

Oracle® DIVArchive

DIVAmigrate Embedded Utility User's Guide

Release 7.3

E64132-02

December 2015

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Primary Author: Lou Bonaventura

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Table of Contents

1	INTRODUCTION	1
1.1	DOCUMENT PURPOSE AND SCOPE	1
1.2	DOCUMENT CONVENTIONS.....	1
1.3	OVERVIEW	2
1.4	SUPPORTED ENVIRONMENTS	2
2	INSTALLATION AND CONFIGURATION	3
2.1	DIVAMIGRATE SERVICE CONFIGURATION.....	4
2.2	LOGGING.....	7
3	OPERATIONS	8
3.1	STARTING AND STOPPING THE SERVICE	8
3.2	MIGRATION JOB COMMAND LINE SYNTAX	9
3.3	DIVAMIGRATE GUI	13
3.3.1	DIVAmigrate Wizard	13
3.3.2	DIVAmigrate Panel	17
3.4	MIGRATION JOBS	19
3.4.1	Migration Job Definitions.....	19
3.4.2	Migration Job Actions.....	19
3.4.3	Migration Job Status	20
3.4.4	Migration Job Parameters.....	21
3.4.4.1	Migration Source.....	21
3.4.4.2	Migration Destination	22
3.4.4.3	Migration Strategy.....	22
3.4.4.4	Migration Options.....	23
3.5	BASIC MIGRATION	26
3.5.1	Copying Data to another Group or Array	26
3.5.2	Moving Data to another Group or Array.....	28
3.5.3	Copying and Migrating Data to the Same Group or Array.....	29
3.5.4	Stopping and Resuming Jobs.....	31
3.6	ADVANCED MIGRATION	32
3.6.1	Speeding Up Tape-To-Tape Migration Using a Disk Buffer	32
3.6.2	Creating Multiple Instances in the Target Group or Array.....	35
3.6.3	Migrating to Multiple Groups or Arrays.....	36
3.6.4	Default Target Instance Count	37
3.6.5	Repacking Tapes.....	38
3.6.6	De-spanning Instances	39
3.6.7	Using Alternate Source Instances.....	39

3.6.8	<i>Excluding Objects from Migration</i>	40
4	MIGRATION ERROR HANDLING AND FAILURE SCENARIOS	41
	APPENDIX	45
A1	<i>DEFAULT CONFIGURATION</i>	45

Tables Index

Table 1: System Component Icons for Non-Screenshot Workflows.....	1
Table 2: DIVAmigrate Configuration Parameters.....	4
Table 3: DIVAmigrate Logging Parameters (in <code>migrate.conf</code>).....	7
Table 4: DIVAmigrate Commands and Options	8
Table 5: Command Line Options.....	10
Table 6: DIVAmigrate Failure Scenarios	41

Figures Index

Figure 1: Action Menu – Migrate Actions – Create Migrate Job	13
Figure 2: First Dialog - Select Migration Strategy	14
Figure 3: Select the Migration Source Type.....	14
Figure 4: Select the Migration Source	15
Figure 5: Select Media and Number of Instances	15
Figure 6: Add Additional Migration Destinations	15
Figure 7: Selecting the Migration Job Options	16
Figure 8: Completing the Wizard	17
Figure 9: Scenario #1 - Initial State	26
Figure 10: Scenario #1 – Full Duplication Complete.....	26
Figure 11: Scenario #2 – Initial State	27
Figure 12: Scenario #2 – Full Duplication Completed.....	27
Figure 13: Scenario #1 - Initial State	28
Figure 14: Scenario #1 - Final State.....	28
Figure 15: Scenario #1 – Instance IDs	29
Figure 16: LTO-3 to LTO-5 Migration Example – Initial State.....	29
Figure 17: LTO-3 to LTO-5 Migration Example – Final State after Copy	30
Figure 18: LTO-3 to LTO-5 Migration Example – Final State after Move.....	31
Figure 19: Workflow for Processing a Series of Objects.....	32
Figure 20: Using a Disk Array as a Buffer	33

1 Introduction

1.1 Document Purpose and Scope

The purpose of this document is to explain the installation, configuration and operations of the DIVAmigrate Embedded Utility. Installation, Administration and Operations personnel should use this document to provide full functionality of the DIVAmigrate Embedded Utility.

1.2 Document Conventions

The following conventions are used with respect to text:

Normal Standard Text.

Italic Used to emphasize a term or variable.

Bold Used to emphasize critical information.

6.1 Refers to a section or sub-section in the document.

Courier New Used for system screen output and system commands.

The following conventions are used with respect to file paths or variables:



- **DIVA_HOME**: The Root Path on the file system where Oracle DIVArchive is installed.

The following conventions are used with respect to figures and drawings:

Red outlined boxes pointing to specific areas in a figure indicate procedural steps, or point out specific parameters, icons, tabs, etc. being discussed in the section text.

Red outlined boxes that surround specific areas in a figure indicate specific areas of the figure being discussed in the section text.

Table 1: System Component Icons for Non-Screenshot Workflows

Icon	Component
	Disc Cache
	Data Tapes

1.3 Overview

DIVAmigrate is installed as part of the Oracle DIVArchive Suite's standard installation, is placed directly under the Program folder and runs as a separate Windows Service. Migration Jobs are created using the DIVArchive Control GUI connected to Oracle DIVArchive Manager or using the Windows Command Line interface through the `client.bat` file located in:

```
DIVA_HOME\Program\Migrate\bin
```

Control of the utility is accomplished via the `migrate.bat` file (*refer to Section 3.1*), also located in:

```
DIVA_HOME\Program\Migrate\bin
```

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 Database so that the Control GUI will be able to show the status to the user.

1.4 Supported Environments

The DIVAmigrate Utility is compatible with the following platforms:

- DIVArchive 7.3 and later
- Windows Server 2003, 2008, and 2012.
 - Windows XP, Vista and Windows 7 are known to work, but aren't officially supported.
 - On 64-bit platforms the utility will run in 32-bit emulation mode.

2 Installation and Configuration

The DIVAmigrate Utility is part of the standard DIVArchive installation and is placed in the DIVArchive Program Folder. It may be installed on the DIVArchive Manager, or any machine capable of communicating with the DIVArchive Manager over TCP/IP protocol (*connectivity may be confirmed through a successful ping to the Manager from the client machine*). The following new folders can be found after initial DIVArchive installation is complete:

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

2.1 DIVAmigrate Service Configuration

The DIVAmigrate Service requires a valid configuration file during its `install` and `start` procedures. The default DIVAmigrate Configuration File is named `migrate.conf` and placed in `DIVA_HOME\Program\conf\migrate\` folder.

The configuration file is a standard Properties File similar to the DIVArchive Manager Configuration File. The Configuration File is not auto-reloadable and any changes made to the Configuration File will not take effect until the DIVAmigrate Service is restarted.

Notes:

- Refer to Appendix A1 Default Configuration for a sample configuration file.
- Additional parameters are being added to the Configuration File on a consistent basis and this document will be updated with new parameters as they are developed.

Make a copy of the `migrate.conf.ini` file and rename it to `migrate.conf`. It is important to make this copy and keep the original file intact to refer back to in case the configuration being worked on either does not work, becomes corrupt, or has/creates errors.

Open the file with Notepad (or any plain text editor) and populate the following parameters.

Important: Do not use Word, WordPad or any other editing tool that adds extra characters to a file.

The parameters in the table below are all mandatory unless otherwise noted in the description.

Table 2: DIVAmigrate Configuration Parameters

Parameter	Definition
SERVICE_NAME	This is the name for the Windows Service and is mandatory . The default value is <code>DivaMigrate</code> .
DIVAMANAGER_HOST	The Hostname or IP Address of the DIVArchive Manager. The default value is <code>127.0.0.1</code> .
DIVAMANAGER_PORT	This is the port to use to connect to the DIVArchive Manager. The default value is <code>9000</code> .
DIVA_MIGRATE_MANAGEMENT_PORT	This is the management port number. The default value is <code>9191</code> .

Parameter	Definition
DIVAMANAGER_DBURL	<p>A URL to connect to the Database instead of using DIVAMANAGER_DBHOST and DIVAMANAGER_DBPORT or DIVAMANAGER_TNSNAME. This gives the additional flexibility to the user to explicitly identify whether to use the <code>jdbc thin</code> driver or the <code>jdbc oci</code> driver to connect to the Oracle Database.</p> <p>Examples:</p> <pre>jdbc:oracle:thin:@host:port:oracle_sid jdbc:oracle:oci:@tnsname</pre> <p>Note: This is not a mandatory parameter.</p>
DIVAMANAGER_DBUSER	<p>The username the DIVArchive Manager uses to connect to the DIVArchive Database. The default value is <code>diva</code> (<i>case sensitive</i>).</p>
DIVAMANAGER_DBPASSWORD	<p>The DIVArchive Manager password for connecting to the Database. The default value is <code>1ib5</code> (<i>case sensitive</i>).</p>
DIVAMANAGER_TNSNAME	<p>TNS Name of the DIVArchive Schema within the Oracle Database. This setting is ignored if the DIVAMNAGER_DBHOST and DIVAMANAGER_DBPORT settings (<i>below</i>) are defined. There must be a corresponding entry in TNSNAMES.ORA found under the Oracle 11 Client installation.</p> <p>Note: This is not a mandatory parameter.</p>
DIVAMANAGER_DBHOST	<p>This parameter specifies the hostname or IP Address of the machine containing the DIVArchive Database. If using a hostname, this must be present in the <code>hosts</code> file on the machine where the DIVArchive Manager is installed.</p> <p>Examples:</p> <pre>127.0.0.1 localhost</pre> <p>Note: This is not a mandatory parameter.</p>

Parameter	Definition
DIVAMANAGER_DBPORT	<p>The Oracle Listener Port configured during the DIVArchive Database installation. The default value is port 1521.</p> <p>Note: This is not a mandatory parameter.</p>
DIVAMANAGER_DBSID	<p>The DIVArchive Database instance System Identifier (<i>SID</i>) in Oracle where the DIVArchive Manager connects. <i>This is typically lib5 in most DIVArchive installations.</i> Consult the location's Delivery Plan for confirmation. The default value is lib5.</p>
MAX_SIMULTANEOUS_REQUESTS	<p>This is the maximum number of simultaneous Manager requests run by DIVAmigrate. The default value is 15.</p> <p>Note: This is not a mandatory parameter.</p>
DB_SCAN_PERIODICITY	<p>This parameter (<i>in seconds</i>) determines how often DIVAmigrate looks for new jobs in the Database. The default value is 60 seconds.</p> <p>Note: This is not a mandatory parameter.</p>
DIVA_RECONNECT_PERIODICITY	<p>This parameter (<i>in seconds</i>) determines the time between reconnection attempts in the event of connectivity loss with the Manager. The default value is 30 seconds.</p>
MAX_FAILED_REQUESTS_PAUSE	<p>This parameter identifies the maximum number of sequential failure requests that may occur before DIVAmigrate pauses the migration job. DIVAmigrate will always pause if the configured number of requests fails sequentially. The default value is 10.</p> <p>Note: This is not a mandatory parameter.</p>
JOB_MAX_INACTIVE_TIME	<p>This parameter identifies the Migrations Plan's maximum inactivity time. If, after the service is restarted, it finds running jobs with the last access time greater than this value, the Migration Plan for those jobs will be recreated. The default value is 24 hours.</p> <p>Note: This is not a mandatory parameter.</p>

Notes:

- The Windows Service Wrapper Configuration should not be modified and is not included in the sample Configuration File in A1 Default Configuration.
- The Logging Section of the Configuration File is discussed in the next section.
- The DIVA Service Options section should not be modified unless instructed by Oracle personnel; however this section does appear in the sample Configuration File in A1 Default Configuration.

