Oracle® DIVArchive

DIVAmigrate Embedded Utility User's Guide

Release 7.5

E79750-02

April 2017

Oracle DIVArchive DIVAmigrate Embedded Utility User's Guide, Release 7.5

E79750-02

Primary Author: Lou Bonaventura

Contents

Preface vii

Audience vii
Documentation Accessibility vii
Related Documents vii
Conventions vii

1 Introduction

DIVAmigrate Embedded Utility Overview 1-1
DIVAmigrate Supported Environments 1-1

2 Installing and Configuring DIVAmigrate

Installing DIVAmigrate 2-1
Windows Files and Folders 2-1
Linux Files and Directories 2-1
Configuring the DIVAmigrate Service 2-2
Configuring the Logging Settings 2-4

3 Operations

Starting and Stopping the DIVAmigrate Service 3-2
Migration Job Command Syntax 3-3
Using the DIVAmigrate GUI 3-5
Migration Job Events 3-5
Using the DIVAmigrate Migration Wizard 3-6
Using the DIVAmigrate Panel 3-6
Migration Job Functions and Parameters 3-7
Migration Job Definitions 3-7
Migration Job Actions 3-8
Migration Job Status 3-8
Migration Job Parameters 3-9
Migration Source 3-9
Migration Destination 3-10
Migration Strategy 3-11
Migration Options 3-11
Basic Migration Jobs 3-13
Copying Data to another Group or Array 3-13

Scenario 1 3-13
Scenario 2 3-13
Moving Data to another Group or Array 3-14
Copying and Migrating Data to the same Group or Array 3-15
Stopping and Resuming Jobs 3-17
Advanced Migration Jobs 3-17
Speeding up Tape to Tape Migration using a Disk Buffer 3-17
Sample Scenario 3-18
Creating Multiple Instances in the Destination Group or Array 3-19
Migrating to Multiple Destination Groups or Arrays 3-20
Default Destination Instance Count 3-21
Repacking Tapes 3-21
Despanning Instances 3-22
Using Alternate Source Instances 3-22
Excluding Objects from Migration 3-23

4 Error Handling and Failure Scenarios

Migration Error Handling 4-1
Migration Failure Scenarios 4-1

A DIVArchive Options and Licensing

B Default Configuration File

List of Tables

4–1 DIVAmigrate Failure Scenarios and Resolutions 4-2

Preface

This document describes installation, configuration, and operations of the Oracle DIVArchive DIVAmigrate Embedded Utility.

Audience

This document is intended for the Oracle Installation Team, System Administrators, and Operations personnel to enable full functionality of the DIVAmigrate Embedded Utility.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documents

For more information, see the Oracle DIVArchive documentation set for this release located at `https://docs.oracle.com/en/storage/#csm`.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1 Introduction

This chapter describes an overview of the Oracle DIVArchive DIVAmigrate Embedded Utility and the environments supported by the tool.

DIVAmigrate Embedded Utility Overview

DIVAmigrate is installed as part of the Oracle DIVArchive Suite's standard installation. It is located in the `%DIVA_HOME%\Program\` folder, and runs as a Windows Service.

You create migration jobs through the DIVArchive Control GUI connected to Oracle DIVArchive Manager, or using the command-line interface through the `client.bat` file located in the `%DIVA_HOME%\Program\Migrate\bin` folder.

You control the utility using the `migrate.bat` file, also located in `%DIVA_HOME%\Program\Migrate\bin` folder. See Starting and Stopping the DIVAmigrate Service for details.

Migration Jobs are stored in the DIVArchive Manager Database. The DIVAmigrate Service monitors the DIVArchive Manager Database, runs new Migration Jobs, and also updates the status of existing Migration Jobs in the database so that the Control GUI displays the status to users.

See Appendix A for Oracle DIVArchive options and licensing information.

DIVAmigrate Supported Environments

The DIVAmigrate Utility is compatible with the following environments:

Windows Server 2012 R2 and later
See the Windows installation instructions in the *Oracle DIVArchive Installation and Configuration Guide* in the *Oracle DIVArchive 7.5 Core documentation* library for detailed information.

Oracle Linux 7 x86_64 and later
See the Linux installation instructions in the *Oracle DIVArchive Installation and Configuration Guide* in the *Oracle DIVArchive 7.5 Core documentation* library for Linux-specific instructions for running DIVArchive components as services under Linux.

2

Installing and Configuring DIVAmigrate

This chapter describes installing and configuring the DIVAmigrate utility and includes the following information:

- Installing DIVAmigrate
- Configuring the DIVAmigrate Service
- Configuring the Logging Settings

Installing DIVAmigrate

The DIVAmigrate Utility is part of the standard DIVArchive installation, and is placed in the `%DIVA_HOME%\Program\` folder. You can install DIVAmigrate on the DIVArchive Manager computer, or any other computer capable of communicating with the Manager using the TCP/IP protocol. You can confirm connectivity by successfully pinging the Manager from the client computer.

Windows Files and Folders

You will find the following new files and folders after you complete the initial DIVArchive installation in Windows:

```
%DIVA_HOME%\Program
   Migrate
      bin
         client.bat
         migrate.bat
      lib
         migrate.jar
   conf
      migrate
         migrate.conf.ini
   log
      migrate
```

Linux Files and Directories

You will find the following new files and directories after you complete the initial DIVArchive installation in Linux:

```
%DIVA_HOME%/Program
   Migrate
      bin
         client.sh
         migrate.sh
      lib
         migrate.jar
   conf
      migrate
         migrate.conf.ini
   log
      migrate
```

Configuring the DIVAmigrate Service

The DIVAmigrate Service requires a valid configuration file during install and start procedures. The default DIVAmigrate configuration file is named `migrate.conf` and is located in the `%DIVA_HOME%\Program\conf\migrate\` folder.

The configuration file is a standard properties file similar to the Manager configuration file. The configuration file is not auto-reloadable, and therefore any changes made to the file do not take effect until the DIVAmigrate Service is restarted.

Notes: The *Windows Service Wrapper* configuration must not be modified, and *is not included* in the default configuration file in Appendix B.

The *DIVA Service Options* section also must not be modified unless instructed by Oracle Support. However, this section *is included* in the default configuration file in Appendix B.

Use the following procedure to modify the DIVAmigrate configuration file:

Caution: Do not use Word, WordPad, or any other word processor or editing tool that adds extra characters to a file. Always use a plain text editor such as Notepad, or Notepad++.

1. Create a copy of the `migrate.conf.ini` file.

   It is important to create a copy and keep the original file intact to refer back to in case the configuration you are working on either does not work, becomes corrupt, or has or creates errors.

2. Rename the copied file to `migrate.conf`.

3. Open the file with any plain text editor and populate the following parameters. These parameters are all mandatory unless otherwise noted in the description.

   `SERVICE_NAME`
   This parameter is the name for the Windows Service. The default value is `DivaMigrate`.

   `DIVAMANAGER_HOST`
   This parameter is the host name or IP address of the DIVArchive Manager. The default value is `127.0.0.1`.

   `DIVAMANAGER_PORT`
   This parameter is the port number to connect to the DIVArchive Manager. The default value is `9000`.

DIVA_MIGRATE_MANAGEMENT_PORT
This parameter is the management port number. The default value is `9191`.

DIVAMANAGER_DBURL
This parameter is a URL to connect to the database. You can use this parameter instead of using the `DIVAMANAGER_DBHOST` and `DIVAMANAGER_DBPORT`, or `DIVAMANAGER_TNSNAME` parameters. When you use this parameter you have the additional flexibility to explicitly identify whether to use the Oracle JDBC thin driver, or the Oracle JDBC OCI driver, to connect to the Oracle Database. For example, you can use `jdbc:oracle:thin:@host:port:oracle_sid` or `jdbc:oracle:oci:@tnsname`. *This is not a mandatory parameter.*

DIVAMANAGER_DBUSER
This parameter is the user name the Manager uses to connect to the DIVArchive Database. The default value is `diva`, and is case sensitive.

DIVAMANAGER_TNSNAME
This parameter is the *TNS Name* of the DIVArchive Schema within the Oracle Database. DIVAmigrate ignores this setting if the `DIVAMNAGER_DBHOST` and `DIVAMANAGER_DBPORT` settings are defined. There must be a corresponding entry in `TNSNAMES.ORA` found under the Oracle 11 Client installation. *This is not a mandatory parameter.*

DIVAMANAGER_DBHOST
This parameter specifies the host name or IP address of the computer containing the DIVArchive Database. If using a host name, this must be present in the `hosts` file on the computer where the DIVArchive Manager is installed. For example, you can use either `127.0.0.1` or `localhost.` *This is not a mandatory parameter.*

DIVAMANAGER_DBPORT
This parameter is the *Oracle Listener Port* you configured during the DIVArchive Database installation. The default value is port `1521`. *This is not a mandatory parameter.*

DIVAMANAGER_DBSID
The DIVArchive Database instance *System Identifier* (SID) in the Oracle Database where the DIVArchive Manager connects. This value is typically `lib5`, which is the default value. Consult your location's System Configuration Plan for confirmation.

