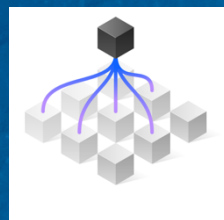


Db2 Shift

Modernizing Db2 Linux Installations

Version 2 Usage Guide



George Baklarz and Phil Downey

Db2 Shift

Copyright © 2022 by International Business Machines Corporation (IBM). All rights reserved. Printed in Canada. Except as permitted under the Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of IBM, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

The contents of this book represent those features that may or may not be available in the current release of any products mentioned within this book despite what the book may say. IBM reserves the right to include or exclude any functionality mentioned in this book for the current release of Db2 Shift V2, or a subsequent release. In addition, any claims made in this book are not official communications by IBM; rather, they are observed by the authors in unaudited testing and research. The views expressed in this book is those of the authors and not necessarily those of the IBM Corporation; both are not liable for any of the claims, assertions, or contents in this book.

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice and at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future feature or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results like those stated here.

U.S. Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM. Information in this eBook (including information relating to products that have not yet been announced by

IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IN NO EVENT SHALL IBM BE LIABLE FOR ANY DAMAGE ARISING FROM THE USE OF THIS INFORMATION, INCLUDING BUT NOT LIMITED TO, LOSS OF DATA, BUSINESS INTERRUPTION, LOSS OF PROFIT OR LOSS OF OPPORTUNITY.

IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs, or services available in all countries in which IBM operates or does business.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks, or other intellectual property right.

IBM, the IBM logo, ibm.com, Aspera®, Bluemix, Blueworks Live, CICS, Clearcase, Cognos®, DOORS®, Emptoris®, Enterprise Document Management System™, FASP®, FileNet®, Global Business Services®, Global Technology Services®, IBM ExperienceOne™, IBM SmartCloud®, IBM Social Business®, Information on Demand, ILOG, Maximo®, MQIntegrator®, MQSeries®, Netcool®, OMEGAMON, OpenPower, PureAnalytics™, PureApplication®, pureCluster™, PureCoverage®, PureData®, PureExperience®, PureFlex®, pureQuery®, pureScale®, PureSystems®, QRadar®, Rational®, Rhapsody®, Smarter Commerce®, SoDA, SPSS, Sterling Commerce®, StoredIQ, Tealeaf®, Tivoli®, Trusteer®, Unica®, urban{code}®, Watson, WebSphere®, Worklight®, X-Force® and System z® Z/OS, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.

Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

All trademarks or copyrights mentioned herein are the possession of their respective owners and IBM makes no claim of ownership by the mention of products that contain these marks.

Initial Publication: April 29th, 2022

Revisions

Initial Publication: April 29th, 2022

Update: June 3rd, 2022

- Added links for downloading technical preview code
- Updated requirements for OpenShift client
- New details on create database option on shift to instance
- Changed help key definition for all panels (^F)
- New modifier key for Page Up/Page Down for OSX workstations
- New chapter on Best Practices

Update: June 12th, 2022

- Updated text to include menu shortcuts

Update: August 9th, 2022

- New installation instructions
- Additional platform support
- Updated instructions and Best Practices

Update: October 4th, 2022

- New syntax options for HADR and SETTINGS
- New mirror option
- Additional best practices

About the Authors

George Baklarz, B. Math, M. Sc., Ph.D. Eng., has spent many years at IBM working on various aspects of database technology. George has written 14 books on Db2 and other database technologies.

Phil Downey, CompSci and Psychology, has over 30 years' experience in the IT industry and 25 years' experience working with enterprise database systems across many different industries in several different technical and product management roles. He is experienced in designing and deploying enterprise data architectures and migrating existing applications to them. He is one of the inventors of the Click to Containerise technology.

