIMESTAMP` data type with the `BC` suffix when using PostgreSQL as a source.
- Fixed an issue where AWS DMS fails to migrate the `TIMESTAMP` value “`±infinity`” when using PostgreSQL as a source.
- Fixed an issue where empty strings are treated as `NULL` when using S3 as a source with the `csvNullValue` setting set to other values.
- Improved the `timestampColumnName` extra connection attribute in a full load with CDC to be sortable during CDC when using S3 as a target.
- Improved the handling of binary data types in hex format such as `BYTE`, `BINARY`, and `BLOB` when using S3 as a source.
- Fixed an issue where deleted records are migrated with special characters when using S3 as a target.
- Fixed an issue to handle empty key values when using Amazon DocumentDB (with MongoDB compatibility) as a target.
- Fixed an issue where AWS DMS fails to replicate `NumberDecimal` or `Decimal128` columns when using MongoDB or Amazon DocumentDB (with MongoDB compatibility) as a source.
- Fixed an issue to allow CDC tasks to retry when there is a fail over on MongoDB or Amazon DocumentDB (with MongoDB compatibility) as a source.
- Added an option to remove the hexadecimal “`0x`” prefix to `RAW` data type values when using Kinesis, Kafka, or OpenSearch as a target.
- Fixed an issue where validation fails on fixed length character columns when using Db2 LUW as a source.
- Fixed an issue where validation fails when only the source data type or the target data type is `FLOAT` or `DOUBLE`.
- Fixed an issue where validation fails on `NULL` characters when using Oracle as a source.
- Fixed an issue where validation fails on XML columns when using Oracle as a source.
- Fixed an issue where AWS DMS tasks crash when there are nullable columns in composite keys using MySQL as a source.
- Fixed an issue where AWS DMS fails to validate both `UNIQUEIDENTIFIER` columns from SQL Server source endpoints and `UUID` columns from PostgreSQL target endpoints.
- Fixed an issue where a CDC task does not use an updated source table definition after it is modified.
- Improved AWS DMS fail over to treat task failures caused by an invalid user name or password as recoverable errors.
- Fixed an issue where AWS DMS tasks fail because of missing LSNs when using RDS for SQL Server as a source.

# AWS Database Migration Service 3.4.3 release notes

The following table shows the new features and enhancements introduced in AWS DMS version 3.4.3.

New feature or enhancement	Description
New Amazon DocumentDB version	Amazon DocumentDB version 4.0 is now supported as a source.
New MariaDB version	MariaDB version 10.4 is now supported as both a source and target.
Support for AWS Secrets Manager integration	You can store the database connection details (user credentials) for supported endpoints securely in AWS Secrets Manager. You can then submit the corresponding secret instead of plain-text credentials to AWS DMS when you create or modify an endpoint. AWS DMS then connects to the endpoint databases using the secret. For more information on creating secrets for AWS DMS endpoints, see <a href="#">Using secrets to access AWS Database Migration Service endpoints (p. 615)</a> .
Larger options for C5 and R5 replication instances	You can now create the following larger replication instance sizes: C5 sizes up to 96 vCPUs and 192 GiB of memory and R5 sizes up to 96 vCPUs and 768 GiB of memory.
Amazon Redshift performance improvement	AWS DMS now supports parallel apply when using Redshift as a target to improve the performance of on-going replication. For more information, see <a href="#">Multithreaded task settings for Amazon Redshift (p. 327)</a> .

The issues resolved in AWS DMS 3.4.3 include the following:

- Fixed an issue where commit timestamp became “1970-01-01 00:00:00” for deferred events when using Db2 LUW as a source.
- Fixed an issue where AWS DMS tasks failed with an NVARCHAR column as primary key when using SQL Server as a source with Full LOB mode.
- Fixed an issue of missing records during cached changes phase when using SQL Server as a source.
- Fixed an issue where records were skipped after AWS DMS tasks were resumed when using RDS for SQL Server as a source.
- Fixed an issue where AWS DMS ASSERTION logging component generates large logs for SQL Server.
- Fixed an issue where data validation failed during CDC phase due to column parsing overflow when using MySQL as a source.
- Fixed an issue where AWS DMS tasks crashed due to a segmentation fault during data validation when using PostgreSQL as a target.
- Fixed an issue where data validation failed on DOUBLE data type during CDC when using PostgreSQL as a source and a target.
- Fixed an issue where records inserted by copy command were not replicated correctly when using PostgreSQL as a source and Redshift as a target.
- Fixed a data loss issue during cached changes phase when using PostgreSQL as a source.
- Fixed an issue which could potentially cause either data loss or record duplicates when using PostgreSQL as a source.