DIVAMANAGER_DBSERVICENAME
This parameter is the DIVArchive Database *Instance Identifier* (SID) in Oracle where DIVArchive Manager connects. Typically, `lib5.world` (the default), is used in most DIVArchive installations. Consult your delivery plan if you are unsure.

You must set either this value, or `DIVAMANAGER_DBSID`, when you use `DIVAMANAGER_DBHOST` and `DIVAMANAGER_DBPORT` for database connections. If you set both parameters, then `SERVICENAME` takes precedence over `SID`.

MAX_SIMULTANEOUS_REQUESTS
This parameter is the maximum number of simultaneous Manager requests processed by DIVAmigrate. The default value is `15`. *This is not a mandatory parameter.*

DB_SCAN_PERIODICITY
This parameter (in seconds) determines how often DIVAmigrate looks for new jobs in the database. The default value is `60` seconds. *This is not a mandatory parameter.*

DIVA_RECONNECT_PERIODICITY
This parameter (in seconds) determines the time between reconnection attempts if connectivity with the Manager is lost. The default value is `30` seconds.

MAX_FAILED_REQUESTS_PAUSE
This parameter identifies the maximum number of sequential failure requests that can occur before DIVAmigrate pauses the migration job. DIVAmigrate pauses the job if the configured number of requests fails sequentially. The default value is `10`. *This is not a mandatory parameter*.

JOB_MAX_INACTIVE_TIME
This parameter (in hours) determines the Migrations Plan's maximum inactivity time. If, after the service is restarted, it finds running jobs having the last access time greater than this value, the Migration Plan for those jobs are recreated. The default value is `24` hours. *This is not a mandatory parameter*.

Configuring the Logging Settings

DIVAmigrate uses the same logging methods used for the DIVArchive Manager. However, DIVAmigrate logs are located in the `%DIVA_HOME%\Program\log\migrate` folder. DIVAmigrate logs are automatically archived and divided into separate files each time the current log file reaches its size limit. You set the following DIVAmigrate logging parameters in the `migrate.conf` file:

LOGGING_DIRECTORY
This parameter identifies the DIVAmigrate log file storage directory. The default is `../../log/migrate`.

LOGGING_ROOT_LEVEL, LOGGING_TRACE_LEVEL, and LOGGING_SERVICE_LEVEL
You can modify these three parameters to suit the required level of activity logging. The default value is **INFO**.

Tip: Only use the higher logging levels when instructed to do so by Oracle Support to avoid large log files being created.

Valid options for each parameter are:

- **DEBUG**
- **INFO**
- **WARN**
- **ERROR**
- **FATAL**

LOGGING_MAXFILESIZE
This parameter identifies the maximum size of the log file before it is archived. When the current log file is archive, a new file is created. You must specify the file size using KB or MB to indicate Kilobytes or Megabytes respectively. The default value is `10MB`. For example, `LOGGING_MAXFILESIZE=10MB`.

LOGGING_LIFETIME
All files older than the value of this parameter are removed. This includes `trace`, `service`, and `.zip` files. The value for this parameter is in hours, and the default value is `50`.

3 Operations

This chapter describes DIVAmigrate operations and includes the following information:

- Starting and Stopping the DIVAmigrate Service
- Migration Job Command Syntax
- Using the DIVAmigrate GUI
  - Migration Job Events
  - Using the DIVAmigrate Migration Wizard
  - Using the DIVAmigrate Panel
- Migration Job Functions and Parameters
  - Migration Job Definitions
  - Migration Job Actions
  - Migration Job Status
  - Migration Job Parameters
- Basic Migration Jobs
  - Copying Data to another Group or Array
  - Moving Data to another Group or Array
  - Copying and Migrating Data to the same Group or Array
  - Stopping and Resuming Jobs
- Advanced Migration Jobs
  - Speeding up Tape to Tape Migration using a Disk Buffer
  - Creating Multiple Instances in the Destination Group or Array
  - Migrating to Multiple Destination Groups or Arrays
  - Default Destination Instance Count
  - Repacking Tapes
  - Despanning Instances
  - Using Alternate Source Instances
  - Excluding Objects from Migration

Starting and Stopping the DIVAmigrate Service

You can start and stop the DIVAmigrate utility from the command-line, or the Windows Service Manager Console. The Windows Service short name displayed in the Windows Service Manager Console is `DivaMgrt`, and the full name displayed is `DIVArchive Migrate - DIVAmigrate`.

To install the DIVAmigrate utility, open a command-line interface, and navigate to the folder where the `migrate.bat` (or `migrate.sh` in Linux) file is located. Run the batch (or script) file with the appropriate parameters required to perform the desired tasks.

Caution: It is your responsibility to confirm that only one instance of the DIVAmigrate Service is running at any time. Running several instances of DIVAmigrate connected to same DIVArchive Manager leads to incorrect Migration Job processing, and may result in data loss.

The following is an example procedure to run the service on a Windows 2012 R2 platform.

1. Open a Windows command-line interface.
2. Change to the `%DIVA_HOME%\Program\Migrate\bin` folder.
3. Execute a command similar to `migrate.bat install`. This specific command installs the service.
4. Execute `migrate.bat start` to start the service. The service is now running.

The following is an example procedure to run the service on an Oracle Linux platform. *Linux commands, file names, and paths are case-sensitive.*

1. Open a command-line interface.
2. Navigate to `/home/diva/DIVA/Program/Migrate/bin`.
3. Execute `migrate.sh`.

The `migrate.bat` command-line syntax is `migrate.bat {command} [option]`. The available commands and options are as follows:

`install`
This command install the module as a Windows service.

`uninstall`
This command removes the module Windows service.

`start`
This command starts the service.

`stop`
This command stops the service.

`restart`
This command stops, and then subsequently starts the service.

`version`
This command displays the module release information, and then exits.

`status`
This command displays the current status of the module (for example, running, stopped, and so on).

`help`
This command displays command and option help information, and then exits.

`-conf` or `-f`
This parameter is optional and specifies a specific configuration file to load.

Migration Job Command Syntax

You can start DIVAmigrate Migration Jobs using either the DIVArchive Control GUI or the command-line interface. The command-line interface uses the `client.bat` batch file in Windows, or the `client.sh` script file in Linux. The Windows batch file is located in the `%DIVA_HOME%\Program\Migrate\bin` folder, and in the Linux script file is located in the `/home/diva/DIVA/Program/Migrate/bin` directory.

The `client` command expects all of the command line options to follow in the exact sequence specified by the syntax.

For example, the following command fails:

```
client -tape 1S0001 -buffer array1 -media group2 -count 1 -delete
```

The displayed error message is:

```
Error. Check parameters. Use client.bat -help for valid commands
description.
```

This is because the `-buffer` option is out of sequence, and is expected to occur after the `-media` and `-count` options.

You can use the following syntax for the associated migration jobs:

```
client -tape {tape_barcode} [-media {media_group}] [-count {count}] [-buffer
{array}] [-delete] [-excl {file_name}] [-autosrc|recovery] [-start {date_time}]
[-end {date_time}] [-priority {value}]

client -tapelist {file_name} [-media {media_group}] [-count {count}] [-buffer
{array}] [-delete] [-excl {file_name}] [-autosrc|recovery] [-start {date_time}]
[-end {date_time}] [-priority {value}]

client -instlist {file_name} [-media {media_group}] [-count {count} [-buffer
{array}] [-delete] [-excl {file_name}] [-autosrc|recovery] [-start {date_time}]
[-end {date_time}] [-priority {value}]

client -tapegroup {mediaName} [-media {media_group}] [-count {count} [-buffer
{array}] [-delete] [-excl {file_name}] [-autosrc|recovery] [-start {date_time}]
[-end {date_time}] [-priority {value}]

client -diskarray {mediaName} [-media {media_group}] [-count {count} [-buffer
{array}] [-delete] [-excl {file_name}] [-autosrc|recovery] [-start {date_time}]
[-end {date_time}] [-priority {value}]

client -{pause|stop|resume|cancel|delete|retry} {jobID}

client -list_running_jobs

client -job_details {jobID}
client -help
```

The command-line options in the previous statements must be in the correct sequence as shown, and are described as follows:

-tape {tape_barcode}
This option identifies the tape to migrate. This will migrate instances present on the tape for the specified barcode.

-tapelist {file_name}
This option identifies a file containing a list of tapes to migrate. This will migrate instances present on the tapes listed in the file. The file is a flat text file (DOS or UNIX format). The format of each line is `barcode`. This option ignores empty lines and lines beginning with `#`.

