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. 495).

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.

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. 495).

**• 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. 339) 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.

Limitations on using Amazon Redshift as a target for AWS

Database Migration Service

When using an Amazon Redshift database as a target, AWS DMS doesn't support the following:

• 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.

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.

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. 342).

Setting LOB support for source databases in an AWS DMS task (blz. 351)

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 permitted 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.

• **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 64 K. Therefore, a limited

LOB size of less than 64 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 is 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.

• 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. 332)

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. 367).

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).

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:

353

AWS Database Migration Service User Guide

Replication starting from a CDC start point

• **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**

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

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.

**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.

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.

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. 367).

• 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. 371).

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. 384).

• 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. 392).

**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": [

369

AWS Database Migration Service User Guide

Selection rules and actions

{

"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 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"

}

}

Following, you can find LOB functions that you can use to build transformation rule expressions.

**LOB functions Description**

hex(x) The hex function receives a BLOB as an argument and returns an

uppercase hexadecimal string version of the BLOB content.

randomblob (*N*) The randomblob(*N*) function returns an *N*-byte BLOB that

contains pseudorandom bytes. If *N* is less than 1, a 1-byte random

BLOB is returned.

zeroblob(N) The zeroblob(*N*) function returns a BLOB that consists of *N* bytes

of 0x00.

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

399

AWS Database Migration Service User Guide

Using parallel load for selected

tables, views, and collections

• Microsoft SQL Server

• MySQL

• PostgreSQL

• 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)

To specify the maximum number of tables, views, and collections to load in parallel, use the

MaxFullLoadSubTasks task setting. To specify the maximum number of threads per table, view, or

collection for a parallel-load task, use the ParallelLoadThreads task setting. To specify the buffer

size for a parallel load task, use the ParallelLoadBufferSize task setting. The availability and

settings of ParallelLoadThreadsand ParallelLoadBufferSize depend on the target endpoint.

For more information about the ParallelLoadThreads and ParallelLoadBufferSize settings,

see Target metadata task settings (p. 332). For more information about the MaxFullLoadSubTasks

setting, see Full-load task settings (p. 334). 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 parallelload

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 tablesettings

objects with parallel-load options.

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 tablesettings

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 tablesettings

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 lobsettings

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 bulkmax-

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).

**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"

}

}

]

}

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. 335).

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**.

Working with events and notifications in AWS Database Migration Service

AWS Database Migration Service (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. 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 (p. 442), 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, you 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.

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 ongoing

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 ongoing

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.

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

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 100 request maximum per second

API request refresh rate 4 requests per second

Replication instances per user 60

Total amount of storage for a replication instance 30,000 GB

Event subscriptions per user 60

Replication subnet groups per user 60

Subnets per replication subnet group 60

Endpoints per user 1000

Endpoints per replication instance 100

Tasks per user 600

Tasks per replication instance 200

Certificates per user 100