- Fixed an issue where schemas with mixed cases failed to migrate with pglogical when using PostgreSQL as a source.
- Fixed an issue where the Last Failure Message did not contain the ORA error when using Oracle as a source.
- Fixed an issue where AWS DMS tasks failed to build UPDATE statements when using Oracle as a target.
- Fixed an issue where AWS DMS tasks did not replicate data when using Oracle 12.2 as a source with ASM and Pluggable Database configuration.
- Improved record parsing by preserving quotes to be compliant with RFC 4180 when using S3 as a source.
- Improved the handling of timestampColumnName so that the column from Full Load is sortable with that from CDC.
- By introducing a new endpoint setting MessageMaxBytes, fixed an issue where AWS DMS tasks failed when there are LOB elements larger than 1MB .
- Fixed an issue where AWS DMS tasks crashed due to a segmentation fault when using Redshift as a target.
- Improved error logging for Redshift test connection.
- Fixed an issue where AWS DMS did not transfer all documents from MongoDB to DocumentDB during Full Load.
- Fixed an issue where AWS DMS tasks reported fatal error when no tables were included in the table mapping rules.
- Fixed an issue where schemas and tables created before restarting AWS DMS tasks did not replicate to the target when using MySQL as a source.
- Fixed an issue where wildcard escape [ ] cannot escape wildcard "\_" in exclude rule when using MySQL as a source.
- Fixed an issue where column of data type UNSIGNED BIGINT did not replicate correctly when using MySQL as a source.

## AWS Database Migration Service 3.4.2 release notes

The following table shows the new features and enhancements introduced in AWS DMS version 3.4.2.

New feature or enhancement	Description
Support for privately connecting your Amazon Virtual Private Cloud (Amazon VPC) to AWS Database Migration Service (DMS) without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection.	<p>You can now connect to and access AWS DMS from your Amazon VPC through a VPC interface endpoint that you create. This interface endpoint allows you to isolate all network activity of your AWS DMS replication instance within the Amazon network infrastructure. By including a reference to this interface endpoint in all API calls to AWS DMS using the AWS CLI or an SDK, you ensure that all AWS DMS activity remains invisible to the public Internet. For more information, see <a href="#">Infrastructure security in AWS Database Migration Service (p. 626)</a>.</p> <p><b>Note</b>                      This feature is available using all supported AWS DMS engine versions.</p>
CDC date-based folder partitioning using Amazon S3 as a target	AWS DMS now supports date-based folder partitioning when replicating data using S3 as a target. For more information, see <a href="#">Using date-based folder partitioning (p. 344)</a> .

The issues resolved in AWS DMS 3.4.2 include the following:

- Added a STATUPDATE option when performing a migration using Redshift as a target.
- Improved validation tasks by introducing a new setting. ValidQueryCdcDelaySecond delays the first validation query on both source and target endpoints to help reduce resource contention when migration latency is high.
- Fixed an issue where AWS DMS took a long time to start validation tasks.
- Fixed an issue where empty records were generated when starting or stopping replication tasks using S3 as a target.
- Fixed an issue where tasks got stuck after a full load completed.
- Fixed an issue where tasks got stuck when a source table has data errors while using S3 as a source .
- Fixed an issue where tasks got stuck while starting when the user account of the source endpoint is disabled.
- Fixed an issue where tasks crashed when using PostgreSQL as a source with REPLICA IDENTITY FULL.
- Fixed an issue where tasks missed transactions when using PostgreSQL as a source with **pglogical** plugin.
- Fixed an issue when AWS DMS didn't delete compressed source files when using Redshift as a target.
- Fixed an issue where validation tasks reported false negatives when using MySQL as both source and target with data type BIGINT UNSIGNED.
- Fixed an issue where validation tasks reported false positives when using SQL Server as a source with a primary key column as CHAR type.
- Fixed an issue where AWS DMS doesn't clear target objects when using start-replication to start replication tasks using S3 as a target.
- Fixed several issues on data validation when using Db2 as a source.
- Fixed an issue where validation tasks got stuck when using SQL Server as a source with VARCHAR column as primary key.
- Added support for data type TIMESTAMP WITH TIMEZONE when using PostgreSQL as a source

## AWS Database Migration Service 3.4.1 Beta release notes

The following table shows the new features and enhancements introduced in AWS DMS version 3.4.1 Beta.

New feature or enhancement	Description
New MongoDB version	MongoDB version 4.0 is now supported as a source.
TLS 1.2 support for SQL Server	AWS DMS now supports TLS 1.2 for SQL Server endpoints.

The issues resolved in AWS DMS 3.4.1 Beta include the following:

- Improved Oracle 19c TDE support.
- Improved support of utf8mb4 character set and identity data type using Redshift as a target.
- Improved replication task failure handling when using MySQL as a source and binary log is not present.