-instlist {file_name}
This option identifies a file containing a list of instances to migrate. The file is a flat text pipe character separated file (DOS or UNIX format). The format of each line is `objectname|category|instanceid`. This option ignores empty lines and lines beginning with `#`.

-tapegroup {mediaName}
This option migrates all object instances in the entire tape group. The `mediaName` is the tape group name.

-diskarray {mediaName}
This option migrates all object instances in the entire disk array. The `mediaName` is the disk array name.

-media {media_group}
This option identifies the media where objects are being migrated. Specifying a different media group other than the source media enables various migration operations. For example, migrating data from one media to another. When this option is not specified, the destination media group is the source media group of the tape or instance being processed, and corresponds to the `COPY_TO_SOURCE` migration strategy. If multiple destination media are specified (by using multiple `-media` options), you must provide the same amount of `-count` options in the same order (or none to use the default). *This option is mandatory with the `-instlist` and `-tapelist` options.*

-count {count}
This option identifies the target instance count for the objects in the destination media. The default value is `1`. If multiple destination media are specified (by using multiple `-media` options), you must provide the same amount of `-count` options in the same order (or none to use the default). When specifying a `COPY_TO_SOURCE` migration type (where source and target media are the same), `-count 2` is required rather than the default of `-count 1`. This specifies that there will be two instances on the target medium.

-buffer {array}
This option identifies the disk array to use as a buffer.

-delete
This option deletes the source instances after they are processed. This option represents the `MOVE` migration strategy.

`-excl {file_name}`
This option identifies a file containing a list of instances to ignore. The file is a flat text file (DOS or UNIX format). The format of each line is `object|category|instanceId`. The `object` is the object name to exclude, and `category` is the object category. You can use an asterisk to match all object names or all categories. These are mutually exclusive. The `instanceId` is the ID of the instance. Blank spaces in `object`, `category`, and `instanceId` are acceptable. Objects and categories containing a pipe operator within the name are permissible, but the character must be escaped using the backslash character. For example, you must write `object|one` as `object\|one`.

`-autosrc`
This option enables using alternate sources. The Migration Service can search whether the source tape instance exists on an array and can use it for migration, rather then using the tape instance.

`-recovery`
This option enables recovery mode.

`-start {date_time}`
This option specifies the scheduled start time for a job in the format `dd-MM-YYYY-HH:mm`.

`-end {date_time}`
This option specifies the scheduled end time for a job in the format `dd-MM-YYYY-HH:mm`. The job enters the **PAUSE** state if the job is not completed before the scheduled end time.

`-priority {value}`
This option specifies the migration job request priority. This value must be between 1 and 100. The default value is `20`.

`-[command] {jobID}`
This option sends the command used for the specific job. Commands include `pause`, `stop`, `resume`, `cancel`, `delete`, and `retry`.

`-list_running_jobs`
This option displays a list of unfinished jobs, including the `jobID` and status.

`-job_details {jobID}`
This option returns the status and progress for a given `jobID`.

`-help`
This option displays the help text.

Using the DIVAmigrate GUI

The DIVArchive Control GUI contains the *Migrate* panel and *Migration Wizard*. You use the Migration Wizard to create new migration jobs. You view the status and results of completed and currently running Migration Jobs on the *Migration* panel. You can also pause, stop, resume, delete, cancel or rerun migration jobs from the *Migration* panel.

Migration Job Events

The migration service includes events for jobs. All job events are displayed under the **Job Events** tab in Job Properties dialog box. Events are loaded in descending order according to time and event id (by default). The *Events* table in the **Job Events** tab highlights events with different colors based on severity. Red indicates an error, yellow indicates a warning, and white indicates informational messages. Click the **Refresh** button to refresh the entire Job Properties dialog box.

Using the DIVAmigrate Migration Wizard

The Migration Wizard is a typical Windows wizard with a series of dialog boxes. Each dialog box presents a set of choices for migration job parameters and, based on your selections, displays the next screen until all required parameters are configured.

The **Create Migrate Job** button is located under the **Migrate Actions** button on the **Action** tab. You click the button to start the Migration Wizard.

The DIVAmigrate Wizard is only available to users with Administrator privileges in the Control GUI. Use the following procedure to perform a migration using the wizard:

1. Navigate to, and then click, the **Create Migration Job** button in the Control GUI.
2. Select the ***Migration Strategy*** and click **Next**.

   Note: If you selected the **Copy to Source** ***Migration Strategy***, then *Step 3* is skipped because only the ***Media Option*** is available for this strategy. COntinue the procedure with *Step 4*.

3. Select the ***Migration Source Type*** and click **Next**.
4. Use the ***Migration Source*** menu to select the source, and then click **Next**.
5. Use the ***Destination Media*** menu to select the destination.
6. Enter the number of instances to migrate, and then click **Next**.
7. Select **Yes** to add more migration destinations, or **No** to proceed. If you select **Yes**, the previous screen is displayed again so you can select additional destinations. This process repeats until you select **No**.
8. Click **Next** when you have finished adding destinations.
9. Select, or enter, the value in each field appropriate for this migration job on the Migration Options screen, and then click **Next**.
10. The final wizard screen now displays with your selections made during the process. On this screen you can either click **Finish** to schedule the job, **Cancel** to exit the Wizard (no job will be created), or **Previous** to go back and make any necessary changes.
11. Confirm your selections, and then click the appropriate button to either finish, modify, or cancel the migration job.

Using the DIVAmigrate Panel

The *DIVAmigrate* panel is located in the DIVArchive Control GUI and displays information about migration jobs in a table containing the following columns. Some columns are filterable as noted.

- ID (this value is a number)
- Status (filterable)
- Type (filterable and the value can be Copy, Move, or Copy to Source)

- Source Type
- Destination Type
- Started At (filterable)
- Finished At (filterable)
- Progress
- Instances to Migrate
- Data to Transfer (Gb)

A single row selection model is used for the table, and the context dialog is available for each row with following options. See Migration Job Actions for detailed information.

- Properties (opens selected migration job properties dialog box)
- Cancel
- Stop
- Pause
- Resume
- Rerun
- Delete

The Request Properties dialog box shows the status for each object affected by the migration job, and the *jobID* of the last request executed for that object. You click the *Request ID* to open the Request Properties dialog box. The *Object Details* table in the Migration Job Properties dialog box displays 1000 objects at a time and supports pagination.

Migration Job Functions and Parameters

This sections describes the migration job functions and parameters.

Caution: DIVAmigrate does not allow any changes to the migration job parameters and options after a migration job is submitted.

Migration Job Definitions

You create migration jobs using DIVAmigrate, and then DIVAmigrate attempts to run them. You must configure each of the following mandatory parameters for each migration:

Migration Source
This parameter describes the set of instances that must be migrated.

Migration Destination
This parameter specifies the location where migrated instance are to be placed.

Migration Strategy
This parameter specifies the type of migration. That is, **Copy**, **Move**, or **Copy to Source**.

Migration Options
These are additional parameters and options available for migration jobs.

Migration Job Actions

You can perform the following actions on migration jobs:

Create New Job
This selection creates a new migration job.

Cancel Job
This selection stops the specified job without the possibility of resuming. The job is assigned the **Cancelled** status.

Rerun Job
This selection creates a new migration job using the same parameters as the original job.

Pause Job
This selection saves the current state of the migration and pauses job execution. After resuming, the job continues execution from the point where it was paused without recollecting data.

Stop Job
This selection stops the job, but maintains the ability to resume later. After resuming, the job's internal state is completely reinitialized, and all migration data is recollected.

Delete Job
This selection deletes the job from the database. If the job was running, or pending, it is stopped first and then deleted. This works for all job states (**Pending**, **Running**, **Completed**, and so on).

Resume Job
This selection resumes the job's execution for a job in a **Stopped** or **Paused** status.

Migration Job Status

Based on the migration job lifecycle, and possible user actions, each migration job has one of the following statuses:

Submitted
This status indicates you configured the migration job, but DIVAmigrate did not start working on it yet.

Pending
This status indicates that DIVAmigrate found a new migration job and started processing the job.

Initializing
This status indicates that DIVAmigrate is collecting all necessary information required to start the migration job.

Copying Content
This status indicates that DIVAmigrate is creating copies of instances selected for migration.

Cleaning Up
When the **Move** migration strategy is selected, DIVAmigrate deletes already migrated instances from the migration source. If a disk buffer is used, DIVAmigrate removes the instances from the buffer.

Finalizing
This status indicates that DIVAmigrate is checking the migration results, and updating the migration job status and other job information.

Completed
This status indicates that the migration job was successfully completed.

Partially Completed
This status indicates that the migration job was completed, but some objects were not migrated.

Cancelling
This status indicates that the command to cancel the migration job was received, but has not yet been processed by DIVAmigrate.

Cancelled
This status indicates that you canceled the migration job.

Aborted
This status indicates that the migration job was not completed due to an error.

Pausing
This status indicates that the command to pause the migration job was received, but has not yet been processed by DIVAmigrate.