2.2 Logging

The logging methods used in the DIVArchive Manager are also used for DIVAmigrate. DIVAmigrate logs are placed in the `DIVA_HOME\Program\log\migrate` folder.

DIVAmigrate logs are automatically archived and divided into separate files each time when current log file reaches its size limit.

Table 3: DIVAmigrate Logging Parameters (in *migrate.conf*)

Parameter	Definition
<code>LOGGING_DIRECTORY</code>	This is the directory where the DIVAmigrate log files are stored. The default is <code>../log/migrate</code> .
<code>LOGGING_ROOT_LEVEL</code> <code>LOGGING_TRACE_LEVEL</code> <code>LOGGING_SERVICE_LEVEL</code>	These three parameters can be modified to suit the required level of activity logging. The default value is <code>INFO</code> and valid options for each are: <ul style="list-style-type: none">• <code>DEBUG</code>• <code>INFO</code> (default value)• <code>WARN</code>• <code>ERROR</code>• <code>FATAL</code>
<code>LOGGING_MAXFILESIZE</code>	This parameter identifies the maximum size of the log file before it is archived and a new file is created. The file size should be specified using <code>KB</code> or <code>MB</code> to indicate Kilobytes or Megabytes respectively. The default value is <code>10MB</code> . Example: <code>LOGGING_MAXFILESIZE=10MB</code>
<code>LOGGING_LIFETIME</code>	All files older than the value of this parameter will be removed (including <i>trace</i> , <i>service</i> , and <i>.zip</i>). The value for this parameter is in hours and the default value is 50 (<i>hours</i>).

3 Operations

3.1 Starting and Stopping the Service

The DIVAmigrate Utility can be started and stopped from the Windows Command Line or Windows Service Manager Console. To install the DIVAmigrate Utility, open a Windows Command Line Window and navigate to the folder where the `migrate.bat` file is located. Run the `migrate.bat` batch file with the appropriate parameters needed to perform the desired task(s).

Windows 2008 example:

Open the **Windows Start Menu**. Click **All Programs**, then **Accessories**, and finally **Command Prompt**. When the Command Prompt Interface Window opens, a command similar to the following may be executed:

```
DIVA_HOME\Program\Migrate\bin> migrate.bat install
```

The Windows Service short name displayed in the Windows Services Manager is `DivaMgrrt` and the full name is displayed as `DIVArchive Migrate - DIVAmigrate`.

Warning: It is the user's responsibility to make sure that only one instance of the DIVAmigrate Service is running at any time. Running several instances of DIVAmigrate connected to same DIVArchive Manager will lead to incorrect Migration Job processing and may result in data loss.

The syntax for the command line interface is:

```
migrate.bat command [options]
```

Table 4: DIVAmigrate Commands and Options

Command	Description
<code>install</code>	Installs the module as a Windows System Service.
<code>uninstall</code>	Removes the module as a Windows System Service.
<code>start</code>	Start the service.
<code>stop</code>	Stops the service if already running.
<code>restart</code>	Stops and subsequently starts the service.
<code>version</code>	Displays the module version information and exits.
<code>status</code>	Displays current service running status.
<code>help</code>	Displays this information and exits.
Option	Description
<code>-conf (-f)</code>	Specifies and configuration file to load.

3.2 Migration Job Command Line Syntax

DIVAmigrate Migration Jobs may be started using the DIVArchive Control GUI or the Windows Command Line interface through the `client.bat` file located in:

```
DIVA_HOME\Program\Migrate\bin
```

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

For example, the following `client` command:

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

Will fail with the following error message because the `-buffer` option is expected to occur after the `-media` and `-count` options:

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

The following syntax is available for use with DIVAmigrate:

```
client -tape barcode
  [[-media mediagroup] [-count count] ...]
  [-buffer array] [-delete] [-excl file]
  [-autosrc|recovery] [-start datetime] [-end datetime]
  [-priority value]
```

```
client -tapelist file
  [[-media mediagroup] [-count count] ...]
  [-buffer array] [-delete] [-excl file]
  [-autosrc|recovery] [-start datetime] [-end datetime]
  [-priority value]
```

```
client -instlist file
  [[-media mediagroup] [-count count] ...]
  [-buffer array] [-delete] [-excl file]
  [-autosrc|recovery] [-start datetime] [-end datetime]
  [-priority value]
```

```
client -tapegroup mediaName
  [[-media mediagroup] [-count count] ...]
  [-buffer array] [-delete] [-excl file]
  [-autosrc|recovery] [-start datetime] [-end datetime]
  [-priority value]
```

```
client -diskarray mediaName
  [[-media mediagroup] [-count count] ...]
  [-delete] [-excl file]
  [-start datetime] [-end datetime]
  [-priority value]
```

```
client -list_running_jobs
```

```
client -job_details jobID
```

```
client -[pause|stop|resume|cancel|delete|retry] jobID
```

```
client -help
```

Table 5: Command Line Options

Option	Definition
-tape <i>barcode</i>	The tape to migrate. This will migrate instances present on the tape for the specified barcode.
-tapelist <i>file</i>	<p>A file containing a list of tapes to migrate. This will migrate instances present on the tapes listed in the file.</p> <p>The <i>file</i> is a flat text file (<i>DOS or Unix format</i>), where the format of each line is:</p> <p style="text-align: center;"><i>barcode</i></p> <p>Empty lines and lines beginning with hash mark (#) are ignored.</p>
-instlist <i>file</i>	<p>A file containing a list of instances to migrate. This will migrate the instances listed in the file.</p> <p>The <i>file</i> is a flat text pipe () character separated file (<i>DOS or Unix format</i>), where the format of each line is:</p> <p style="text-align: center;"><i>objectname category instanceid</i></p> <p>Empty lines and lines beginning with a hash mark (#) are ignored.</p>
-tapegroup <i>mediaName</i>	This option migrates all Object Instances in the entire Tape Group. <i>mediaName</i> is the Tape Group's Name.

Option	Definition
-diskarray <i>mediaName</i>	This option migrates all Object Instances in the entire Disk Array. <i>mediaName</i> is the Disk Array's Name.
-media	<p>Media to which objects should be migrated.</p> <p>Specifying a different Media Group other than the Source Media allows for various migration operations; e.g. migrating data from one media to another.</p> <p>When this option isn't specified, the Destination Media Group will be the Source Media Group of the tape or instance being processed and will correspond to the <code>COPY_TO_SOURCE</code> migration strategy.</p> <p>Note: This option is mandatory with the <code>-instlist</code> and <code>-tapelist</code> options.</p>
-count <i>count</i>	<p>The Target Instance Count for the objects in the Destination Media. The default is 1.</p> <p>If multiple Destination Media are specified (<i>by using several <code>-media</code> options</i>), the user must provide the same amount of <code>-count</code> options in the same order (<i>or none to use the default</i>).</p> <p>Note that when specifying a <code>COPY_TO_SOURCE</code> Migration Type (<i>where Source and Target Media are the same</i>), <code>-count 2</code> is required rather than the default of <code>-count 1</code> to specify that there will be 2 instances on the Target Medium.</p>
-delete	Deletes the Source Instances once processed. This option represents the <code>MOVE</code> migration strategy
-autosrc	Enables the Alternate Sources option. This enables the Migration Service to search if the Source Tape Instance exists on an Array and uses it for migration.
-recovery	Enables Recovery Mode.

Option	Definition
-excl <i>file</i>	<p>Ignore instances listed in <i>file</i>.</p> <p><i>file</i> is a flat text file (DOS or Unix format).</p> <p>The format of each line is:</p> <pre>object category instanceId</pre> <p>where object is the Object Name to exclude and category is the Object Category, or * to match all Object Names or (<i>mutually exclusive</i>) all Categories. instanceId is the ID of the instance.</p> <p>Blank spaces in object, category, and instanceId are accepted.</p> <p>Objects and Categories containing a ' ' (<i>vertical bar</i>) within the name are permissible, but the character must be escaped using the backward slash character like this: '\ '.</p> <p>Example:</p> <p>'object one' should be written as 'object\ one'.</p>
-buffer array	Specifies the Disk Buffer to use.
-start <i>datetime</i>	<p>Specifies Scheduled Start Time for a job in the following format:</p> <pre>dd-MM-YYYY-HH:mm</pre>
-end <i>datetime</i>	<p>Specifies Scheduled End Time for a job in the following format:</p> <pre>dd-MM-YYYY-HH:mm</pre> <p>The job will enter the PAUSE state if the job is not completed before the scheduled end time.</p>
-priority <i>value</i>	Specifies the Migration Job Request Priority. This value must be between 1 and 100. The default value is 20.
-[<i>command</i>] <i>jobID</i>	<p>Sends the command for specific job. Commands include:</p> <ul style="list-style-type: none"> • pause • stop • resume • cancel • delete • retry
-list_running_jobs	Prints a list of unfinished jobs with their IDs and statuses.
-job_details <i>jobID</i>	Returns the status and progress for a given job.
-help	Displays the help text.

3.3 DIVAmigrate GUI

The DIVArchive Control GUI has been enhanced with the new **Migrate Panel** and **Migration Wizard**.

The **Migration Wizard** allows users to create new Migration Jobs.

The **Migration Panel** allows users to view the status and results of completed and currently running Migration Jobs. It also allows users to **Pause**, **Stop**, **Resume**, **Delete**, **Cancel** and **Rerun** Migration Jobs.

3.3.1 DIVAmigrate Wizard

The Migration Wizard is a series of Dialog Windows. Each Dialog Window presents the user with a set of choices for Migration Job Parameters and, based on user's selections, will show the next dialog in typical *wizard fashion*, until all required parameters are configured.

The Button to open the Migration Wizard is on the Button Toolbar at the top of the DIVArchive Control GUI window and to the **Actions Button Group** in the **Ribbon**.

The DIVAmigrate Wizard is only available to users with Administrator privileges in the Control GUI.