- Improved data validation support on various data types and character sets.
- Improved null value handling with a new endpoint setting `IncludeNullAndEmpty` when using Kinesis and Kafka as a target.
- Improved error logging and handling when using Kafka as a target.
- Improved DST time offset when using SQL Server as a source.
- Fixed an issue where replication tasks try to create existing tables for Oracle as a target.
- Fixed an issue where replication tasks get stuck after the database connection is killed when using Oracle as a source.
- Fixed an issue where replication tasks failed to detect and reconnect to the new primary when using SQL Server as a source with `AlwaysON` setting.
- Fixed an issue where replication tasks do not add a "D" for "OP" column under certain conditions for S3 as a target.

## AWS Database Migration Service 3.4.0 Beta release notes

The following table shows the new features and enhancements introduced in AWS DMS version 3.4.0.

New feature or enhancement	Description
New MySQL version	AWS DMS now supports MySQL version 8.0 as a source, except when the transaction payload is compressed.
TLS 1.2 support for MySQL	AWS DMS now supports TLS 1.2 for MySQL endpoints.
New MariaDB version	AWS DMS now supports MariaDB version 10.3.13 as a source.
Non-SysAdmin access to self-managed Microsoft SQL Server sources	AWS DMS now supports access by non-SysAdmin users to on-premise and EC2-hosted SQL Server source endpoints.  <b>Note</b> This feature is currently in Beta. If you want to try it out, contact AWS support for more information.
CDC tasks and Oracle source tables created using CREATE TABLE AS	AWS DMS now supports both full-load and CDC and CDC-only tasks running against Oracle source tables created using the <code>CREATE TABLE AS</code> statement.

The issues resolved in AWS DMS 3.4.0 include the following:

- Improved premigration task assessments. For more information, see [Enabling and working with premigration assessments for a task \(p. 548\)](#).
- Improved data validation for float, real, and double data types.
- Improved Amazon Redshift as a target by better handling this error: "The specified key does not exist."
- Supports multithreaded CDC load task settings, including `ParallelApplyThreads`, `ParallelApplyBufferSize`, and `ParallelApplyQueuesPerThread`, for Amazon OpenSearch Service (OpenSearch Service) as a target.
- Improved OpenSearch Service as a target by supporting its use of composite primary keys.
- Fixed an issue where test connection fails when using PostgreSQL as a source and the password has special characters in it.

- Fixed an issue with using SQL Server as a source when some VARCHAR columns are truncated.
- Fixed an issue where AWS DMS does not close open transactions when using Amazon RDS SQL Server as a source. This can result in data loss if the polling interval parameter is set incorrectly. For more information on how to setup a recommended polling interval value, see [Using a Microsoft SQL Server database as a source for AWS DMS \(p. 175\)](#).
- Fixed an issue for Oracle Standby as source where CDC tasks would stop unexpectedly when using Binary Reader.
- Fixed an issue for IBM DB2 for LUW where the task failed with the message "The Numeric literal 0 is not valid because its value is out of range."
- Fixed an issue for a PostgreSQL to PostgreSQL migration when a new column was added on the PostgreSQL source and the column was created with a different data type than the data type for which the column was originally created on the source.
- Fixed an issue with a MySQL source the migration task stopped unexpectedly when it was unable to fetch binlogs.
- Fixed an issue related to an Oracle target when BatchApply was being used.
- Fixed an issue for MySQL and MariaDB when migrating the TIME data type.
- Fixed an issue for an IBM DB2 LUW source where migrating tables with LOBs fail when the tables don't have a primary key or unique key.

## AWS Database Migration Service 3.3.4 release notes

The issues resolved in AWS DMS 3.3.4 include the following:

- Fixed an issue where transactions are dropped or duplicated when using PostgreSQL as a source.
- Improved the support of using dollar sign (\$) in schema names.
- Fixed an issue where replication instances do not close open transactions when using RDS SQL Server as a source.
- Fixed an issue where test connection fails when using PostgreSQL as a source and the password has special characters in it.
- Improved Amazon Redshift as a target by better handling this error: "The specified key does not exist."
- Improved data validation support on various data types and character sets.
- Fixed an issue where replication tasks try to create existing tables for Oracle as a target.
- Fixed an issue where replication tasks do not add a "D" for "OP" column under certain conditions for Amazon S3 as a target.

## AWS Database Migration Service 3.3.3 release notes

The following table shows the new features and enhancements introduced in AWS DMS version 3.3.3.

New feature or enhancement	Description
New PostgreSQL version	PostgreSQL version 12 is now supported as a source and target.