Paused
This status indicates that the migration job was paused by DIVAmigrate.

Stopping
This status indicates that the command to stop the migration job was received, but has not yet been processed by DIVAmigrate.

Stopped
This status indicates that the migration job was stopped by DIVAmigrate.

Resuming
This status indicates that the command to resume the migration job was received from the client (GUI), but has not yet been processed by DIVAmigrate.

Deleting
This status indicates that you submitted the command to delete the migration job entirely.

Migration Job Parameters
This section describes (in detail) each of the migration job parameters.

Migration Source
This ***Migration Source*** parameter defines the set of instances for migration. You can provide the ***Migration Source*** in any of following ways:

- Barcode for a single tape

- List of Tapes
- Single Tape Group (`mediaName`)
- Single Disk Array (`mediaName`)
- List of Instances (object name, object category, and instance ID)

You must provide valid values for the ***Migration Source*** when using *List of Tapes* or *List of Instances*. DIVAmigrate will not try to validate List of Tapes, List of Instances and the Excluded Instances List before starting the migration job.

The asterisk mask filter is not supported when you use List of Tapes or List of Instances as a ***Migration Source***. The asterisk mask filter is only supported for the Excluded Instance List.

You have the option to provide the Tape List and Instances List in a file. The following rules apply to the file:

- File names can have any name and extension.
- For Tape Lists, each line in the file should contain only one barcode.
- For Instances Lists, each line in the file must contain the object name, object category and instance ID. They must be exactly in this sequence, and you must use the pipe operator as the separator.

  For example, `objectname|category|instanceid`.

Note: Character escaping is supported if the pipe operator is present in the object name or category. For example, you must write `object|one` as `object\|one`.

Migration Destination

This ***Migration Destination*** parameter defines the target media where objects are migrated. The valid ***Migration Destinations*** are **Single Media**, **Multiple Media**, and the **Same as Source** as the ***Migration Source*** (for the **Copy to Source** migration strategy).

The following rules apply to the number of new instances being created on the ***Migration Destination***:

- DIVAmigrate checks how many instances of each object being migrated already exist on the destination, and counts already existing instances as already migrated.

  For example, one instance for *Object-A* already exists on the destination media called `default`. You set the number of instances on the destination media (`default`) to `1`. The migration job performs no additional copy of *Object-A* to the destination media (`default`) because one instance already existed before the migration job started.
- If you select **Multiple Media** for the ***Migration Destination***, you must provide the number of instances that must be present for each media separately.
- If you create a job with multiple destinations, using a combination of both tape groups and disk arrays, the migration service always migrates to the disk destination first.
- If the job has multiple media destinations, and one of the destinations is a disk array, and the source is a single tape, tape group, or tape instance, DIVAmigrate always uses instances migrated to the disk destination as the source for migration

to any tape destinations. If the alternative instance's migration fails, the original source instance is used for migration.

Migration Strategy

The ***Migration Strategy*** parameter defines the type of migration being performed. The following ***Migration Strategy*** options are currently implemented:

Copy

This strategy migrates the configured number of instances from the ***Migration Source*** to the ***Migration Destination***, but does not remove original instances from the source.

Copy to Source

This strategy is the same as the **Copy Strategy**, but the ***Migration Source*** and ***Migration Destination*** are the same. This strategy is applicable only for a **Single Media** source.

Move

This strategy migrates the configured number of instances from the source to the destination, and deletes original instances from the source.

Migration Options

The following Migration Options are currently implemented:

Requests Priority

This parameter identifies the priority of requests sent to the DIVArchive Manager by the DIVAmigrate Service during the migration job. *This is a mandatory parameter*.

Excluded Instances List

This parameter is a list of instances you provide. The instances in this list are excluded from the migration. You have the option to input the list as a file with values separated by the pipe operator, or you can enter them manually. Both the file and manual input process use the following similar conventions:

- The object name and object category support filtering using an asterisk.
- The instance ID can be a number or an asterisk. An asterisk indicates any and all instances.

For example, `objectname|category|instanceid`. *This is an optional parameter*.

Alternative Instance Sources

If you enable this option, DIVAmigrate performs the following processing for each instance it attempts to migrate:

- If the instance is on tape, DIVAmigrate checks for a disk instance of the same object anywhere in the system and uses it instead.
- Alternative sources are selected during the initialization stage of a migration job, and are not updated during the migration job processing.
- The migration tool always migrates the disk instances available on the alternative disk first as part of the migration plan.

DIVAmigrate always considers all disk instances to be online and does not perform an availability check for them. If the disk where the instances are located is broken, the migration of the instance fail and the error is logged.

The original instance is always used if the migration of an alternative instance fails. *This is an optional parameter and is turned off by default*.

Recovery Mode

Recovery Mode is an extension to the ***Alternative Instance Sources*** option, which allows searching for any online instance (both disk and tape) anywhere in the DIVArchive system, and enables implementation of recovery scenarios for damaged offline tapes.

When this option is on, the ***Alternative Instance Source*** option is automatically also on. If the migration operation fails during the migration of an alternative recovery instance, the original source instance is used for migration.

This is an optional parameter and is turned off by default.

Disk Buffer Name

You can specify the disk array name to use as the migration's ***Disk Buffer Name***. Currently, you can only assign one disk array as a disk buffer. Tape objects are initially migrated to the disk buffer from ***Migration Source***. After being migrated to the disk buffer, they will be migrated from disk buffer to the identified ***Migration Destination***.

You configure the maximum percentage of space available for migration on the disk arrays in the Configuration Utility. During the migration process, the maximum used space on the disk buffer is periodically compared to the value of this parameter. If the percentage of used space is greater than, or equal to, the configured value, then the migration job attempts to migrate objects from the disk buffer to the destination. The disk buffer is then cleaned up, and the job continues with the remaining objects being migrated. If, after disk buffer clean up is performed, there is still not enough free space the job will be paused.

The following rules are applied when using the disk buffer:

- The Migration Service does not check the disk buffer availability status before submitting a request.
- If the disk buffer runs out of space during migration job processing, all currently cached objects are migrated from the disk buffer to the ***Migration Destination***, and then cleaned from the disk buffer. After that, the Migration Job continues processing as usual.
- If the ***Alternative Sources*** option is on, and for a tape instance there is alternative disk instance, the instance is not copied to the disk buffer.
- Instances cached to the disk buffer are deleted from it as part of the same migration job.
- DIVAmigrate checks if the disk buffer percentage of used space is less than the configured value for each disk array. If it is more than the configure value, then the migration job is paused.
- The disk buffer is not used for instances that are migrated from a disk source, an alternative disk source, or to a disk destination.

This is an optional parameter.

Scheduled Start Time and Scheduled End Time

You can schedule migration jobs by providing a ***Scheduled Start Time***, a ***Scheduled End Time***. You can provide none, both, or just one of them.

For a ***Scheduled Start Time***, DIVAmigrate guarantees not to start the migration job until this time is reached. DIVAmigrate attempts to start the job as soon as the scheduled time is reached if it has enough available resources.

For a ***Scheduled End Time***, DIVAmigrate guarantees to stop the job processing after this time is reached. A stopped migration job receives the **Paused** status. The ***Scheduled End Time*** parameter is removed from the job after it resumes.

The DIVAmigrate Service validates your input for new migration jobs in the GUI when you create a job, and in the Service when the job is initialized. No job parameter validation is performed for jobs submitted through the command line.

*These are both optional parameter*s.

Basic Migration Jobs

You must use a migration job to change a tape format from Legacy to AXF; repacking a tape will not change the tape format. Repacking existing Legacy format objects retains the format of the tape even if the tape group format was updated in the configuration from Legacy to AXF format.

Copying Data to another Group or Array

The following scenarios describe different possible outcomes when you copy data to another group or array using the DIVAmigrate Utility.

Scenario 1

A series of objects (*Object-A*, *Object-B*, and *Object-C*) are in the *Main Tape Group* and must be duplicated to the *Backup Tape Group*. The following table identifies the initial state and location of the tapes:

Main Tape Group	Backup Tape Group
Object-A	
Object-B	
Object-C	

You submit a copy job to create a copy of *Object-A*, *Object-B*, and *Object-C* to the *Backup Tape Group*. This command results in *Object-A*, *Object-B*, and *Object-C* having instances in both tape groups as follows:

Main Tape Group	Backup Tape Group
Object-A	Object-A
Object-B	Object-B
Object-C	Object-C

Scenario 2

When the initial state includes one or more objects in the *Main Tape Group* already existing in the *Backup Tape Group*, the initial state is as shown in the following table:

Main Tape Group	Backup Tape Group
Object-A	Object-A
Object-B	
Object-C	

You submit a copy job to create a copy of *Object-A*, *Object-B*, and *Object-C* to the *Backup Tape Group*. This command results in *Object-A*, *Object-B*, and *Object-C* having instances in both tape groups as follows:

Main Tape Group	Backup Tape Group
Object-A	Object-A
Object-B	Object-B
Object-C	Object-C

The result is one instance of each object in *Backup Tape Group*.