Table of Contents

Revisions.....	iv
About the Authors	v
Table of Contents	vi
About this Book	i
Chapter 1: Introduction.....	2
Chapter 2: Pre-Requisites	5
Target Users	5
Db2 Shift Program Requirements	5
Connectivity Requirements.....	6
Userid and Authentication Requirements	7
Target Database Requirements	8
Instance Owner and DBADM and SECADM Userid.....	9
Chapter 3: Installation	11
Program Download.....	12
Chapter 4: Db2 Shift Syntax.....	14
Syntax Summary	15
Db2 Shift User Interface	16
Enabling UI Mode	17
Menu System Overview.....	19
Interacting with the UI.....	19
<i>Text Input</i>	21
<i>Numeric Input</i>	21
<i>Long Fields</i>	21
<i>Options</i>	21
<i>Check Boxes</i>	22
<i>Lists</i>	22
<i>Cut and Paste</i>	23
Disclaimer	24
Help Panels.....	25
Main Screen	27
Executing Commands	28
Summary Screen	29

Contents	vii
Db2 Shift Execution	30
Database Analysis	32
<i>Overriding Parameters</i>	35
Local Analysis	35
Chapter 5: Shift Scenarios.....	38
Db2 Shift Command	38
Db2 Containerization to OpenShift, Kubernetes, or Cloud Pak for Data	39
<i>Required Options</i>	39
<i>Optional Settings</i>	39
Db2 Shift to another Db2 Instance	39
<i>Required Options</i>	40
<i>Optional Settings</i>	40
Clone an Existing Database.....	40
<i>Required Options</i>	41
<i>Optional Settings</i>	41
Clone Deployment to OpenShift, Kubernetes, or CP4D	41
<i>Required Options</i>	41
<i>Optional Settings</i>	41
Clone Deployment to Another Db2 Instance	42
<i>Required Options</i>	42
<i>Optional Settings</i>	42
Initialize HADR between Source and Target POD	42
<i>Required Options</i>	43
<i>Optional Settings</i>	43
Initialize HADR between Source and Target Instance.....	43
<i>Required Options</i>	43
Initialize DMC and LDAP Authentication for CP4D.....	44
<i>Required Options</i>	44
<i>Optional Settings</i>	44
Clone Copy.....	44
<i>Required Options</i>	45
<i>Optional Settings</i>	45
Chapter 6: Db2 Containerization to OpenShift, Kubernetes, or Cloud Pak for Data	47
Required Options.....	47
Optional Settings.....	48
Mode Option	48
<i>Move Database Only</i>	49
<i>Move Database and Set up LDAP and DMC Settings</i>	49
Target Client (Instance to POD)	49
Source Database	49
Source or Instance Owner	50

Contents	viii
Destination Database	50
Destination Server (POD)	50
Stored Procedures and Functions	51
Destination Pod Namespace or Project	51
HADR Setup	51
Synchronization Options	52
Metadata Generation	53
<i>Blank Slate</i>	54
<i>Generate Settings</i>	54
<i>Verification</i>	54
Online or Offline Move.....	55
Performance: Threading	56
Performance: Compression	56
Chapter 7: Db2 Shift to Another Db2 Instance.....	58
Required Options.....	58
Optional Settings.....	59
Mode Option	59
Target Client (Instance to Instance).....	60
Source Database	60
Source or Instance Owner	60
Destination Database	60
Force Destination Database Creation	61
Mirror Path.....	61
Destination Server (Instance).....	62
Stored Procedures and Functions	62
HADR Setup	62
Synchronization Options	63
Metadata Generation	64
<i>Blank Slate</i>	65
<i>Generate Settings</i>	65
<i>Verification</i>	65
Online or Offline Move.....	66
Performance: Threading	67
Performance: Compression	67
Chapter 8: Clone an Existing Database.....	69
Required Options.....	69
Optional Settings.....	69
Mode Option	70
Source Database	70
Source or Instance Owner	71
Clone Directory.....	71
Stored Procedures and Functions	71