Below are the different stages and dialogs of the **DIVAmigrate Wizard**. The process is self-explanatory in the screens shown:

Figure 1: Action Menu – Migrate Actions – Create Migrate Job

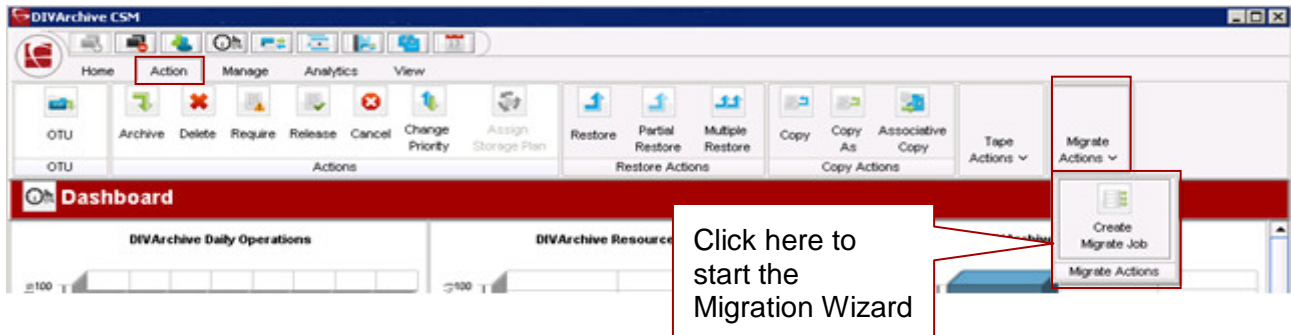


Figure 2: First Dialog - Select Migration Strategy

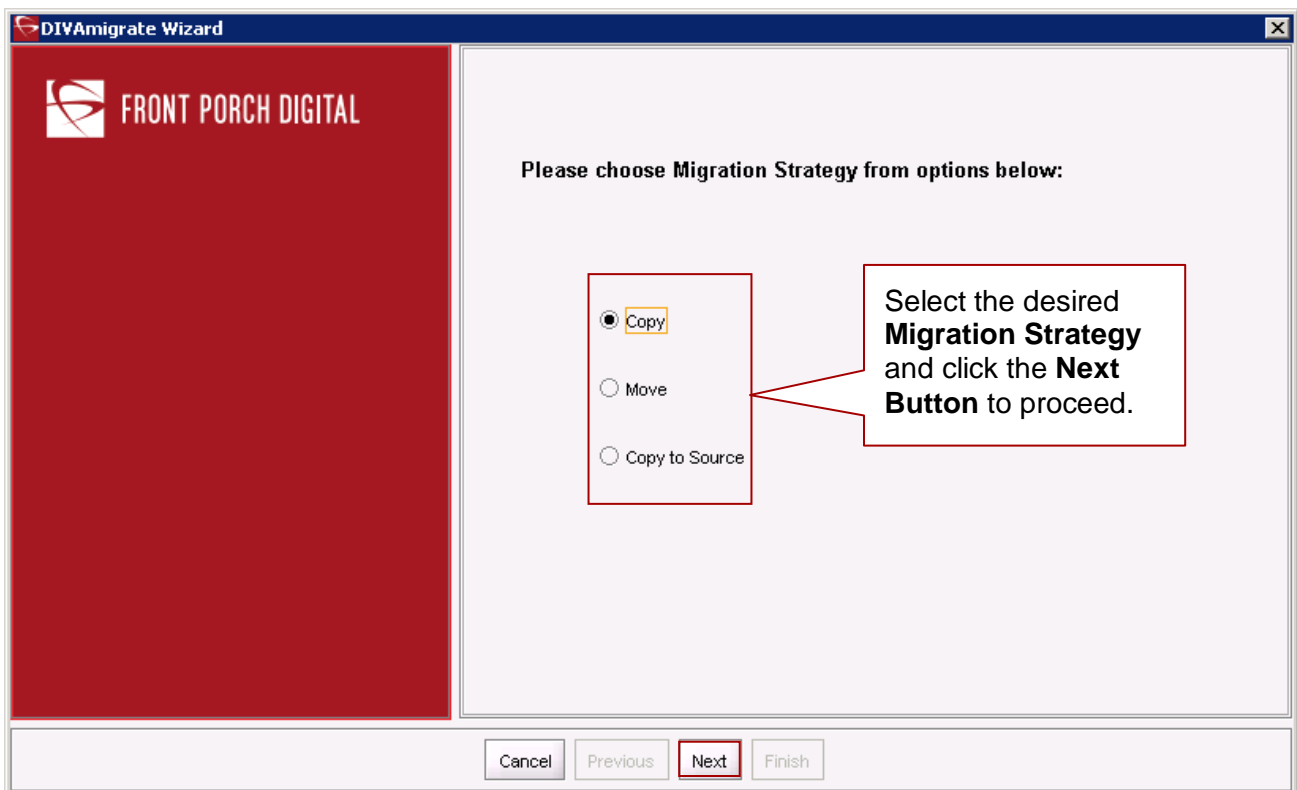
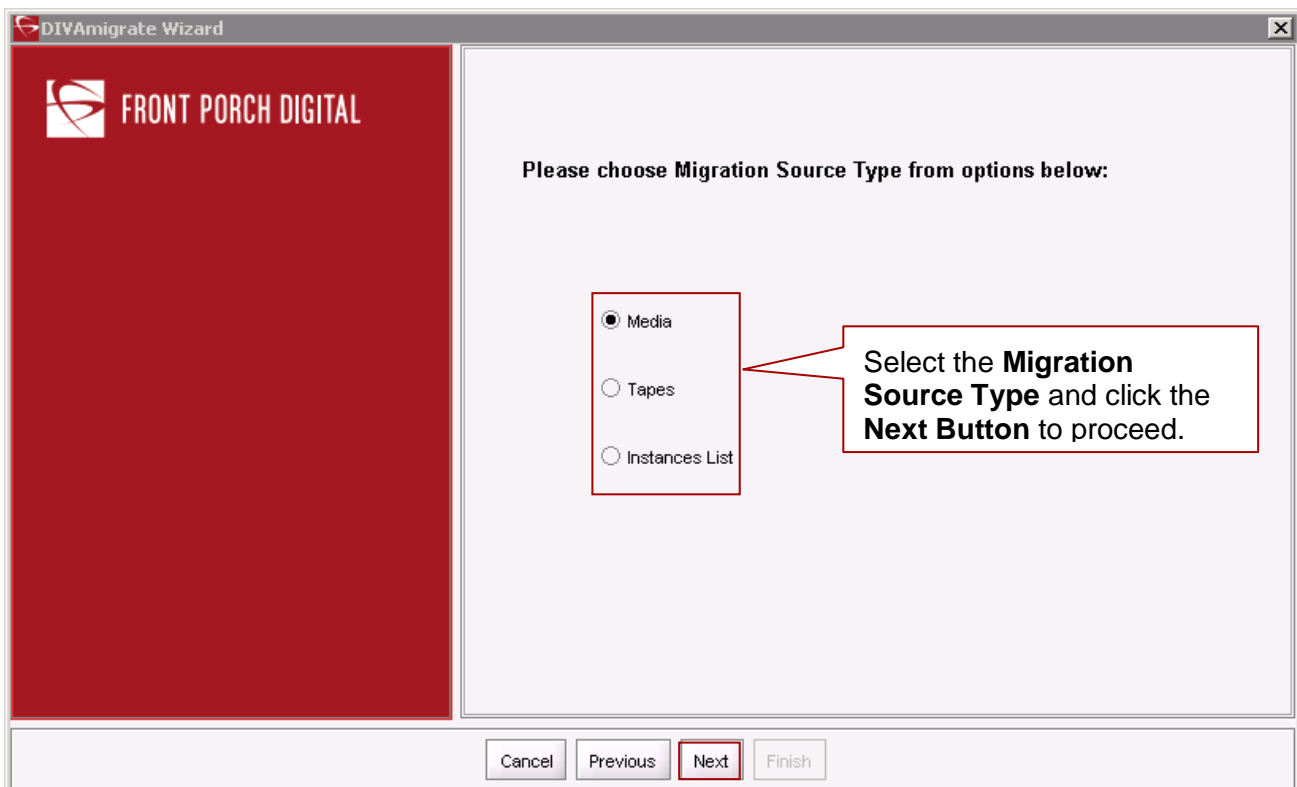


Figure 3: Select the Migration Source Type



If the user previously selected the **Copy to Source Migration Strategy**, the Dialog Window above is skipped because only the **Media Option** is available for that strategy.

Figure 4: Select the Migration Source

FRONT PORCH DIGITAL

Please choose Migration Source:

Please select Media: group_001_axf

Use the pull-down box to select the **Migration Source** and click the **Next Button** to proceed.

Cancel Previous **Next** Finish

Figure 5: Select Media and Number of Instances

FRONT PORCH DIGITAL

Please choose Migration Destination:

Select Media: default

Select number of instances to migrate: 1

Use the pull-down box to select the **Destination Media**, enter the number of instances to migrate, and click the **Next Button** to proceed.