There are many possible reasons for *Object-A* already being in the *Backup Tape Group*, including (but not limited to) the following:

- A user manually copied the object to the *Backup Tape Group*.
- At one time, a user possibly configured the Oracle DIVArchive Storage Plan Manager to make copies to the *Backup Tape Group*, and then either disabled the corresponding slot or modified the configuration.
- The DIVAmigrate Utility was requested to duplicate objects, and then the process was interrupted before it completed.

In production environments, it is not generally known how many object instances currently exist in the target group (possibly none). However, it is typically known exactly how many are desired. In the previous example, the desired result is to have one instance in the *Backup Tape Group* for each *Main Tape Group* object after the duplication run. The DIVAmigrate Utility was designed to perform in this manner. The utility expects you to specify the desired number of instances in the target group or array. Depending on the actual situation, DIVAmigrate only performs the copy operations required to achieve the desired result.

Moving Data to another Group or Array

You will eventually encounter scenarios where moving the data is required, rather than replicating it. For example, this is useful when migrating data to a new technology (for example, from LTO-3 to LTO-5), and it is not necessary to keep the old tape instances.

In the following scenario, a new *Tape Group Main-LTO-5* was created and assigned a Tape Set ID populated with LTO-5 tapes. The following table displays the initial state:

Main Tape Group	Tape Group Main-LTO-5
Object-A	
Object-B	
Object-C	

You use the following command to instruct DIVAmigrate to perform the job correctly:

```
client {object list} -media Main-LTO-5 -count 1 -delete
```

This command includes the `-delete` option, which tells DIVAmigrate to delete the source instances after copying them. In the example, the instances of *Object-A*, *Object-B*, and *Object-C* located in *Main Tape Group* are removed.

DIVAmigrate uses the following process for each object to complete this job:

1. Copy the object to the destination *Tape Group Main-LTO-5*.
2. Run `DeleteInstance` on *Main Tape Group*.
3. Mark the process complete, and repeat the process for the next object in the list.

This sequence is repeated until all objects have been moved. The final state is as follows:

Main Tape Group	Tape Group Main-LTO-5
	Object-A Object-B Object-C

The instances were moved to the new tape group. However, they are essentially different instances of the objects because the original instances were actually copied to the new tape group, and then deleted from the source tape group.

Each time DIVArchive copies an object, a new instance of that object is created, and it is identified by a new Instance ID. In this scenario, the instance of *Object-A* located in the new *Tape Group Main-LTO-5* has a different Instance ID than the instance that was located in *Main Tape Group*. Because the new instance is a copy of the same object, the data is the same, and the data was actually moved.

The following table indicates the possible Instance ID for each object. The instances deleted after the run are shown in the *Main Tape Group*, and marked deleted in the table. The *Main Tape Group* still exists, but the instances in it no longer exist because they were deleted.

Main Tape Group	Tape Group Main-LTO-5
Object-A, InstanceID 1 (deleted)	Object-A, Instance ID 2
Object-B, Instance ID 1 (deleted)	Object-B, Instance ID 2
Object-C, Instance ID 1 (deleted)	Object-C, Instance ID 2

Copying and Migrating Data to the same Group or Array

Sometimes moving or replicating data to the same group or array as the one containing the source instances is required. For example, if you are performing a tape technology migration, and you want to use the existing tape group name for instances stored on the new technology tapes. The old and new technology tapes are placed in the same `SetID` and the same group will contain both tape types. Oracle recommends setting the old type tapes to *Not Writable* (or *Protected*) to force DIVArchive to write new data to the new technology tape type.

You may find it efficient to create a group that only contains the desired new technology tapes and use that as the ***Destination Media***.

The following is an example of migrating objects from LTO-3 to LTO-5 tapes in the same *Main Tape Group*. The table shows the initial state of the objects.

LTO-3 Tapes	LTO-5 Tapes
Object-A, Instance ID 1	
Object-B, Instance ID 1	
Object-C, Instance ID 1	

You use the following command for replicating the data:

```
client -tapelist tapelist.txt -media Main -count 2
```

- The `tapelist.txt` parameter is a file listing the barcodes of the tapes being migrated. If you are performing a technology migration, the file will contain all LTO-3 barcodes of *Main Tape Group*. Oracle Support can provide you with tools to generate the listings automatically.
- The `-media Main` parameter tells DIVAmigrate to copy the objects to *Main Tape Group*, which is the same group as the source data.
- The `-count 2` parameter is important because DIVAmigrate is replicating to the same ***Destination Group*** from the same ***Source Group***. Therefore, two instances of each object after the job completes are present instead of one.

If you specify `-count 1` instead of `-count 2`, DIVAmigrate verifies that one instance exists for each object. Because this is already the case, DIVAmigrate considers the job complete and does nothing. This is a good way to perform a dry run with the DIVAmigrate Utility for a quick sanity check.

This is the final outcome after the job completes:

LTO-3 Tapes	LTO-5 Tapes
Object-A, Instance ID 1	Object-A, Instance ID 2
Object-B, Instance ID 1	Object-B, Instance ID 2
Object-C, Instance ID 1	Object-C, Instance ID 2

You can use the following command to move the data instead of replicating it:

```
client -tapelist tapelist.txt -media Main -count 1 -delete
```

- The `-delete` parameter tells DIVAmigrate to delete the original instances after they have been copied.
- The `-count 1` parameter tells DIVAmigrate that there must be only one instance of each object after the job completes.

Executing the previous command results in the following final state:

LTO-3 Tapes	LTO-5 Tapes
	Object-A, Instance ID 2
	Object-B, Instance ID 2
	Object-C, Instance ID 2

Because DIVAmigrate can skip unnecessary tasks, you can use the previous two example commands one after the other with the same results. The first command replicates the data, and the second command only performs the deletes. DIVAmigrate recognizes that the copies are already completed and skips that step in the second command.

You can use this two-step migration if you do not initially have faith in your new technology tape drives. If you prefer to keep the old instances temporarily, for data safety, use the first (replicate) command as a first step. You only run the first command until you have been using the new technology tapes and drives for a while to save a copy of the old objects. When you are satisfied with their functionality and stability, you can then use the second (move) command to delete the old instances.

Stopping and Resuming Jobs

The DIVAmigrate Utility only performs the steps necessary to reach the intended goal, and nothing more. Any time a migration job is started, DIVAmigrate examines the DIVArchive catalog and calculates which copy and delete operations remain incomplete. It does not matter to DIVAmigrate if the job was already partially completed during a previous run, it will always adapt to the current state without the need to store any progress in a status or log file.

For this reason, you can freely interrupt a DIVAmigrate job at any time and resume it later. There are no specific actions to complete before resuming an interrupted job.

Advanced Migration Jobs

This section describes advanced migration operations.

Speeding up Tape to Tape Migration using a Disk Buffer

The DIVAmigrate Utility processes objects sequentially. For example, if an object is copied from a *Tape-1* to a *Tape-2*, a Copy request is sent to the Manager. The Manager allocates some *Cache Disk* space to store the object and proceeds in two sequential steps:

1. Read the object from *Tape-1* and write it to the *Cache Disk*.
2. Read the object from the *Cache Disk* and write it to *Tape-2*.

DIVAmigrate also processes series of objects. For example, migrating a tape volume to another tape group. DIVAmigrate uses the following sequential procedure:

1. Read *Object-A* from *Tape-1* and write it to the *Cache Disk*.
2. Read *Object-A* from the *Cache Disk* and write it to *Tape-2*.
3. Read *Object-B* from *Tape-1* and write it to the *Cache Disk*.
4. Read *Object-B* from the *Cache Disk* and write it to *Tape-2*.

This process continues until all objects on the tape volume have been processed.

You can possibly write the data to more than one tape in the ***Destination Tape Group*** based on the activity and Manager workflow. However, for this example consider that all objects go to a single *Tape-2*.

During Step 2 in the previous sequence, the source drive is idle and you can assign it to other requests. So *Tape-1* might dismount and another tape mounts. If this occurs, you must mount *Tape-1* and position it again in Step 3. This will probably require dismounting a tape from the source drive before you remount *Tape-1*. This causes a substantial delay in operations. While Step 3 proceeds, *Tape-2* might get dismounted because the destination drive is now idle. Therefore, you may need to be mounted and positioned again in *Step-4*.

A large migration using tape to tape copies can result in tremendous dismount, mount, and positioning overhead. Oracle recommends that heavy migration operations are conducted on dedicated Oracle DIVArchive Actors with dedicated tape drives. See Appendix A for Actor licensing information.