Synchronization Options 72

Metadata Generation 73

Blank Slate 74

Generate Settings 74

Verification 75

Online or Offline Move..... 75

Performance: Threading 76

Chapter 9: Clone Deployment to OpenShift, Kubernetes, or CP4D

..... 78

Required Options..... 78

Optional Settings..... 78

Mode Option 79

Target Client (Instance to POD)..... 80

Source Database 80

Source or Instance Owner 80

Clone Directory..... 81

Destination Database 81

Destination Server (POD) 81

Destination Pod Namespace or Project..... 81

Stored Procedures and Functions 82

HADR Setup 82

Synchronization Options 82

Metadata Generation 84

Blank Slate 84

Generate Settings 85

Verification 85

Online or Offline Move..... 85

Performance: Threading 87

Performance: Compression 87

Chapter 10: Clone Deployment to Another Db2 Instance 89

Required Options..... 89

Optional Settings..... 89

Mode Option 90

Target Client (Instance to Instance)..... 91

Source Database 91

Source or Instance Owner 91

Clone Directory..... 92

Destination Database 92

Force Destination Database Creation..... 92

Mirror Path..... 92

Destination Server (Instance) 93

Stored Procedures and Functions 93

Contents	x
HADR Setup	94
Synchronization Options	94
Metadata Generation	96
<i>Blank Slate</i>	96
<i>Generate Settings</i>	97
<i>Verification</i>	97
Online or Offline Move.....	97
Performance: Threading	98
Performance: Compression	98
Chapter 11: Initialize HADR between Source and Target POD ...	101
Required Options.....	101
Optional Settings.....	102
Mode Option	102
Target Client (Instance to POD).....	103
Source Database	103
HADR Source and Destination Server.....	103
HADR ports	104
Destination Server (POD)	104
Destination Pod Namespace or Project.....	104
Chapter 12: Initialize HADR between Source and Target Instance	106
.....	106
Required Options.....	106
Mode Option	107
Target Client (Instance to Instance).....	107
Source Database	108
Destination Server (Instance).....	108
HADR Source and Destination Server.....	108
HADR ports	109
Chapter 13: Initialize DMC and LDAP Authentication for CP4D.	111
Required Options.....	111
Optional Settings.....	111
Mode Option	112
Target Client	113
Source or Instance Owner	113
Destination Database	113
Destination Server (POD)	113
Destination Pod Namespace or Project.....	114
Chapter 14: Clone Copy	116
Required Options.....	116
Optional Settings.....	116
Mode Option	117

Source Clone Directory	117
Target Clone Directory	117
Target Client	118
Destination Server (POD)	118
Destination Pod Namespace or Project	118
Chapter 15: Best Practices	120
Database Migration	120
Online Versus Offline Mode	121
<i>Online Mode</i>	121
<i>Offline Mode</i>	121
Instance Ownership	122
Target Db2 Environment	122
<i>Cloning into a Containerized Environment</i>	123
<i>Cloning into a Traditional Db2 Instance</i>	123
HADR Considerations	124
<i>HADR to CP4D POD</i>	124
<i>HADR using a Clone Copy</i>	125
Database Storage Relocation	125
Force Database Creation and Mirror Paths	128
<i>Mirror Path (Only)</i>	129
<i>Mirror Path and Force Database Creation</i>	129
Encrypted Databases	130
Database, Instance, and Environment Settings	130
Threads and Compression	130
<i>Threads</i>	131
<i>Test Scenario</i>	131
<i>Compression</i>	134
<i>Test Scenario</i>	134
Appendix A: Options Reference	137
Mode Option	137
<i>Move Database Only</i>	137
<i>Clone a Database</i>	138
<i>Apply a Clone to a POD or Instance</i>	138
<i>Copy a Clone from or to a Pod</i>	138
<i>HADR Setup</i>	138
<i>LDAP Security and DMC Setup</i>	139
<i>Copy Mode</i>	139
Target Client Instance to Instance	139
Target Client Instance to POD	139
Source Database	140
Source or Instance Owner	140
Destination Database	140