Cancel Previous **Next** Finish

Figure 6: Add Additional Migration Destinations

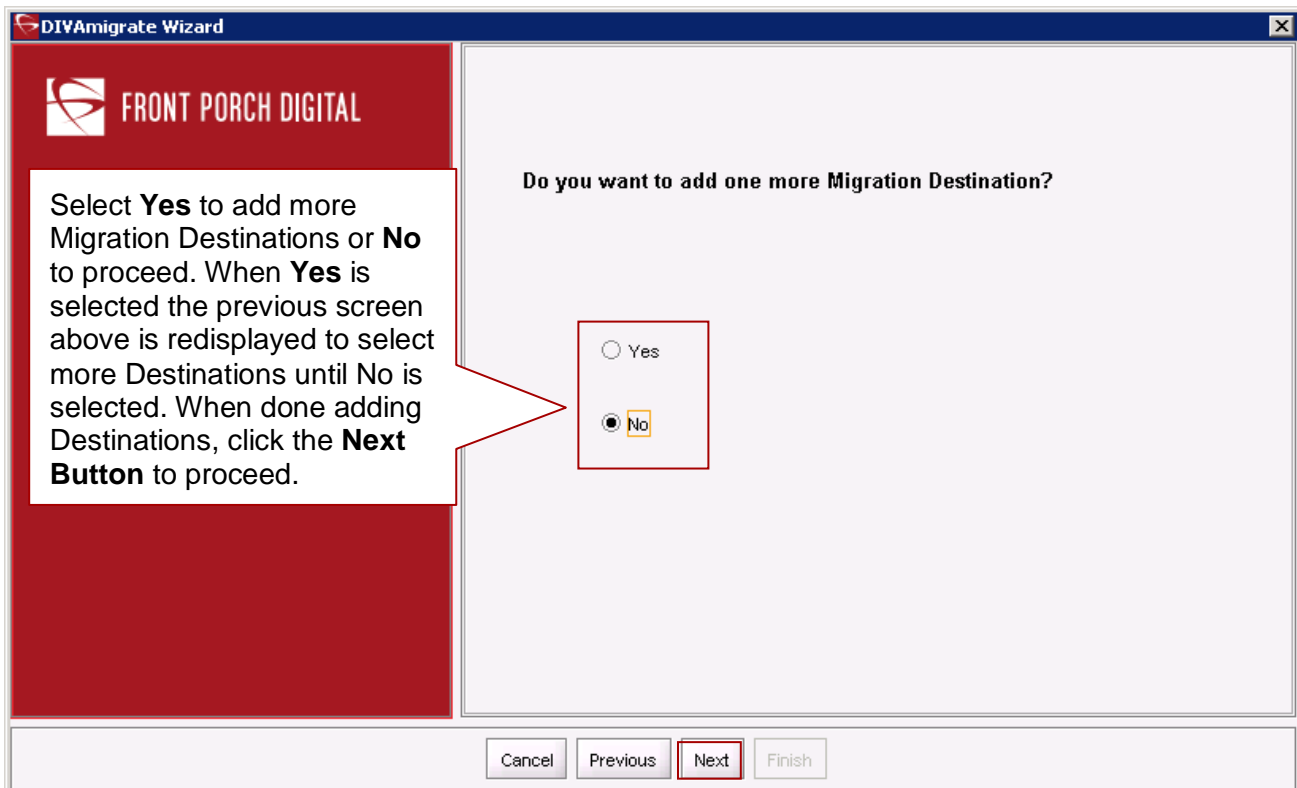


Figure 7: Selecting the Migration Job Options

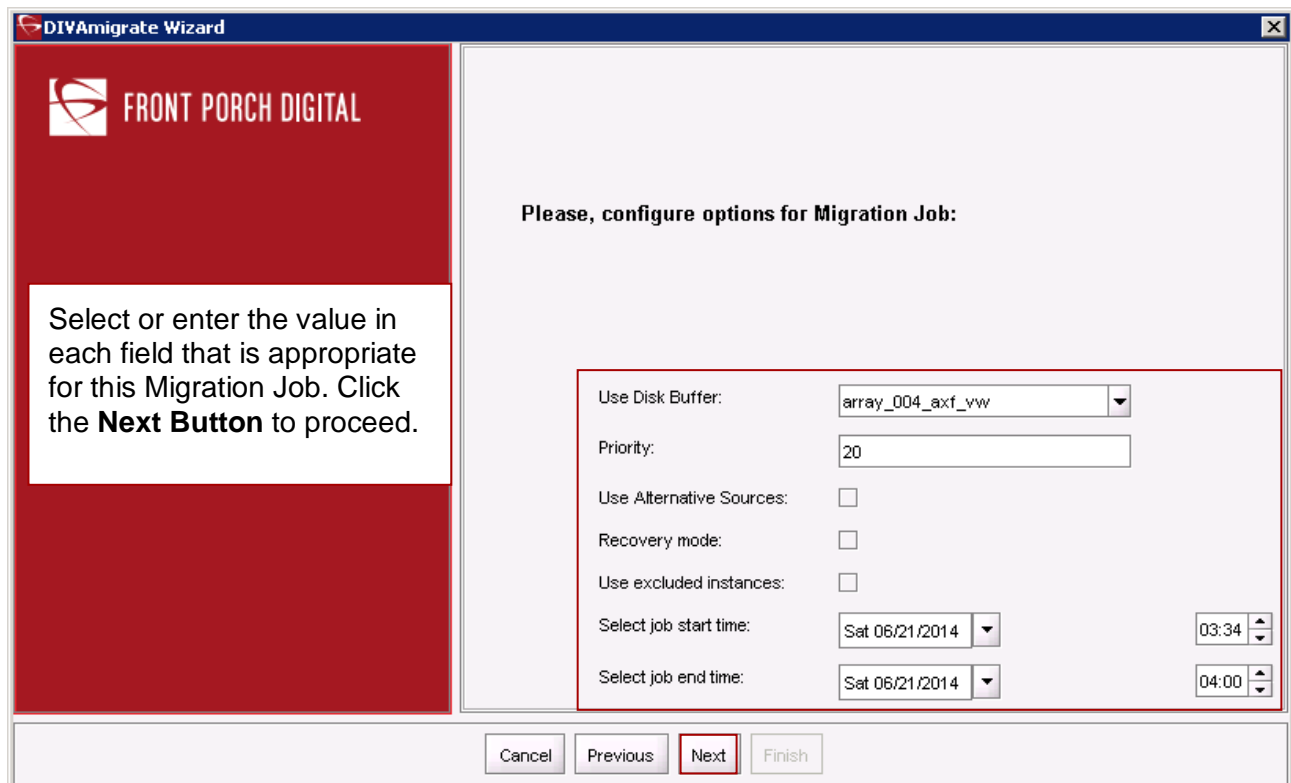
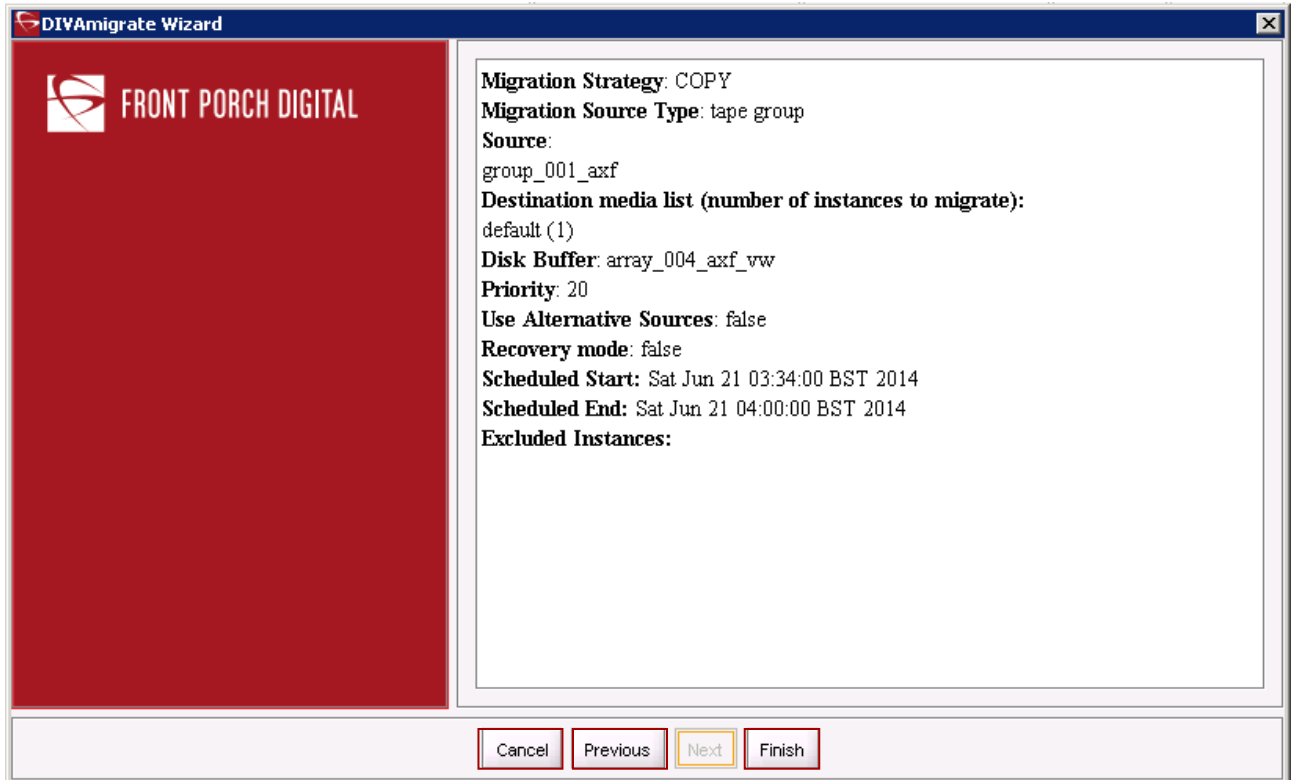


Figure 8: Completing the Wizard



This screen will appear once the Migration Wizard has been completed and show the selections made during the process. At this point the user may either click the **Finish Button** to schedule the job, click the **Cancel Button** to exit the Wizard (*no job will be created*), or use the **Previous Button** to go back and make changes as desired.

3.3.2 DIVAmigrate Panel

The new **DIVAmigrate Panel** is located within the DIVArchive Control GUI and displays information about Migration Jobs in a table with following columns:

- ID (*number*)
- Status
- Type (*Copy, Move, Copy to Source*)
- Source Type
- Destination Type
- Started At
- Finished At
- Progress
- Instances to Migrate
- Data to Transfer (*Gb*)

The following columns from the list above allow filtering:

- Type
- Scheduled Start Time
- End time
- Status

A Single Row Selection Model is used for the table and the Context dialog is available for each row with following options:

- Properties...
 - Opens Migration Job Properties Dialog.
- Cancel
 - Allows the user to cancel running Migration Jobs.
- Stop
- Pause
- Resume
- Rerun
 - Creates new Migration Job with same parameters.
- Delete

The Properties Dialog will show the status for each object affected by the Migration Job as well as the ID of the last request executed for that object. It is possible to directly open the **Request Properties Dialog** by clicking on the **Request ID**. The **Object Details** table in the **Migration Job Properties Dialog** displays 1000 objects at a time and supports pagination.

3.4 Migration Jobs

Important: DIVAmigrate will not allow any changes to the Migration Job parameters and options after a Migration Job is submitted.

3.4.1 Migration Job Definitions

DIVAmigrate allows users to create **Migration Jobs** and then it will try to run them. Each Migration Job consists of following **mandatory** parameters configured by the user:

- **Migration Source**
 - Describes the set of instances which should be migrated.
- **Migration Destination**
 - Specifies the location where migrated instances should be placed.
- **Migration Strategy**
 - Specifies the type of Migration (*Copy, Move, Copy to Source*)
- **Migration Options**
 - Other parameters and options specified by the user.

3.4.2 Migration Job Actions

Users are able to perform following actions with Migration Jobs:

- **Create New Job**
 - Creates new job
- **Cancel Job**
 - Stops the specified job without the possibility of resuming. The Job receives the Cancelled Status.
- **Rerun Job**
 - Creates a new job with parameters same as in original job
- **Pause**
 - Saves the current state of the migration and pauses job execution. After resuming, the job will continue execution from the point where it was paused without recollecting data.
- **Stop**
 - Stops the job and maintains the ability to resume later. After resuming, the job's internal state is completely reinitialized and all migration data is recollected.
- **Delete**
 - Deletes the job from the Database. If the job was running or pending it will be stopped first and then deleted. This works for all job's states (*Pending, Running, Completed, etc.*).

- **Resume**
 - Resumes the job's execution for a job in a Stopped or Paused status.

3.4.3 Migration Job Status

Based on the Migration Job Lifecycle described above and possible user actions, each Migration Job will have one of the following statuses:

- **Submitted**
 - The Migration Job was configured by the user, but DIVAmigrate didn't start working on it yet.
- **Pending**
 - DIVAmigrate found a new Migration Job and started processing the job.
- **Initializing**
 - DIVAmigrate collects all necessary information required to start Migration.
- **Copying Content**
 - DIVAmigrate creates copies of instances selected for Migration.
- **Cleaning Up**
 - In the case of a **Move Migration Strategy**, DIVAmigrate deletes already migrated instances from Migration Source.
 - If a Disk Buffer is used, DIVAmigrate removes the instances from the buffer.
- **Finalizing**
 - DIVAmigrate checks the results of Migration and updates Migration Job status and other information.
- **Completed**
 - The Migration Job was completed.
- **Partially Completed**
 - The Migration Job was completed, but some objects weren't migrated.
- **Cancelling**
 - The command to cancel the Migration Job was received but has not yet been processed by the DIVAmigrate Service.
- **Cancelled**
 - The Migration Job was cancelled by the user.
- **Aborted**
 - The Migration Job was not completed due to an error.
- **Pausing**
 - The command to pause the Migration Job was received but has not yet been processed by the DIVAmigrate Service.
- **Paused**

- The Migration Job has been paused by the DIVAmigrate Service.
- **Stopping**
 - The command to stop the Migration Job was received but has not yet been processed by the DIVAmigrate Service.
- **Stopped**
 - The Migration Job was stopped by the DIVAmigrate Service.
- **Resuming**
 - The command to resume the Migration Job was received from the client (*GUI*) but has not yet been processed by the DIVAmigrate Service.
- **Deleting**
 - The user submitted the command to delete the Migration Job entirely.