Even with dedicated tape drives and no mount and dismount overhead, you might observed that one drive is idle while the other is working. For example, in the previous sample sequence, the target drive is idle in Step 1 and Step 3, and the source drive is idle in Step 2 and Step 4.

To suppress drive idle time, you can use a disk array as a buffer for the migration job. When a *Disk Buffer* is specified, DIVAmigrate first enters a *Read Phase*. During this phase, it continuously reads the source tape and stacks as many objects as space allows in the designated array. It then switches to the *Write Phase*, where it continuously writes objects from the disk to the destination tape.

Caution: You must choose the array used as a disk buffer carefully. There are two very important rules to consider:

- Make sure that the array is not managed by the Storage Plan Manager. When SPM manages an array, the data DIVAmigrate stores there might be replicated to tape and deleted automatically. This hinders DIVAmigrate operations, and ends up with undesired tape instances, resulting in a waste of tape storage space.
- Make sure the array you use is not a repository containing resident instances of the objects being migrated. If it is, DIVAmigrate considers those instances to be *buffered instances* from a previous job, and they are deleted after the job completes.

The obvious benefits are that only one drive is used in each phase, and it is used continuously. There is still a possibility that the drive will be dismounted between two Copy requests and assigned to a more prioritized request from the Manager queue. However, this will not happen if the drives and Actors are dedicated to the migration.

There are recognizable similarities between this workflow and the way the Repack workflow functions. The difference is that DIVAmigrate creates standard object instances on a disk array, and must explicitly delete them after the job. Repack workflows create temporary instances on a cache disk that are cleaned by the Manager after the request. You must restart a terminated migration job using the *exact same command line*, and specifically, the *exact same buffer array* you used in the original job. If you change arrays between two runs, the second run overlooks the buffered instances from the first run, and recreates new instances in the new array resulting in:

- Wasted time re-creating disk instances that were already created on the previous array.
- Wasted space in the previous array because the instances created there are no longer managed by a migration job, and are never deleted.

The following is a sample migration command using a *Disk Buffer*:

```
client -tape 000001 -media Main-LTO-5 -count 1 -buffer NAS001
```

Sample Scenario

This scenario is an example using the previous command line where *Tape 000001* is an LTO-3 tape from the *Main Group*, that must be migrated to *Tape Group Main-LTO-5* using the *NAS001* array as a buffer.

Tape 000001 contains *Object-A* and *Object-B*. DIVAmigrate performs the following action sequence, assuming the objects do not have instances in the new tape group:

1. Copy *Object-A* from *Tape 000001* to *Array NAS001*.
2. Copy *Object-B* from *Tape 000001* to *Array NAS001*.
3. Copy *Object-A* from *Array NAS001* to a tape in *Tape Group Main-LTO-5*.
4. Delete the *Object-A* instance on *Array NAS001*.
5. Copy *Object-B* from *Array NAS001* to a tape in *Tape Group Main-LTO-5*.
6. Delete the *Object-B* instance on *Array NAS001*.

According to this workflow, instances are copied to the buffer in a single stream. This way the source drive is kept busy. Each buffered instance is discarded as soon as it is no longer needed. That is, after the target copies of the related object are complete.

To delete instances from the source tape after migration, you add the `-delete` option as follows:

```
client -tape 000001 -media Main-LTO-5 -count 1 -buffer NAS001 -delete
```

This is the corresponding action sequence:

1. Copy *Object-A* from *Tape 000001* to *Array NAS001*.
2. Copy *Object-B* from *Tape 000001* to *Array NAS001*.
3. Copy *Object-A* from *Array NAS001* to a tape in *Tape Group Main-LTO-5*.
4. Delete the *Object-A* instance on *Array NAS001*.
5. Delete the *Object-A* instance on *Tape 000001*.
6. Copy *Object-B* from *Array NAS001* to a tape in *Tape Group Main-LTO-5*.
7. Delete the *Object-B* instance on *Array NAS001*.
8. Delete the *Object-B* instance on *Tape 000001*.

The workflow is the same as the previous workflow, except that this time, after deleting the buffered instance, the source instance is also deleted. If no errors were encountered, *Tape 000001* is empty after the migration job.

Creating Multiple Instances in the Destination Group or Array

DIVAmigrate can create any number of instances in the target group or array using the `-count` option. DIVAmigrate performs as many copies as necessary to reach the specified count.

DIVArchive never stores two instances of the same object on the same volume or disk. Therefore, if a count greater than `1` is specified, the ***Destination Tape Group*** or ***Destination Disk Array*** must have more than one tape volume or disk identified within it.

If a *Disk Buffer* is used (`-buffer` option), the instances are buffered only once regardless of how many instances are being created in the ***Destination Tape Group*** or ***Destination Disk Array***.

Assuming that *Backup* is a tape group, *Tape 000001* contains *Object-A* and *Object-B*, and these objects do not have instances in *Tape Group Backup*, the following command translates into the following sequence:

```
client -tape 000001 -buffer NAS001 -media Backup -count 2
```

1. Copy *Object-A* from *Tape 000001* to *Array NAS001*.
2. Copy *Object-B* from *Tape 000001* to *Array NAS001*.
3. Copy *Object-A* from *Array NAS001* to a tape in *Tape Group Backup*.
4. Copy *Object-B* from *Array NAS001* to a tape in *Tape Group Backup*.
5. Copy *Object-A* from *Array NAS001* to another tape in *Tape Group Backup*.
6. Delete the *Object-A* instance on *Array NAS001*.
7. Copy *Object-B* from *Array NAS001* to another tape in *Tape Group Backup*.
8. Delete the *Object-B* instance on *Array NAS001*.

The two instances of each object in the backup group were created separately. DIVAmigrate performs the first copy of *Object-A* and *Object-B*, then the second copy, instead of creating two copies of *Object-A* sequentially, then two copies of *Object-B*. This is because there is only one tape drive to use. If the same object is copied twice to the ***Destination Tape Group***, the Manager must use a different tape for each instance because DIVArchive does not allow two instances of the same object to be stored on the same volume. This causes a dismount and mount operation.

By performing the first copy of *Object-A* and *Object-B* sequentially, you can keep the same tape on the drive for both objects. You must only change the tapes when entering the second copy phase. DIVAmigrate refers to this as a *Migration Step* in its logs. There are two migration steps, a first copy to the *Backup Tape Group*, and then a second copy to the same group. DIVAmigrate always attempts to process as many objects as possible sequentially within a particular migration step. This minimizes the mount and dismount overhead.

The Migrate Utility does not remove extraneous instances in the ***Destination Tape Group*** or ***Destination Disk Array***. If one instance of each object is requested in the ***Destination Tape Group*** or ***Destination Disk Array***, and some objects already have two or more existing instances, DIVAmigrate considers the job complete for these objects because there is at least one instance. It will not delete some existing instances to reduce the count back to one.

Migrating to Multiple Destination Groups or Arrays

You can use more than one `-media` option for DIVAmigrate to copy data to more than one group or array. Each `-media` option must be accompanied by a `-count` option, and the default count is `1`. For example:

```
client -tape 000001 -media Main -count 1 -media Backup -count 2 -delete
```

If a *Disk Buffer* is used (`-buffer` option) the instances are buffered only once, regardless of how many ***Destination Tape Groups*** or ***Destination Disk Arrays*** are specified.

Assuming *Backup* and *Ext* are tape groups, *Tape 000001* contains *Object-A* and *Object-B*, and these objects do not have instances in any ***Destination Group***, the following command translates into the following sequence:

```
client -tape 000001 -media Backup -count 2 -media Ext -count 1 -buffer NAS001
```

1. Copy *Object-A* from *Tape 000001* to *Array NAS001*.

2. Copy *Object-B* from *Tape 000001* to *Array NAS001*.
3. Copy *Object-A* from *Array NAS001* to a tape in *Tape Group Backup*.
4. Copy *Object-B* from *Array NAS001* to a tape in *Tape Group Backup*.
5. Copy *Object-A* from *Array NAS001* to another tape in *Tape Group Backup*.
6. Copy *Object-B* from *Array NAS001* to another tape in *Tape Group Backup*.
7. Copy *Object-A* from *Array NAS001* to a tape in *Tape Group Ext*.
8. Delete the *Object-A* instance on *Array NAS001*.
9. Copy *Object-B* from *Array NAS001* to a tape in *Tape Group Ext*.
10. Delete the *Object-B* instance on *Array NAS001*.

As explained in the previous section, DIVAmigrate divides the sequence in several individual migration steps. There are three in the previous example:

- A first copy to *Backup*.
- A second copy to *Backup*.
- A copy to *Ext*.

For each migration step, DIVAmigrate processes all objects in a row to minimize the amount of mount and dismount overhead. If this is not done, it would require three tape changes per object:

- One for the second copy to the same group.
- One for the copy to a different group.
- One for the switch back to the initial group for the next object.