New feature or enhancement	Description
Support for composite primary key with Amazon OpenSearch Service as target	As of AWS DMS 3.3.3, use of a composite primary key is supported by OpenSearch Service targets.
Support for Oracle extended data types	Oracle extended data types for both Oracle source and targets are now supported.
Increased number of AWS DMS resources per account	The limit on the number of AWS DMS resources you can create has increased. For more information, see <a href="#">Quotas for AWS Database Migration Service (p. 652)</a> .

The issues resolved in AWS DMS 3.3.3 include the following:

- Fixed an issue where a task crashes using a specific update statement with Parallel Apply in Amazon Kinesis.
- Fixed an issue where a task crashes on the ALTER TABLE statement with Amazon S3 as a target.
- Fixed an issue where values on polygon columns are truncated when using Microsoft SQL Server as a source.
- Fixed an issue on Unicode converter of JA16SJISTILDE and JA16EUCTILDE when using Oracle as a source.
- Fixed an issue where MEDIUMTEXT and LONGTEXT columns failed to migrate from MySQL to S3 comma-separated value (CSV) format.
- Fixed an issue where boolean columns were transformed to incorrect types with Apache Parquet output.
- Fixed an issue with extended varchar columns in Oracle.
- Fixed an issue where data validation tasks failed due to certain timestamp combinations.
- Fixed an issue with Sybase data definition language (DDL) replication.
- Fixed an issue involving an Oracle Real Application Clusters (RAC) source crashing with Oracle Binary Reader.
- Fixed an issue with validation for Oracle targets with schema names' case.
- Fixed an issue with validation of IBM Db2 versions 9.7 and 10.
- Fixed an issue for a task not stopping two times with StopTaskCachedChangesApplied and StopTaskCachedChangesNotApplied enabled.

# Document history

The following table describes the important changes to the AWS Database Migration Service user guide documentation after January 2018.

You can subscribe to an RSS feed to be notified of updates to this documentation. For more details on AWS DMS version releases, see [AWS DMS release notes \(p. 712\)](#)

Change	Description	Date
<a href="#">Support VPC source and target endpoints</a>	AWS DMS now supports virtual private cloud (VPC) endpoints as sources and targets. AWS DMS can now connect to any AWS service with VPC endpoints when explicitly defined routes to the services are defined in their AWS DMS VPC.	June 30, 2022
<a href="#">Support SQL Server read replica as a source</a>	AWS DMS now supports SQL Server read replica as a source. Using AWS DMS, you can now perform live migrations from SQL Server read replica to any AWS DMS supported target.	June 30, 2022
<a href="#">Support IBM Db2 z/OS databases as a source for full load only</a>	AWS DMS now supports IBM Db2 z/OS databases as a source. Using AWS DMS, you can now perform live migrations from Db2 mainframes to any AWS DMS supported target.	June 30, 2022
<a href="#">Support EventBridge DMS events</a>	AWS DMS supports managing event subscriptions using EventBridge for DMS events.	June 30, 2022
<a href="#">Support Babelfish as a target</a>	AWS DMS now supports Babelfish as a target. Using AWS DMS, you can now migrate live data from any AWS DMS supported source to a Babelfish, with minimal downtime.	June 30, 2022
<a href="#">Support Aurora Serverless v2 as a target</a>	AWS DMS now supports Aurora Serverless v2 as a target. Using AWS DMS, you can now perform live migrations to Aurora Serverless v2.	June 30, 2022
<a href="#">Getting Started Tutorial</a>	An update for the Getting Started Tutorial for AWS DMS. The tutorial uses a MySQL database as a source and a PostgreSQL database as a target.	May 20, 2021

<a href="#">Support for Amazon Neptune as a target</a>	Added support for Amazon Neptune as a target for data migration.	June 1, 2020
<a href="#">Support for Apache Kafka as a target</a>	Added support for Apache Kafka as a target for data migration.	March 20, 2020
<a href="#">Updated security content</a>	Updated and standardized security content as a response to customer requests.	December 20, 2019
<a href="#">Migrating with AWS Snowball Edge</a>	Added support for using AWS Snowball Edge to migrate large databases.	January 24, 2019
<a href="#">Support for Amazon DocumentDB (with MongoDB compatibility) as a target</a>	Added support for Amazon DocumentDB (with MongoDB compatibility) as a target for data migration.	January 9, 2019
<a href="#">Support for Amazon OpenSearch Service and Amazon Kinesis Data Streams as targets</a>	Added support for OpenSearch Service and Kinesis Data Streams as targets for data migration.	November 15, 2018
<a href="#">CDC native start support</a>	Added support for native start points when using change data capture (CDC).	June 28, 2018
<a href="#">Db2 LUW support</a>	Added support for IBM Db2 LUW as a source for data migration.	April 26, 2018
<a href="#">SQL Server as target support</a>	Added support for Amazon RDS for Microsoft SQL Server as a source.	February 6, 2018

# AWS glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS General Reference*.