3.4.4 Migration Job Parameters

This section describes in detail each of Migration Job parameters:

- Migration Source
- Migration Destination
- Migration Strategy
- Migration Options

3.4.4.1 Migration Source

Defines the set of instances for migration. The Migration Source can be provided by the user 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, instance id*).

The user 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.

Note: List of Tapes and List of Instances used as a Migration Source doesn't support the asterisk (*) mask filter. Only Excluded Instance List supports the asterisk (*) mask filter.

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

- Files 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 should contain the **Object Name**, **Object Category** and **Instance ID** in exactly that order with the pipe symbol (|) used as a separator.

Example:

```
objectname|category|instanceId
```

Note: Character escaping is supported in the case the pipe symbol (|) is present in the Object's Name or Category.

3.4.4.2 Migration Destination

Defines the media(s) to which objects should be migrated. The following Migration Destinations are valid:

- Single Media
- Multiple Media
- Same media as the Migration Source (*for Copy to Source Migration Strategy*)

In the case of Multiple Media being selected for the Migration Destination, the user must provide the number of instances that should be present for each media separately.

For the number of new instances which should be created on the Migration Destination, the following rules apply:

- DIVAmigrate will check how many instances of each object to be migrated already exist on the Migration Destination and will count already existing instances on the destination as already migrated.

Example: one copy of instance for `object-A` already exists on the Destination Media called `default` and the migration job was configured with the number of instances on the Destination Media `default` as 1. The Migration Job would perform no additional copy of `object-A` to the Destination Media `default` as one instance already existed before the Migration Job started.

- In the case when Multiple Media is selected as the Migration Destination, the user must provide the number of new instances for each Destination Media separately.
- In the case of the Job having multiple destinations with a combination of both Tape Groups and Disk Arrays, the Migration Service will always migrate to the Disk destination first.
- In the case of the job having 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 will always use 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 will be used for migration.

3.4.4.3 Migration Strategy

Defines the type of Migration to be performed. The following Migration Strategies are currently implemented:

- **Copy**
 - Migrates the configured number of instances from the Migration Source to the Migration Destination, but **does not** remove original instances from the Migration Source.
- **Copy to Source**
 - This is the same as the Copy Strategy, but the Migration Source and Migration Destination are the same. **This strategy is applicable only for Single Media Migration Source.**
- **Move**
 - This strategy will migrate the configured number of instances from the Migration Source to the Migration Destination, **and** deletes original instances from the Source.

3.4.4.4 Migration Options

The following Migration Options are currently implemented:

- **Requests Priority** (*mandatory parameter*)
 - The priority of requests is sent to the DIVArchive Manager by the DIVAmigrate Service during the Migration Job.
- **Excluded Instances List** (*optional parameter*)
 - A list of instances provided by the user that should be excluded from the migration. The user has the option to input the list as a file with values separated by pipe symbol (|) or manually. Both the file and manual input use a similar convention:
 - The **Object Name** and **Object Category** support filtering with the asterisk (*) symbol.
 - The **Instance ID** can be a number or an asterisk (*), where (*) means *any/all instance(s)*.

Example: objectname|category|instanceId

- **Use Alternative Instance Sources** (*this is an optional parameter and is turned off by default*).
 - If this option is enabled, DIVAmigrate will perform the following processing for each instance it tries to migrate:
 - If the instance is on tape, DIVAmigrate checks for a Disk Instance of the same object anywhere in the system and use 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 will always migrate the Disk Instances available on the alternative disk first as part of the migration plan.

Notes:

- **DIVAmigrate always considers all Disk Instances as being online and doesn't perform an availability check for them. If for some reason the disk where the instances are located is broken, the migration of the instance will fail and the error is logged.**
 - **The original instance is always used if the migration of alternative instance is failed.**
- **Recovery Mode** (*this is an optional parameter and is turned off by default*)
 - 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. This allows implementation of recovery scenarios for damaged offline tapes.
 - When this option is on, it automatically means that Alternative Instance Source option is also on. If the migration operation fails during the migration of an alternative recovery instance, the original source instance will be used for migration.
- **Disk Buffer Name** (*this is an optional parameter*)
 - Users are able to specify the Disk Array Name to use as the Migration Disk Buffer. Currently only one Disk Array may be assigned as a Disk Buffer.
 - Tape objects will be initially migrated to Disk Buffer from Migration Source, and only after that they will be migrated from Disk Buffer to Migration Destination.
 - New configuration parameter will be added to Disk Arrays on DIVA level to specify Max space percentage available for Migration. This parameter will be configurable in DIVA Configuration Utility. During migration process max used space on Disk Buffer will be periodically compared to this parameter, and if used space percentage is greater or equal to the configured parameter, then Migration Job will try to migrate objects from Disk Buffer to Destination and Clean up Disk Buffer and continue with the remaining objects to be migrated. If after Disk Buffer clean up there is still not enough free space then job will be paused.

The following rules will be applied when using the Disk Buffer:

- The Migration Service will **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 will be migrated from the Disk Buffer to the Migration Destination and cleaned from the Disk Buffer. After that, the Migration Job will continue processing as usual.
- If the Option to use Alternative Sources for migrating instances is on, and for some Tape Instance there is Alternative Disk Instance, the instance will not be copied to the Disk Buffer.
- Instances cached to the Disk Buffer will be deleted from it as a part of the same Migration Job.

- DIVAmigrate will check if the Disk Buffer percentage of used space is less than the configured value for each Disk Array in the DIVArchive Configuration Utility. If not, the Migration Job will be 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.
- **Scheduled Start and End Time for Migration Job** (*this is an optional parameter*).
 - The user can provide Start and/or End Date and Time for the new Migration Job.
 - For **Scheduled Start Time**, DIVAmigrate guarantees not to start the Migration Job until this time is reached. DIVAmigrate will try to start the Migration Job as soon as its Scheduled Time is reached if it has enough available resources.
 - For **Scheduled End Time**, DIVAmigrate guarantees to stop the Migration Job's processing after this time has been reached. A stopped Migration Job will receive the **Paused Status**, and after resuming, the End Time parameter will be removed from the job (*this is currently how the function is implemented. In future updates, it is planned to allow users to edit the End Time parameter*).
 - Both the **Start Time** and **End Time** parameters are **optional**. Users can provide none, both or either one of them.

The current implementation of the DIVAmigrate Service validates user input for a new Migration Job on two levels:

- In the GUI when the user creates a job
- In the Service when the job is initialized.

No job parameter validation is performed for jobs submitted through the command line.

3.5 Basic Migration

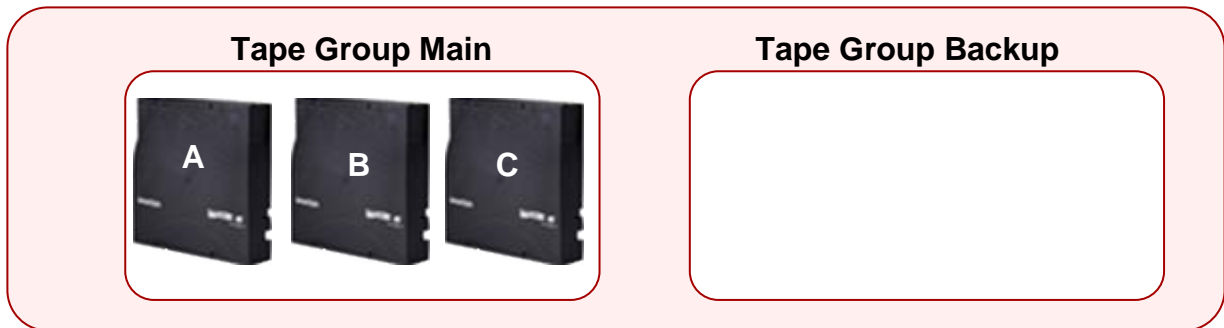
3.5.1 Copying Data to another Group or Array

The following scenarios will describe different possible outcomes when copying data to another Group or Array using the DIVAmigrate Utility.

Scenario #1:

A series of objects (A, B and C) are in **Tape Group Main** and they need to be duplicated to **Tape Group Backup**. The diagram below displays the initial state:

Figure 9: Scenario #1 - Initial State



Submit a copy job to “*Make a copy of objects A, B and C to tape group Backup*”.

This command would result in the following state, which is what was requested: Objects A, B, and C now have instances in both Tape Groups:

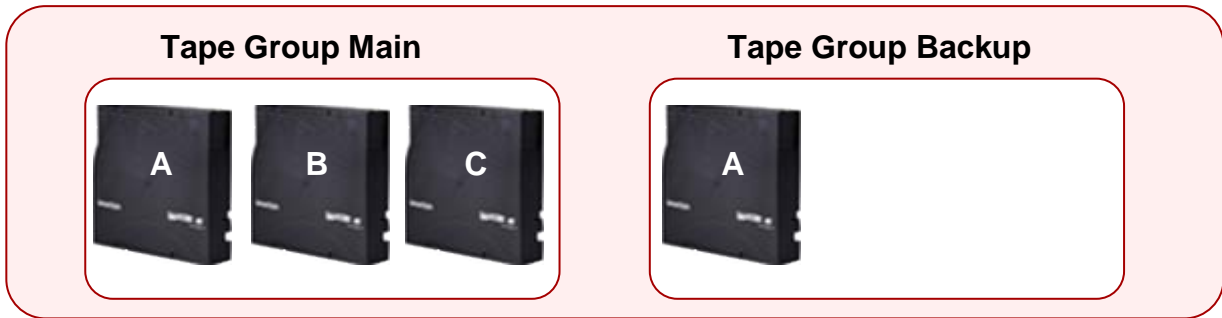
Figure 10: Scenario #1 – Full Duplication Complete



Scenario #2:

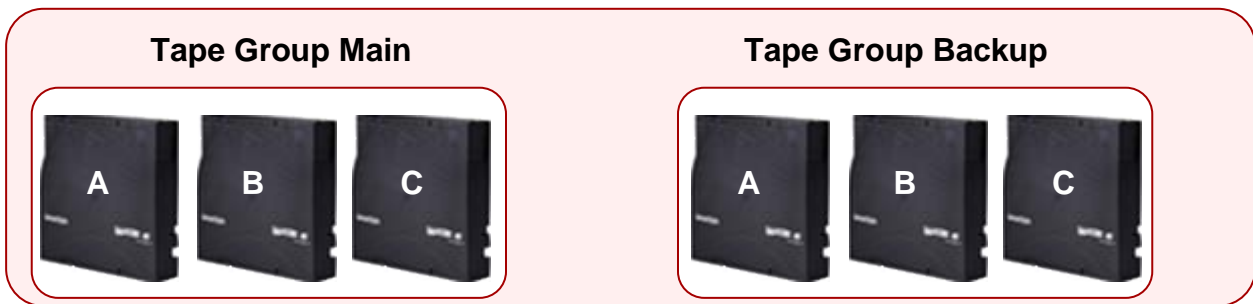
If the initial state was as shown in the figure below instead of as in Scenario #1:

Figure 11: Scenario #2 – Initial State



Then asking for a copy of objects A, B and C would end up with this state:

Figure 12: Scenario #2 – Full Duplication Completed



There is now one instance of each object in **Tape Group Backup**.