Default Destination Instance Count

Omitting the destination instance count option (`-count`) results in DIVAmigrate using a default value of `1`. The following examples assume migration of the data from *Tape 000001* belonging to *Tape Group Main*:

`client -tape 000001 -media Backup`
The destination count is set to `1` by default. This command tells DIVAmigrate to create an instance of each object from *Tape 000001* in *Tape Group Backup*; replicating the data on *Tape 000001*.

`client -tape 000001 -media Main`
This command creates an instance of each object from *Tape 000001* in *Tape Group Main*. However, this is the same group as the source tape so there is already an instance of each object in that group (the one on *Tape 000001*). DIVAmigrate considers the job already complete and does nothing more. To force the duplication of *Tape 000001* to the same group, the `-count 2` option must be added.

`client -tape 000001 -media Main -delete`
This command creates an instance of each object from *Tape 000001* in *Tape Group Main*. This is the same group as the source tape, but this time DIVAmigrate is told to delete the source instances. Therefore, DIVAmigrate cannot include the source instance in the final count as it did in the previous example. Instead, DIVAmigrate creates an additional instance of each object in *Tape Group Main* before deleting the original instance on *Tape 000001*. The result is similar to a Repack operation.

Repacking Tapes

You must use a migration job to change a tape format from Legacy to AXF. Repacking a tape will not change the tape format. Repacking of existing Legacy format objects retains the format of the tape even if the tape group format was updated in the configuration from Legacy to AXF.

To repack a suspected corrupt tape, using DIVAmigrate may be more effective than executing a Repack operation because it does not halt on errors. For example, the following command will mimic a Repack of a *Tape 000001* belonging to *Tape Group Main*:

```
client -tape 000001 -media Main -count 1 -delete -recovery
```

The `-delete` option tells DIVAmigrate to move the data. That is, copy it to the *Destination Group* and then delete the original instance.

The `-count 1` option tells DIVAmigrate that after the migration completes, there should only be one instance of each object in *Tape Group Main* remaining.

The `-recovery` option causes DIVAmigrate to look for, and use if available, any alternative online instances either on disk or tape. You use this option if the source tape is known to be corrupt. If no alternative is found, DIVAmigrate still attempts to use the instance from the source tape.

The DIVAmigrate Utility copies each source instance to the same group, and then deletes the source instance. This is similar to a Repack operation.

There are, however, a few differences with Repack operations:

- Repack is a single request in the Manager queue. An equivalent DIVAmigrate job produces numerous Copy and DeleteInstance requests.
- Repack moves instance elements (file segments), not necessarily whole instances. If an instance is in one piece on the repacked tape, it will be moved in it's entirety. However, if the instance is spanned, Repack only moves the portion of the instance that resides on the repacked tape; not the other portions residing on other tapes. DIVAmigrate moves the whole instance because it uses the standard Copy function. This makes DIVAmigrate a useful tool to despan instances.
- Repack always writes the source tape content to a single destination tape. DIVAmigrate uses the standard Copy function, and the only guarantee is that the data will end up on the requested destination group. However, not necessarily on a single tape.

Despanning Instances

A spanned instance is an object instance that is written across multiple tapes (usually two). Spanned instances require additional tape mount operations during reading, which can cause issues in some contexts. For example, direct restore from tape may experience a timeout on the Source/Destination during tape remounting.

You sue the following command to despan a series of instances:

```
client -instlist file.txt -media Main -delete
```

Oracle Support can provide you with a utility to generate an instance list file containing the list of instances spanned in the system.

Due to the high capacity capabilities of modern tape units, they tend to suffer from random reduced capacity from intermittent bad block writes. Therefore, the EOT (End of Tape) marker may be prematurely encountered. This defeats DIVArchive's remaining tape space calculations and causes accidental spanning.

Using Alternate Source Instances

DIVAmigrate always defaults to using the specified source instances to perform the destination copies, or the buffer copy if a *Disk Buffer* is used.

- If an *Object* or *Object List* is used as the source specifier (`-tape` or `-tapelist` option respectively), DIVAmigrate uses the instances located on these tapes.
- If an *Instance List* is used (`-instlist` option), DIVAmigrate uses the instances listed in the file.
- If an *Object List* is used (`-tapelist` option), DIVAmigrate identifies an optimal instance list and uses that instance.

There are situations where allowing DIVArchive to choose the source instance is desirable. For example, if you are migrating data off of a suspicious tape and some objects possibly still have instances on a storage disk. By default, DIVAmigrate will read the source instances from tape. If you want the Manager (rather than DIVAmigrate) to choose the source instance (which will select an existing disk instance), you use the `-autosrc` option as follows:

```
client -tape 000001 -media Main -delete -autosrc
```

The `-delete` option always deletes the source instances obtained from the source specifier in the command line (`-tape`, `-tapelist`, or `-instlist`), regardless of which particular object instance was actually selected as a copy source by the Manager. For example, if you specify `-tape 000001` on the command line with the `-autosrc` option, and one of the objects has a disk instance, this object is copied from disk not from tape. However, the instance deleted after the run is still the one located on tape `000001`.

Excluding Objects from Migration

If you observe repeated read errors on some source instances, and you do not want to delete them at this point, you can exclude them from the migration process using the `-excl` option as follows:

```
client -tape 000001 -media Main -delete -excl file.txt
```

This command ignores all instances belonging to the objects listed in the file named `file.txt`. This plain text file lists objects to be excluded in the form `object|category`, one entry per line. For example:

```
Clip001|HD
Clip002|SD
```

4 Error Handling and Failure Scenarios

This chapter describes error handling and possible failures that may occur during migration, and includes the following information:

- Migration Error Handling
- Migration Failure Scenarios

Migration Error Handling

DIVAmigrate performs error handling based on following process:

1. If the configured number of request failures occur sequentially, the migration job attempts to switch to another ***Migration Destination***. The job is paused if there are no other destinations.
2. After failing for the configured number of request attempts for a destination, the migration job switches to another destination. DIVAmigrate then attempts to process the job using the new destination until the configured number of requests fails sequentially. If this attempt also fails, DIVAmigrate attempts to switch to the next destination (if any exist). After all destinations have been exhausted, the job will pause (if no more destinations exist).

   The only exception to this rule is failed requests due to a too many running requests error; these requests are always retried.

In general DIVAmigrate does not retry migration for objects. DIVAmigrate stores the error codes for each object, and moves on to next object requiring migration. You can rerun the migration job later, and it only affects new objects on the source, and those that failed during the previous run; objects successfully migrated the first time are not retried when you rerun the failed job.

Migration Failure Scenarios

This section describes specific failure scenarios, actions by DIVAmigrate to resolve the failure, and suggested actions for you.

Table 4–1 DIVAmigrate Failure Scenarios and Resolutions

Failure	DIVAmigrate Action	Suggested User Action
DIVAmigrate lost the connection to the DIVArchive Manager.	Tries to reconnect until connection is restored.	If the Manager is offline or not running, then restart the Manager. If DIVAmigrate cannot connect to the Manager, you can identify it because new migration jobs stay too long in the **Submitted** state, and do not change to the **Pending** state. You can also check the DIVAmigrate Logs. At this point you should investigate common problems with the connection.
DIVAmigrate cannot connect to the Manager Database.	Retry the connection three times every 10 seconds; if not successful then quit.	You must investigate and resolve common database connectivity issues, and then restart the DIVAmigrate service.
The environment where DIVAmigrate is running fails.	After restart, DIVAmigrate checks the status of each job in the database. If they are not expired, it continues their processing. Otherwise, DIVAmigrate recreates their migration plan and resumes job processing as soon as free resources are available.	You must resolve the operating system environment issue, and then restart DIVAmigrate.
There are no online instances for an object on the ***Migration Source***.	DIVAmigrate checks for online instances only during the *Initialization* phase. If an object does not have any online instances, DIVAmigrate skips it and assign the correct error code to the object in the database. If the object was online, but later became offline during the *Initialization* phase, then some requests sent to the Manager will fail. The error code is stored in the database, and the request is not retried. If the requests continuously fail up to the maximum number of failures, the job pauses. If the job has multiple sources or destinations, it switches to the next source or destination. If alternative instances migrations fail, the original instances are used instead.	You can browse the results of migration for each completed migration job. The results contain a list of all objects eligible for migration, and resulting error codes (if any errors occurred). You should make objects accessible online again, and then rerun the migration job.