Force Destination Database Creation 141

Mirror Path 141

Destination Owner 142

Destination Server POD 142

Destination Server Instance 142

HADR Setup 143

HADR Source or Destination Server 143

HADR ports 143

Destination Pod Namespace or Project 144

Clone Directory 144

Source Clone Directory 144

Target Clone Directory 145

Synchronization Options 145

Metadata Generation 147

Blank Slate 147

Generate Settings 148

Verification 148

Online or Offline Move 148

Threading 149

Compression 149

Stored Procedures and Functions 149

Overrides 150

Appendix B: Additional Resources for Db2 152

International Db2 User Group (IDUG) 152

Support 152

About this Book

Deploying an existing Db2 to OpenShift, Kubernetes, or Cloud Pak for Data usually involves some form of export and import and a lot of work! The new Db2 Shift tool moves your on-premises database directly to the Cloud - with no exporting of data!

Db2 Shift can migrate your 10.5, 11.1 and 11.5 database directly into a Db2 container with no additional effort. This eBook will take you through the IBM Db2 Shift program and how it can help modernize your Db2 databases quickly and efficiently!

Coverage Includes:

- How Db2 Shift works and what environments it is suitable for
- The installation process
- Overview of the command line and User Interface
- A summary of the scenarios that Db2 Shift can be used for
- Detailed instructions on every type of Shift
- Appendix with detailed explanations of all settings

George and Phil

How This Book Is Organized

We organized this book into 14 chapters that cover many of the highlights and key features of Db2 Shift.

- Chapter 1: Overview - An introduction to what Db2 Shift is used for
- Chapter 2: Pre-Requisites - Some of things that you will need to have before trying to shift a database
- Chapter 3: Installation - Installation instructions and O/S requirements
- Chapter 4: Db2 Shift Syntax and UI - How to run the Db2 Shift command and interact with the User Interface
- Chapter 5: Shift Scenarios - A summary of the various Shift Scenarios
- Chapter 6: Shift a Db2 database to OpenShift, Kubernetes or CP4D
- Chapter 7: Shift a Db2 database to another Db2 instance
- Chapter 8: Create a Cloned copy of the Db2 database for later deployment
- Chapter 9: Deploy a clone into an OpenShift, Kubernetes or CP4D container
- Chapter 10: Deploy a clone into another Db2 instance
- Chapter 11: Initialize HADR between Source and Target POD
- Chapter 12: Initialize HADR between Source and Target Instance
- Chapter 13: Initialize DMC and LDAP Authentication for CP4D
- Chapter 14: Copy Cloned databases to a POD
- Chapter 15: Best Practices
- Appendix A: Command Details
- Appendix B: Additional Db2 Resources

Chapter 5 summarizes the Shift Scenarios. Determine which type of Shift you want to use and then read the details in the chapter assigned to that Shift type.

1

Introduction

MODERNIZING YOUR DB2 LANDSCAPE

Chapter 1: Introduction

The Db2 Click to Containerize family encompasses several tools that provide customers with the ability to quickly modernize their Db2 landscape. The Db2 Shift utility is part of the Click to Containerize family and can be used to clone a copy of Db2 into an OpenShift, Kubernetes, Cloud Pak for Data (CP4D), or a standard Db2 instance.

The utility is intended to help customers move their existing Db2 databases on Linux into a containerized environment with the minimum amount of effort. Some benefits of Db2 Shift are:

- Automated, fast, and secure movement of Linux databases to Hybrid Cloud
- Massively reduces time to containerize database workloads
- Enables alignment with Agile Delivery and Project Lifecycle with Cloning capabilities once in cloud

Features of Db2 Shift include:

- The ability to move a database without the need to unload, export, decrypt, or backup the database
- Automatic upgrades from Db2 Version 10.5 to the latest version (11.5.8) of Db2
- Shifting of all database settings and objects, including external functions located in the Db2 library path
- Row, Columnar, and Encrypted databases can be moved
- OLTP, SMP, and MPP databases can be moved (excluding pureScale installations at this time)
- Easy setup of HADR servers for staged migration

The Db2 Shift program allows a customer to shift their current databases to one of four platforms:

- OpenShift cluster
- Kubernetes cluster
- Cloud Pak for Data
- Another Db2 instance on premise, on Cloud, or in a Virtual Environment

In addition to directly shifting the database from one location to another, the Db2 Shift program also provides the ability to clone a database for future deployment. This feature is useful for environments where the target server is air-gapped, or unavailable for direct connection from the source server.