There are many possible reasons for **Object A** already being in the **Tape Group Backup**:

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

In production situations, it is not generally known how many instances the objects currently have in the Target Group (*possibly none, or maybe one such as Object A in the example above*); however, it **is** typically known exactly how many are desired. In the example above, the desired result is to have one instance in Tape Group Backup for each object at the end of the duplication run; which is how the DIVAmigrate Utility was designed to perform. The utility expects the user to specify the desired number of instances in the Target Group or Array. Depending upon the actual situation, DIVAmigrate will only perform the copy operations required to achieve the desired end result.

3.5.2 Moving Data to another Group or Array

There will be scenarios where moving the data is required instead of replicating it. For example, this is useful when migrating data to a new technology (e.g. from LTO-3 to LTO-5), and it is not necessary (or desired) to keep the old tape instances.

Scenario #1:

In this scenario, a new **Tape Group Main-LTO-5** has been created and assigned a Tape Set ID populated with LTO-5 tapes. The diagram below displays the initial state:

Figure 13: Scenario #1 - Initial State



This command will instruct DIVAmigrate to perform the job correctly:

```
client <object list> -media Main-LTO-5 -count 1 -delete
```

Notice that this command is very similar to the one seen previously; the only difference being the addition of the `-delete` option. This option tells DIVAmigrate to delete the source instances after copying them. In the example, the instances of Objects A, B and C located in Tape Group Main will be removed.

Here is what DIVAmigrate will do in this scenario:

Object A → Copy → DeleteInstance → Complete

Object B → Copy → DeleteInstance → Complete

Object C → Copy → DeleteInstance → Complete

The final state is as follows:

Figure 14: Scenario #1 - Final State

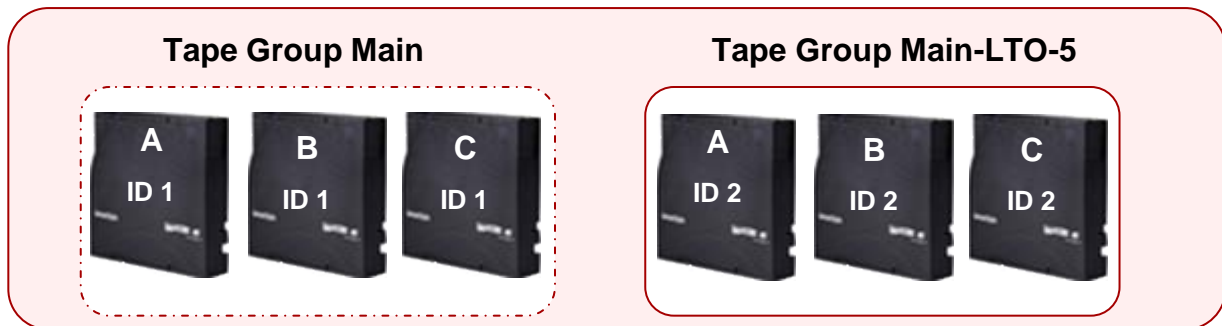


The instances have been (*basically*) moved to the new Tape Group; however, they are essentially not the same instances of the objects because the original instances were actually copied to the new Tape Group and then deleted from the source Tape Group.

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 **Main-LTO-5 Group** has a different **Instance ID** than the instance that was located in **Tape Group Main**. Since the new instance is a copy of the same object as the old one, the data is exactly the same and the data was actually moved.

The figure below shows what the **Instance IDs** could be in this scenario. The instances deleted at the end of the run are shown in **Tape Group Main** with the dotted outlines – Tape Group Main still exists, the instances in it do not because they were deleted.

Figure 15: Scenario #1 – Instance IDs



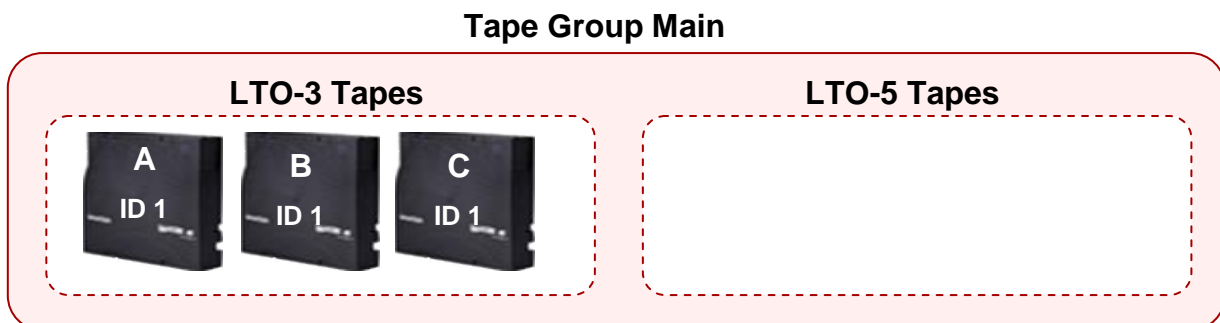
3.5.3 Copying and Migrating Data to the Same Group or Array

Sometimes moving or replicating data to the same Group or Array as the one containing the Source Instances is desired - for example, if a tape technology migration is being performed and the user wants 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 Set ID and the same group will contain both Tape Types. The Oracle Technical Team will generally set the tapes of the old type to **Not Writable** (or *alternately to Protected*) to force DIVArchive to write new data to the tapes of the new technology type.

Note: The user might find it efficient to create a new group that only contains the desired LTO-5 Tapes and use that as the Target Media.

This is an example of a migration of LTO-3 to LTO-5:

Figure 16: LTO-3 to LTO-5 Migration Example – Initial State



The Migrate command line for replicating the data will be:

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

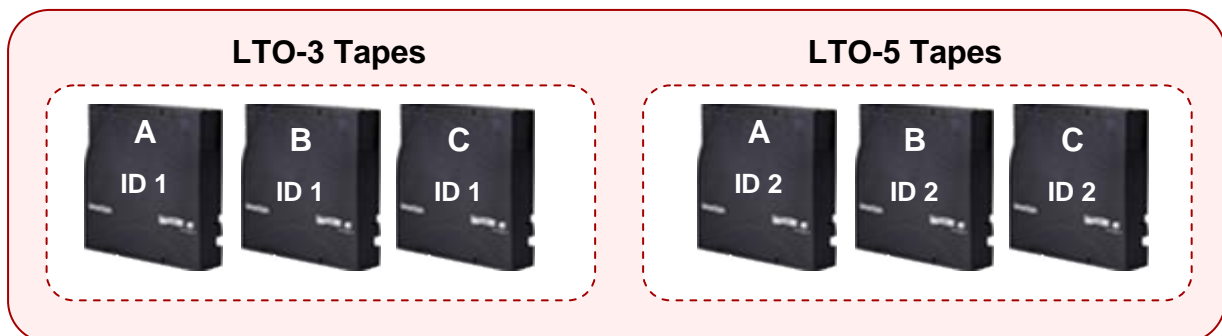
- The **tapelist.txt** is a file listing the barcodes of the tapes to be migrated. In the case of a technology migration, it will contain all LTO-3 barcodes of Tape Group Main (*the Oracle Technical Team can provide the tools to generate such listings automatically*).
- The **-media Main** tells DIVAmigrate to copy the objects to **Tape Group Main**, which happens to be the same group as the source data.
- The **-count 2** is important because DIVAmigrate is replicating to the same Destination Group as the Source Group. Therefore, two instances of each object at the end of the run will be present instead of just one.

If the user happens to specify **-count 1** instead of **-count 2**, DIVAmigrate will verify that one instance exists for each object. Since this is already the case, DIVAmigrate will consider the job complete and do nothing (*which is a good way to perform a dry run with the DIVAmigrate utility for a quick sanity check*).

Here is the final state after the run has completed:

Figure 17: LTO-3 to LTO-5 Migration Example – Final State after Copy

Tape Group Main



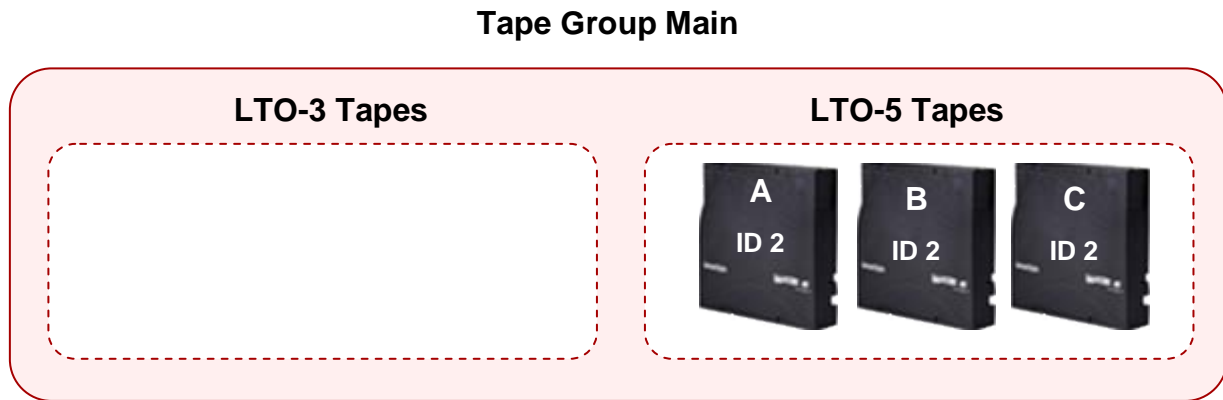
Use the following command in order to move the data instead of replicating it:

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

- The **-delete** tells DIVAmigrate to delete the original instances after they have been copied.
- The **-count 1** tells DIVAmigrate that only one instance of each object at the end of the run is desired.

Using the above command results in the following final state:

Figure 18: LTO-3 to LTO-5 Migration Example – Final State after Move



Note: Because the DIVAmigrate Utility is able to skip unnecessary tasks (*it has already been stated that it will not perform a copy if the required Instance Count is already present in the Destination Group/Array*), the two example commands above could be used one after the other. The first command will replicate the data while the second command will just do the deletes (*DIVAmigrate will realize the copies are already done and skip that step*).

This two-step migration is sometimes used by customers who don't initially have faith in their new technology tape drives. They prefer to keep the old instances temporarily for safety and use the replicate command as a first step. After they've been using the new technology tapes and drives for a period of time and are satisfied with their functionality and stability, they will use the move command to delete the old instances.