Table 4–1 (Cont.) DIVAmigrate Failure Scenarios and Resolutions

Failure	DIVAmigrate Action	Suggested User Action
You perform a migration from a tape group that has offline tapes.	If the ***Alternative Sources*** option is turned on, DIVAmigrate attempts to locate disk instances for all offline instances, and then migrates them. If ***Recovery Mode*** is turned on, DIVAmigrate attempts to locate disk or tape instances for offline source instances, and then migrates them. If no online alternative instances are found for some offline instances, they are skipped from migration. An *Object Offline* error code is assigned to the objects. If the tape transitions to offline during migration, the job pauses after the maximum sequential failure requests. If the job has multiple sources or destinations, it switches to the next one, and then pauses after processing them.	You can browse the results of migration for each completed migration job. The results contain a list of all objects eligible for migration, and error codes (if any errors occurred). You must make objects accessible online again, and then rerun the migration job.
You submitted the migration job with invalid parameters.	DIVAmigrate assigns the **Aborted** status to a job.	You must confirm the job parameters, and then create a new migration job with correct parameters.
A user deletes unmigrated objects from the source during a migration procedure.	DIVAmigrate attempts to migrate the objects, and stores the appropriate error codes in the database.	You should use the DIVArchive Control GUI to **Cancel** or **Delete** the migration job.
You realize that you submitted a migration job with incorrect parameters.	Not applicable	You should use the DIVArchive Control GUI to **Cancel** or **Delete** the migration job.
There is not enough free space on the Destination.	DIVAmigrate fails the configured maximum number of failed requests sequentially, and then places the job in the **Paused** state.	You must resolve the issue on the destination, and then resume the job.
All ***Migration Destination*** media goes offline.	DIVAmigrate fails the configured maximum number of failed requests sequentially, and then places the job in the **Paused** state.	You must resolve the issue on the destination, and then resume the job.
The *Source Tape List*, *Instance List*, or *Exclude List* has invalid values, but the syntax is correct. For example, a tape barcode is invalid.	DIVAmigrate ignores invalid values. If all of source is invalid, the job must terminate.	Confirm the values entered into these lists, make any necessary corrections, and then rerun the job.

A

DIVArchive Options and Licensing

The following table identifies DIVArchive options and licensing metrics.

Part Number	Description	Licensing Metric
L101163	Oracle DIVArchive Nearline Capacity	Per TB
L101164	Oracle DIVArchive Archive Capacity	Per Slot
L101165	Oracle DIVArchive Actor	Per Server
L101166	Oracle DIVArchive Manager	Per Server
L101167	Oracle DIVArchive Partial File Restore	Per Wrapper
L101168	Oracle DIVArchive Avid Connectivity	Per Server
L101169	Oracle DIVArchive Application Filtering	Per Server
L101170	Oracle DIVArchive Storage Plan Manager (2 storage plans are included with a DIVArchive Manager License)	Per Server
L101171	Oracle DIVAnet	Per Server
L101172	Oracle DIVAdirector	Per User
L101918	Oracle DIVArchive Export / Import	Per Server
L101919	Oracle DIVArchive Additional Archive Robotic System	Per Tape Library
L101920	Oracle DIVArchive Automatic Data Migration	Per Server

B Default Configuration File

This appendix is a copy of the default configuration file delivered with DIVAmigrate.

```
# Name for Windows Service
# This is mandatory parameter.
SERVICE_NAME=DIVAmigrate

# Host name or IP Address of DIVA Manager
# Default value is 127.0.0.1.
# This is mandatory parameter.
DIVAMANAGER_HOST=127.0.0.1

# Port used to connect to DIVA Manager
# Default value is 9000.
# This is mandatory parameter.
DIVAMANAGER_PORT=9000

# Port used to connect to DIVA Migrate Service
# Default value is 9191.
# This is mandatory parameter
DIVA_MIGRATE_MANAGEMENT_PORT=9191

##### Database connection details. #####
# A URL to connect to DB instead of using DIVAMANAGER_DBHOST and
# DIVAMANAGER_DBPORT or DIVAMANAGER_TNSNAME. It gives the additional
# flexibility to the user to explicitly mention if they want to use
# jdbc thin driver or jdbc oci driver to connect to oracle database.
# examples:jdbc:oracle:thin:@host:port:oracle_sid
# jdbc:oracle:oci:@tnsname
# This is not a mandatory parameter.
DIVAMANAGER_DBURL=

# User Name the DIVArchive Manager uses to connect to the DIVArchive
# Database. This is case sensitive.
# Default value is diva.
# This is a mandatory parameter.
DIVAMANAGER_DBUSER=diva

# TNS Name of the DIVArchive Schema within the Oracle database.
# This setting is ignored if the DIVAMANAGER_DBHOST and
# DIVAMANAGER_DBPORT settings below are defined.
# There must be a corresponding entry in TNSNAMES.ORA found under the
# oracle 11 client installation.
# This is not a mandatory parameter.
DIVAMANAGER_TNSNAME=

# This specifies the Hostname or IP Address of the machine containing
# the DIVArchive Database.
# If using a hostname, this must be present in the hosts file on the
# machine where
# the DIVArchive Manager is installed.
# examples: 127.0.0.1, localhost
# This is not a mandatory parameter.
DIVAMANAGER_DBHOST=

# The Oracle Listener port configured during the DIVArchive Database
# installation.
# Default value is 1521.
# This is not a mandatory parameter.
DIVAMANAGER_DBPORT=1521

# The DIVArchive Database Instance Identifier (SID) in Oracle where DIVArchive
Manager connects.
# Typically lib5.world in most DIVArchive installations. Consult your delivery
plan if you are not sure.
# Default value is lib5.world
# THIS VALUE OR DIVAMANAGER_DBSID MUST BE SET, when DIVAMANAGER_DBHOST &
DIVAMANAGER_DBPORT is used for Database connections. IF BOTH ARE SET, SERVICENAME
TAKES PRECEDENCE OVER SID.
DIVAMANAGER_DBSERVICENAME=lib5.world

# Max simultaneous DIVA Manager requests run by DIVAmigrate.
# Default value is 15.
# This is not mandatory parameter.
MAX_SIMULTANEOUS_REQUESTS=15
# DB scan periodicity (seconds). Determines how often DIVAmigrate
# looks for new jobs in Database.
# This is not mandatory parameter.
# Default value is 60 seconds
DB_SCAN_PERIODICITY=60

# Time in seconds between reconnect attempts in case if connection to
# Manager was lost.
DIVA_RECONNECT_PERIODICITY=30
# Max requests failed in a row before Pause
# DIVAmigrate will always pause if configured number of requests fails
# sequentially.
# Default: 10
MAX_FAILED_REQUESTS_PAUSE=10

# Migration plan max inactive time
# If after service is restarted it finds running jobs with last access
# time greater then this parameter, then migration plan for such jobs
# will be recreated. If there are multiple destination job will switch
# to next destination.
# Default: 24 hours
JOB_MAX_INACTIVE_TIME=24

##### Windows Service Wrapper Configuration Section (not included in this example)
#####

#********************************************************************
# DIVA Migrate Logging
#********************************************************************
##### Logging #####
LOGGING_DIRECTORY=../../log/migrate/
```

```
# Levels can be: DEBUG, INFO, WARN, ERROR, FATAL
# The default value is INFO.
LOGGING_ROOT_LEVEL=INFO
LOGGING_TRACE_LEVEL=INFO
LOGGING_SERVICE_LEVEL=INFO

# File size should be specified using the convention: #KB|MB
# The default value is 10MB.
LOGGING_MAXFILESIZE=10MB

# All files (trace, service and .zip) older than this will be removed
# hours
# The default value is 50.
LOGGING_LIFETIME=50
#*******************************************************************
# DIVA Service Options
# The following service parameters should not be changed without
# the consent of Oracle Support.
#*******************************************************************

# Level can be: DEBUG, INFO, STATUS, ERROR, NONE
# To include thread dumps, set to DEBUG or INFO.
# The default value is INFO.
wrapper.logfile.loglevel=INFO
# The maximum size to allow Wrapper log files to reach before
# rolling. File size should be specified using the convention: #k|m
# The default value is 1m.
wrapper.logfile.maxsize=1m

# Mode in which the service is installed. AUTO_START starts the
# service automatically when the system is rebooted. DEMAND_START
# which requires that the service be started manually.
# The default value is AUTO_START.
wrapper.ntservice.starttype=AUTO_START

# Time without CPU before JVM will issue warning and extend timeout
# (in sec). Timeout will be extended by a few seconds at least once
# before the service shuts down.
#wrapper.cpu.timeout=30

# Number of seconds to allow between the time that the Wrapper
# launches the JVM process and the time that the JVM side of the
# Wrapper responds that the application has started.
# The default value is 60.
wrapper.startup.timeout=60

# Number of seconds to allow between the wrapper pinging the JVM
# and the response
# The default value is 60.
wrapper.ping.timeout=60

# Number of seconds to allow between the time that the Wrapper asks
# the JVM to shutdown and the time that the JVM side of the Wrapper
# responds that it is stopping.
# The default value is 60.
wrapper.shutdown.timeout=60

# Java Library Path (Add location of OCI driver ex: ocijdbc11.dll if using
DIVAMANAGER_TNSNAME.
# Ex: wrapper.java.library.path=.;C:\app\oracle\product\11.1.0)
# The default value is .
#wrapper.java.library.path=../lib/.
```