Finally, the Db2 Shift program has two modes of operation. For expert users, the Db2 Shift command can be issued with the appropriate options and run directly from a command line or a script. For those users who require more help, the

Chapter 1: Introduction

program can also be run in an interactive mode, with detailed instructions and help for the various shift scenarios.

In summary, Db2 Shift provides the ability to quickly, and easily, shift your Db2 Linux database into a containerized environment with a minimal amount of effort.

2

Pre-Requisites

SUPPORTED ENVIRONMENTS

Chapter 2: Pre-Requisites

There are several pre-requisites for running the Db2 Shift program itself, as well as requirements related to the Db2 database and the target Db2 system. The Db2 Shift program will analyze the source database and prevent any shifts from occurring that will not work because of the target database environment.

An additional analysis feature is provided which will determine if any INSTANCE parameters on the source system need to be moved to the TARGET system. The database can be shifted without the corresponding target INSTANCE settings being modified, but it may affect the behavior of the database when it is running in the new environment.

Target Users

The Db2 Shift program has been designed for experienced DBAs or casual users through the GUI interface. It is more likely that the user interface will be used for analysis and initial testing while production use will use the command line version.

The user requires intimate knowledge of current INSTANCE settings to make sure the containerized environment has sufficient resources to run the workloads. Further details of what is required to run the Db2 Shift program is described below.

Db2 Shift Program Requirements

The Db2 Shift program is a single executable Linux program that can be installed in any directory. The only requirement is that the program is marked as executable and is accessible to the Db2 INSTANCE owner.

The program itself is self-contained and does not require any additional libraries to run. The program can be removed by simply deleting the file. When the program executes, it will generate temporary files that are used during the shift process. It is recommended that the program be run from within a directory so that all files generated can be easily found.

The program itself requires only 8M of space and sufficient disk space for the log and control files that are generated.

The following Linux environments have been tested as source systems:

- Linux X64, CentOS (6,7,8), CentOS Stream, Red Hat (6,7,8) Ubuntu (18.04, 20.04, 22.04), SUSE 15, openSUSE 15
- Linux PPC (PowerPC) Little Endian has been tested as a source and target location

Connectivity Requirements

The shifting of a database from one environment to another requires connectivity between the servers. The process by which Db2 Shift moves data requires a connection to either an OpenShift/Kubernetes/CP4D cluster, a server-less ssh connection, or a local connection.

When shifting data to OpenShift, Kubernetes, or CP4D, the Db2 Shift program will require the following:

- A suitable client for the Cluster software
 - OpenShift CLI (oc) 4.0+ for OpenShift and Cloud Pak for Data shifts
 - Kubernetes CLI (kubectl) for all Kubernetes clusters
 - SSH serverless connection to the target system
 - Local connection does not require a client but is only available for deployment of cloned databases
- OpenShift Version 3.11, 4.x
- Cloud Pak for Data 3.5, 4.x
- Kubernetes Version 1.19+
- A connection between the Source and Target servers
- Sufficient disk space in /tmp or other directory (for Clone operations only)

Note 1: Only the OpenShift 4.0+ client is supported for shift operations. The target cluster can be OpenShift 3.11 or OpenShift 4.0+.

Note 2: Some Linux distributions do not support the OpenShift CLI. Customers wanting to shift from an environment that does not support the OpenShift CLI will need to use the Clone option and then copy the clone directory to a client that does support the OC client.