3.5.4 Stopping and Resuming Jobs

An interesting aspect of the DIVAmigrate Utility is that it only performs the steps necessary to reach the intended goal and nothing more. Any time a migration job is started DIVAmigrate will examine the DIVArchive Catalog and calculate which copy and delete operations remain incomplete. It doesn't 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, users may freely interrupt a DIVAmigrate job at any time and resume it later on. There are no specific actions to be completed before resuming an interrupted job.

3.6 Advanced Migration

3.6.1 Speeding Up Tape-To-Tape Migration Using a Disk Buffer

The DIVAmigrate Utility processes objects sequentially, one after the other. For example, if an object is copied from a Tape 1 to a Tape 2, a Copy Request will be sent to the Manager. The Manager will allocate some Cache Disk Space to store the object and proceed in two steps:

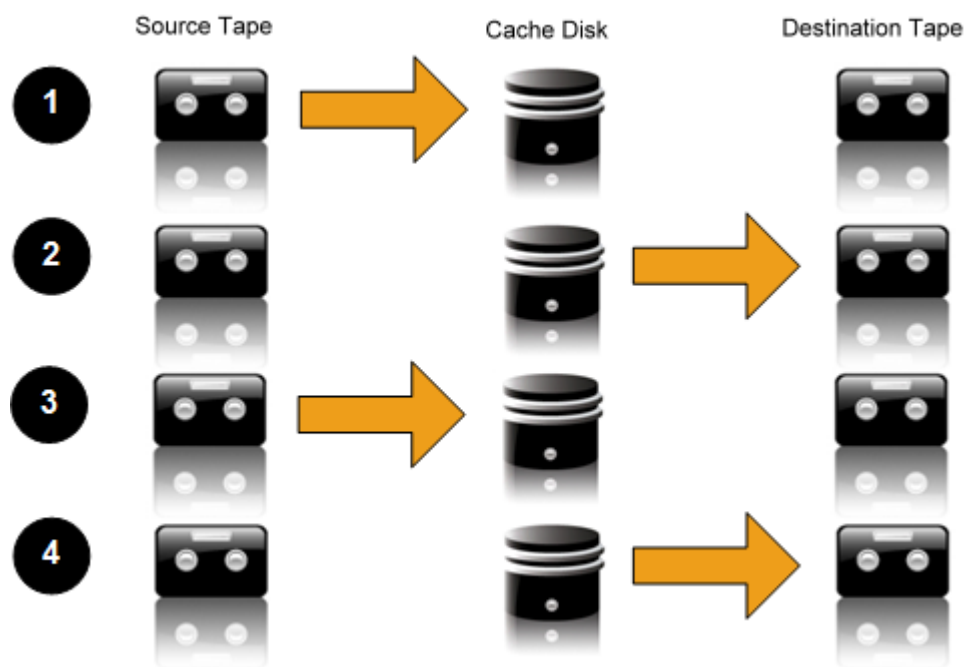
1. Read the object from Tape 1 and write it to the Cache Disk.
2. Read the object from the Cache Disk and write it to Tape 2.

DIVAmigrate also often processes series of objects, for example migrating a Tape Volume to another Tape Group:

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.

Figure 19: Workflow for Processing a Series of Objects



The data could possibly be written to more than one tape in the Destination Tape Group (*based on the activity and Manager workflow*), but for this example consider that everything goes to a single Tape 2.

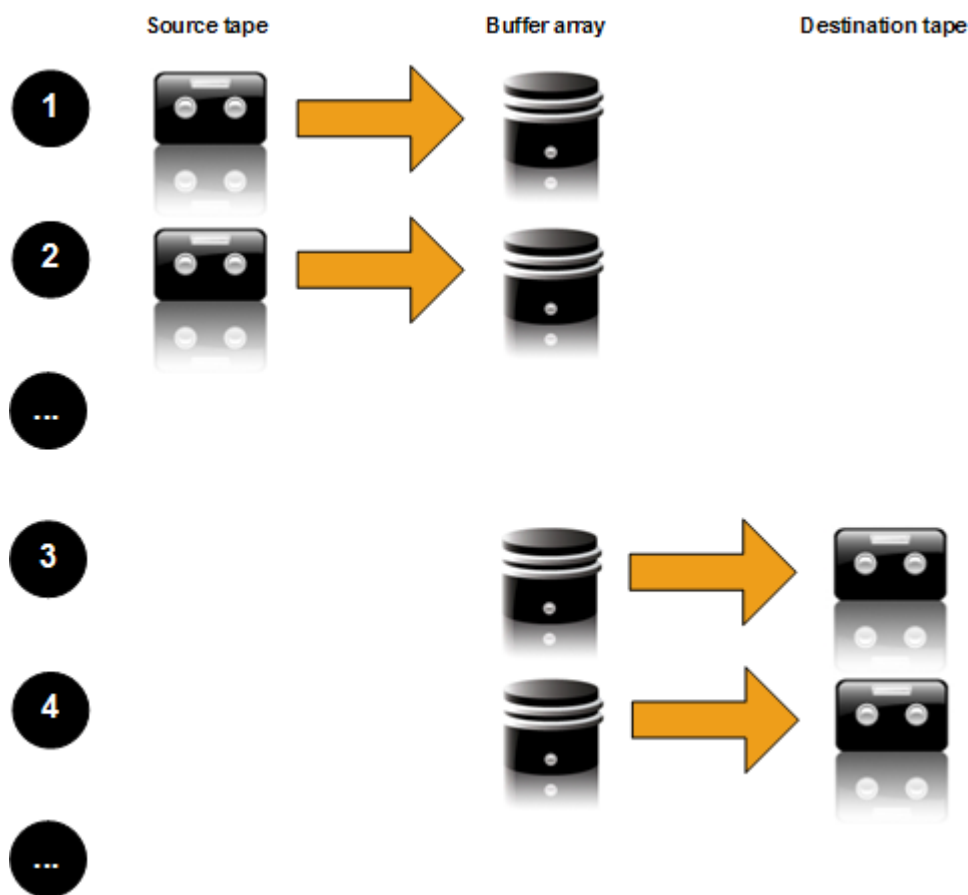
During Step 2 above, the Source Drive is idle and can be assigned to other requests, so the possibility exists that Tape 1 will be dismounted and another tape mounted. In this case, Tape 1 will need to be mounted and positioned again in Step 3 (*which will probably require dismounting a tape from the Source Drive beforehand*). This will cause a substantial delay in operations. While Step 3 proceeds, there is also a possibility that Tape 2 gets dismounted because the Destination Drive is now idle. So it may need to be mounted and positioned again in Step 4.

A massive migration using Tape-To-Tape copies can result in a tremendous dismount/mount/positioning overhead. For this reason, it is recommended that heavy migration operations are conducted on dedicated Oracle DIVArchive Actors with dedicated Tape Drives.

Even with dedicated Tape Drives and no mount/dismount overhead, it may be observed that one drive is idle while the other is working (*the Target Drive is idle in Steps 1 and 3 above, and the Source Drive in steps 2 and 4, etc.*).

To suppress Drive Idle Time a Disk Array can be used as a buffer for the Migration Job. When a Disk Buffer is specified, DIVAmigrate first enters a Read Phase where 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:

Figure 20: Using a Disk Array as a Buffer



The benefits are clear - only one drive is used in each phase, and it is used continuously. There is still a possibility that the drive will get dismounted between two Copy Requests and assigned to a more prioritized request from the Manager Queue, but this won't 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 being that DIVAmigrate creates standard Object Instances on a Disk Array and has to explicitly delete them at the end of the run; Repack creates temporary instances on a Cache Disk that are cleaned by the Manager at the end of the request. This is the main reason why a restart of an aborted Migration Job needs to be restarted with the exact same command line, and in particular the exact same Buffer Array used in the previous run. If arrays are switched between two runs, the second run will overlook the buffered instances from the first run and recreate new instances in the new array resulting in:

- Wasting time recreating Disk Instances that were already created in the previous array.
- Wasting space in the previous array because the instances created there aren't managed by a Migration Job any longer and will never get deleted.

The following is a sample Migration Command using a Disk Buffer:

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

Scenario:

Tape 000001 is an LTO-3 Tape from **Group Main** and needs to be migrated to **Tape Group Main-LTO-5** using the **NAS001 Array** as a buffer.

Tape 000001 contains Objects A and B. The following is the action sequence performed by DIVAmigrate (*assuming the objects don't 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 **Object A**'s instance on **Array NAS001**.
5. Copy **Object B** from **Array NAS001** to a tape in **Tape Group Main-LTO-5**.
6. Delete **Object B**'s instance on **Array NAS001**.

According to the workflow described above, 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; i.e. after the target copies of the related object are complete.

To delete instances from the Source Tape after migration simply add the `-delete` option as shown:

```
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 **Object A**'s instance on **Array NAS001**.
5. Delete **Object A**'s instance on **Tape 000001**.
6. Copy **Object B** from **Array NAS001** to a tape in **Tape Group Main-LTO-5**.
7. Delete **Object B**'s instance on **Array NAS001**.
8. Delete **Object B**'s instance on **Tape 000001**.

The workflow is the same as before except that this time, after deleting the Buffered Instance, the Source Instance is also deleted. At the end of the migration, if there were no errors, **Tape 000001** will be empty.

Warning:

- **The Array used as a disk buffer must be chosen carefully. There are two very important rules to consider:**
 - **Make sure that the Array isn't managed by the Storage Plan Manager. When an Array is managed by SPM, the data stored there by DIVAmigrate may be replicated to tape and deleted automatically, which will hinder DIVAmigrate operations and end up with undesired tape instances resulting in a waste of tape storage space.**
 - **Make sure the Array being used isn't a repository containing resident instances of the objects to be migrated. If it is, those instances will be considered by the utility to be Buffered Instances from a previous run and will be deleted at the end of the run.**

3.6.2 Creating Multiple Instances in the Target Group or Array

DIVAmigrate is capable of creating any number of instances in the target Group/Array using the `-count` option. As previously discussed, DIVAmigrate will perform 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, so if a count greater than 1 is specified, the Destination Tape Group or Disk Array must have more than one Tape Volume or Disk Unit identified within it.

If a Disk Buffer is used (`-buffer` option), the instances will be buffered once only no matter how many instances are to be created in the Destination Group/Array.

The following command:

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


Translates into the following sequence - assuming **Backup** is a **Tape Group**, **Tape 000001** contains Objects A and B, and these objects don't have instances in **Group Backup**:

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 of **Tape Group Backup**.
4. Copy **Object B** from **Array NAS001** to a tape of **Tape Group Backup**.
5. Copy **Object A** from **Array NAS001** to another tape of **Tape Group Backup**.
6. Delete **Object A**'s instance on **Array NAS001**.
7. Copy **Object B** from **Array NAS001** to another tape of **Tape Group Backup**.
8. Delete **Object B**'s instance on **Array NAS001**.

Note that the two instances of each object in the **Backup Group** were created separately. It does the first copy of Objects A and B, then the second copy, instead of doing two copies of A in a row, then two copies of B. The reason for that is that there is only one tape drive to work with. If the same object is copied twice to the Destination Tape Group, the Manager will need to use a different tape for each instance because by design DIVArchive doesn't allow two instances of the same object to be stored on the same volume. This will cause a dismount/mount operation.

By performing the first copy of Objects A and B in a row, the same tape may be kept on the drive for both objects, and the tapes only need to be changed 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 a second copy to the same group. The DIVAmigrate Utility always tries to do as many objects as possible in a row within a particular migration step in order to minimize the mount/dismount overhead.

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

3.6.3 Migrating to Multiple Groups or Arrays

More than one **-media** option may be used for DIVAmigrate to copy data to more than one Group or Array. Each **-media** option must be accompanied by a **-count** option. The default count is 1.

Example:

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

If a Disk Buffer is used (**-buffer** option) the instances will be buffered only once no matter how many Destination Groups/Arrays are specified.

The following example command:

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

translates in the following sequence - assuming **Backup** and **Ext** are Tape Groups, **Tape 000001** contains Objects A and B, and these objects don't have instances in any Destination 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 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 **Object A**'s instance on **Array NAS001**.
9. Copy **Object B** from **Array NAS001** to a tape in **Tape Group Ext**.
10. Delete **Object B**'s instance on **Array NAS001**.

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

- 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/dismount overhead. If it didn't do so, 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.

3.6.4 Default Target Instance Count

Omitting the Destination Instance Count Option (*-count*) will cause DIVAmigrate to use a default value of 1.

The examples below all assume migration of the data from a **Tape 000001** belonging to **Tape Group Main**:

```
client -tape 000001 -media Backup
```

- The Destination Count is set to 1 by default, so this asks DIVAmigrate to create an instance of each object from **Tape 000001** in **Tape Group Backup**; replicating **Tape 000001**'s data.

```
client -tape 000001 -media Main
```

- This 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 will consider the job already complete and do nothing. 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 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 asked to delete the Source Instances, so DIVAmigrate can't include the Source Instance in the final count like it would in the previous example. DIVAmigrate will therefore create an additional instance of each object in **Tape Group Main** before deleting the original one on **Tape 000001**. The result is similar to a Repack Operation.

3.6.5 Repacking Tapes

To repack a suspected corrupt tape, DIVAmigrate may be more effective than a Repack Operation because it doesn't 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 (*copy to **Destination Group** then delete the original instance on the **Source Tape***).

The `-count 1` tells DIVAmigrate that once the migration has completed (*which means after the copies and the instance deletions are done*) there should one instance of each object in **Tape Group Main** remaining.

The `-recovery` option will cause DIVAmigrate to look for any alternative online instance, either in disk or tape, and if available, it will be used. This option should be used if the source tape is known to be corrupt. If no alternative is found, DIVAmigrate will still try the instance from the source tape.

The DIVAmigrate Utility will copy each Source Instance to the same group and delete the source instance afterwards - this is more or less what a Repack Operation does.

There are, however, a few differences with Repack:

- Repack is a single request in the Manager's Queue. An equivalent DIVAmigrate run will produce 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 entirely moved. However, if it is spanned, the Repack will only move the portion of the instance that resides on the repacked tape and not the other portions residing

on other tapes. DIVAmigrate will move the whole instance because it uses the standard `copy` function (*which makes DIVAmigrate a useful tool to de-span instances*).

- Repack always writes the Source Tape's content to a single Destination Tape. DIVAmigrate uses the standard `copy` function and the only guarantee is that the data will land on the requested Destination Group, but not necessarily on a single tape.

3.6.6 De-spanning 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 (*e.g. Direct Restore from tape may experience a timeout on the Source/Destination side during tape remounting*).

To de-span a series of instances, use the command:

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

The Oracle Technical Team can provide users with a utility to generate an Instance List File containing the list of instances spanned in the system.

Note: Modern tape units have capacities pushing the limits of technology and tend to suffer from random reduced capacity because of intermittent bad block writes. Because of this, the EOT (*End of Tape*) marker may be prematurely encountered - defeating DIVArchive's Remaining Tape Space calculations and causing accidental spanning.

3.6.7 Using Alternate Source Instances

DIVAmigrate will always use the Source Instances specified by the user to do the destination copies (*or the buffer copy if a Disk Buffer is used*) by default:

- If an Object or Object List is used as the Source Specifier (*-tape or -tapelist option*) it will use the instances located on these tapes.
- If an Instance List is used (*-instlist option*) it will use the instances listed in the file.
- If an Object List is used (*-tapelist option*) it will compute an optimal instance list and use that.

There are situations where letting DIVArchive chose the Source Instance is desirable. For example, you are migrating data off a suspicious tape and there is a possibility that some objects still have instances on a storage disk. By default, DIVAmigrate will read the Source Instances from tape. If the user wants DIVAmigrate to let the Manager chose the Source Instance (*which will select the Disk Instance as a source if there is one*) use the `-autosrc` modifier as follows:

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

Notes:

- The `-delete` option always deletes the Source Instances obtained from the Source Specifier in the command line (`-tape`, `-tapelist` or `-instlist` option) no matter which instance of a particular object was actually selected as a Copy Source by the Manager. For example, if `-tape 000001` is specified on the command line with the `-autosrc` option, and one of the objects has a Disk Instance, this object will be copied from disk not from tape; however the instance deleted at the end of the run will still be the one located on Tape 000001.

3.6.8 Excluding Objects from Migration

If repeated Read Errors of some Source Instances are observed, and it is not desired to delete them at this point, they may be excluded from the migration process by using the `-excl` option:

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

This command will ignore all instances belonging to the Objects listed in the file named `file.txt`. This file lists objects to be excluded in the form `object|category` as shown below:

```
Clip001|HD
```

```
Clip002|SD
```

4 Migration Error Handling and Failure Scenarios

This section describes error handling and different failures which may occur during Migration and DIVAmigrate's response to those failures.

DIVAmigrate performs error handling based on following process:

- If the configured number of requests fails in a row, the Migration Job tries to switch to another Migration Destination. If there are no other Destinations the job is paused.
- After processing all other Migration Destinations, the Migration Job will be paused.
- If, after failing for the configured number of requests for a Destination, the Migration Job switches to another Destination and will try to process it until the configured number of requests fails in a row. Upon this failure, it will try to switch to the next Destination (*if any exist*). Once all Destinations have been exhausted, the job will pause (*if no more destinations exist*).
- The only exception to this rule is requests that have failed because of a *Too many running requests* error - these requests are always retried.

In general DIVAmigrate will not retry migration for objects. DIVAmigrate will store the error code(s) for each object and move on to next object that needs to be migrated. The user can rerun the Migration Job later and it should affect only new objects on the Source and those which failed during the previous run.

The table below describes specific scenarios:

Table 6: DIVAmigrate Failure Scenarios

Issue	DIVAmigrate Action	Suggested User Action
Lost connection to DIVArchive Manager.	Tries to reconnect until connection is restored.	If the Manager is down, restart the Manager. If DIVAmigrate can't connect to the Manager the user will notice it by new Migration Jobs being too long in the Submitted state and not changing to the Pending state, and by checking the DIVAmigrate Logs. At that point user should investigate common problems with the connection.
Can't connect to the Manager Database.	Retry connection 3 times every 10 seconds, if not successful then quit.	The user should investigate and resolve common database connectivity issues and restart the DIVAmigrate service.

Issue	DIVAmigrate Action	Suggested User Action
<p>Environment on which DIVAmigrate is running crashes.</p>	<p>After restart, DIVAmigrate will check statuses of each job in the database and if they are not expired it will continue their processing. Otherwise DIVAmigrate will recreate their Migration Plan and resume job processing as soon as free resources are available.</p> <p>Job is considered expired if its last access time by the Migrate Service is greater than the period of time specified in the service configuration.</p>	<p>The user should resolve the environment issue and restart DIVAmigrate.</p>
<p>For an object on the Migration Source there are no Online Instances.</p>	<p>DIVAmigrate will check for Online Instances only during the Initialization phase. If an object doesn't have such instances it will skip it and assign the correct error code to it in the database.</p> <p>If during the Initialization phase the object was online but later became offline, then some requests sent to the Manager will fail. The Error Code will be stored in the Database and the request will not be retried. If the maximum requests continuously fail, the job will pause, or if the job has Multiple Sources or Destinations, it will switch to the next S/D.</p> <p>For Alternative Instances, if their migration fails, the original instances will be used instead.</p>	<p>The user will be able to browse the results of migration for each finished Migration Job. These results will contain a list of all objects eligible for migration along with error codes (<i>if any</i>). The user should make objects accessible online again and rerun the Migration Job.</p>

Issue	DIVAmigrate Action	Suggested User Action
User performs migration from Tape Group that has offline tapes	<p>If the Alternative Sources option is on, DIVAmigrate will try to find Disk Instances for all offline instances and migrate them.</p> <p>If recovery mode is on, DIVAmigrate will try to find disk or tape instances for Offline Source Instances and migrate them.</p> <p>If for some offline instances there are no Alternative Instances found, they will be skipped from migration and an Object Offline error code will be assigned to the objects.</p> <p>If the tape goes off line during migration, the job will pause after the maximum sequential failure requests. If the job has multiple Sources or Destinations, it will switch to the next one and pause after processing them.</p>	The user will be able to browse the results of migration for each finished Migration Job. These results will contain a list of all objects eligible for migration along with error codes (<i>if any</i>). The user should make objects accessible online again and rerun the Migration Job.
Migration Job was submitted with invalid parameters.	DIVAmigrate will assign the Aborted status to a job.	The user should check the parameters and recreate a new Migration Job with correct parameters.
User deletes un-migrated object(s) from the Source during a migration procedure.	DIVAmigrate will try to migrate the object(s) and will store the appropriate error code(s) in Database.	The user should Cancel or Delete the Migration Job using the DIVArchive Control GUI.
User realizes he submitted a Migration Job with incorrect parameters.	N/A	The user should Cancel or Delete the Migration Job using the DIVArchive Control GUI.
There is not enough free space on the Destination.	DIVAmigrate will fail the configured number of requests in a row and after that place the job in the Paused state.	The user should resolve the issue with the Destination and resume the job.
All Migration Destination medias go offline.	DIVAmigrate will fail the configured number of requests in a row and after that place the job in the Paused state.	The user should resolve the issue with the Destination and resume the job.

Issue	DIVAmigrate Action	Suggested User Action
<p>Source Tape List, Instance List, Exclude List has invalid values but the syntax is correct.</p> <p>For example an invalid tape barcode.</p>	<p>DIVAmigrate will ignore invalid values. If all of source is invalid job must abort.</p>	

APPENDIX

A1 *Default Configuration*

```
# 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
```

```

# DIVArchive Manager Password for connecting to the database.  This is
# case sensitive.
# Default value is lib5.
# This is a mandatory parameter.
DIVAMANAGER_DBPASSWORD=lib5

# 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 System Identifier (SID) in Oracle
# where DIVArchive Manager connects.
# Typically lib5 in most DIVArchive installations.  Consult your
# delivery plan if you are not sure.
# Default value is lib5.
# This is a mandatory parameter.
DIVAMANAGER_DBSID=lib5

# 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/.
```