For an OpenShift or Cloud Pak for Data environment, download a copy of the OC version 4 command line interface from:

- Red Hat Customer Portal <https://access.redhat.com>
- OpenShift GitHub <https://github.com/openshift/oc>
- OpenShift Community Distribution https://docs.okd.io/latest/cli_reference/openshift_cli/getting-started-cli.html

For Kubernetes distributions, you can download the code from:

- Kubernetes Tools <https://kubernetes.io/docs/tasks/tools>

You can also issue the following command from a Linux terminal window to download the code:

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

Once the code has been downloaded, place the code into the local bin directory and make sure that it has been marked as executable.

```
chmod +x kubectl
mkdir -p ~/.local/bin
mv ./kubectl ~/.local/bin/kubectl
```

Userid and Authentication Requirements

The Db2 Shift operation must take place under the userid of the INSTANCE owner of the database being shifted. Traditionally this userid has been db2inst1 but may be different in your environment. The user must be logged in as the INSTANCE owner and must have access to the db2shift command.

The instance owner must also have ssh server-less connectivity to the TARGET system if a Db2-to-Db2 instance shift is being performed. The ssh connection is not required when shifting into a containerized (OpenShift, Kubernetes, CP4D) environment.

To access the TARGET pod in a cluster, the user must have authenticated to OpenShift or Kubernetes and have access to the POD that Db2 is running on. For OpenShift environments, authentication is done either through a userid and password or a token. An example of connecting with a userid/password:

```
oc login -u oadmin -p oadmin
```

Shifting to Cloud Pak for Data does not require a CP4D userid. The OpenShift userid or token for the underlying cluster is required to authenticate for the Db2 Shift operation.

Kubernetes installations will require the use of a profile (Config) to connect to the cluster.

For both OpenShift and Kubernetes shift operations, the namespace (Kubernetes) or project (OpenShift) name will be required. In the case of CP4D the namespace is usually cp4d, but it depends on how the cluster was deployed. In OpenShift and Kubernetes, the default namespace/project must be set **prior** to running the Db2 Shift command. You can override the namespace and project in the command but setting the namespace prior to executing the script will minimize errors with Db2u pods not being found.

Target Database Requirements

The target database should be created before running the Db2 Shift command. The Db2 Shift command cannot create the database for you in containerized environments (OpenShift, Kubernetes, CP4D). You must use the CP4D console to create a new database, or the Db2u Operators on OpenShift or Kubernetes to create and deploy the Db2 database. For traditional Db2 installations, the database would be created using the CREATE DATABASE command.

The new database name should be the same as the database being shifted. This is a strict requirement for CP4D databases, but not necessary for the other environments. Db2 Shift will take the existing database and clone it into the target database name.

For shifting or cloning a database to a traditional Db2 Instance, Db2 Shift will create the database on your behalf if you specify the create database flag. Otherwise, the program will stop execution because the target database will not be found.

Db2 databases created in clustered environments are always created at the latest level (11.5.7+). For traditional Db2 instances created using the db2install command, a minimum level of 11.5 should be used since prior versions are now considered out of service.

In addition, the nodes of an MPP configuration must match that of the SOURCE database and the memory, disk, and cores **should** match the SOURCE database requirements.

Source Database Requirements

The source database can be moved if the following conditions are met:

- The database resides on a Linux server (X64 or Power Linux LE)
- The database was created with Automatic storage
- The database is an OLTP, SMP, or MPP system
- Row or Column mode storage, including encrypted databases
- Mirror, Archive, and Overflow Logs use Disk only
- User Defined Functions/Procedures located in the SQL lib Directory
- Db2 Version 10.5, 11.1, or 11.5 servers can be moved and upgraded at the same time

The following features are not currently supported:

- pureScale Feature (Not available yet in the Db2u container)
- Text Extender

There are a few configuration settings which cannot be shifted:

- Only databases created with automatic storage are supported

- The system contains external procedures which are not in the standard Db2 library - these will need to be manually recreated and catalogued
- The LOGARCHMETH1/2 setting only supports DISK as a target in Db2u
- The database encryption keys will be moved to the new location, but if the target already has encrypted databases, then you will need to manually migrate the encryption key to the target location
- The 11.5.7 `ctrl_recov_file_path` database setting must be set to NULL before shifting a database. If the setting is anything other than NULL, the target database will not be able to start until an identical path is available at the target.

Instance Owner and DBADM and SECADM Userid

Special care must be taken when moving a database into Cloud Pak for Data or into any Db2U deployment. The Db2U POD always uses `db2inst1` as the instance owner. If your instance owner is different (i.e., `db2inst2`), the `db2inst1` user at the target will not have any privileges on this database. You will not be able to manage the database.

In addition, for Cloud Pak for Data environments, your database name on CP4D must match the source database name. If the database names are mismatched, the Data Management Console and CP4D services will be unable to detect and manage the database.

If you are shifting a database to Cloud Pak for Data or a Db2u POD, and your source INSTANCE userid is **not** `db2inst1`, you must execute the following SQL commands from a userid that has SECADM authority:

```
GRANT SECADM ON DATABASE TO USER db2inst1
GRANT DBADM ON DATABASE TO USER db2inst1
```

The `db2inst1` userid does not need to exist in the Operating system to grant these privileges to the userid. If you are moving the database to a traditional Db2 instance, you could always create a new userid at the OS level to manage the database, but this option is not available in a container.

3

Installation

GETTING STARTED WITH DB2 SHIFT

Chapter 3: Installation

The Db2 Shift program (db2shift) is a single executable image that can be run directly on Linux. This is a bundled application which means that it contains files and settings that are part of the executable code. From a user perspective, no additional software is required to run it.

The program does not create any directories on your system, but it will generate files during the execution of a shift. A best practice would be to create a new directory only for the purposes of the running the Db2 Shift code.

The Db2 Shift program is bundled into a single tar file which contains version of the code for different Linux distributions (there may be more than those listed). Unpacking the tar file will create a directory structure that contains the following:

- License information
- Documentation
- A directory for each distribution of Linux that is supported
 - RHEL-6 – CentOS 6, Red Hat 6
 - RHEL-7 – CentOS 7, Red Hat 7
 - RHEL-8 – CentOS 8, Red Hat 8, CentOS Stream
 - Ubuntu-18 – Ubuntu 18.04
 - Ubuntu-20 – Ubuntu 20.04
 - Ubuntu-22 – Ubuntu 22.04
 - SUSE-15 – openSUSE 15, SUSE 15
 - PowerLE – Power Linux LE

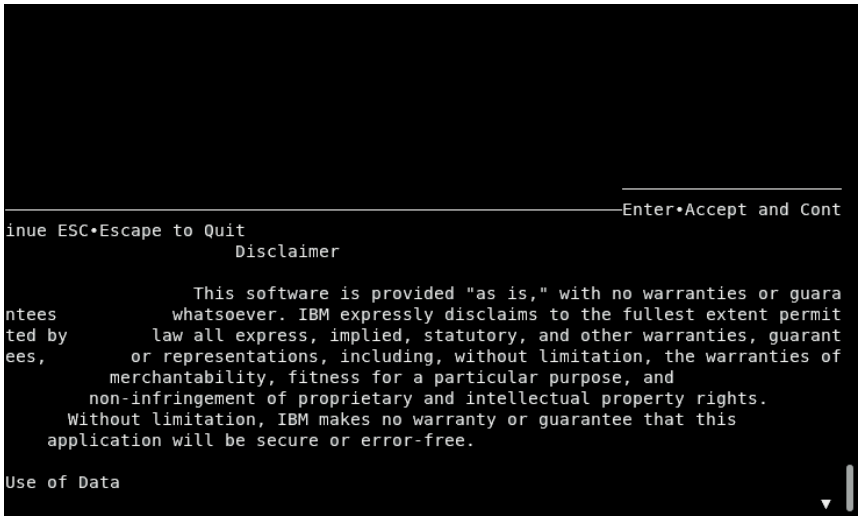
Each directory will contain the db2shift program suitable for that Linux distribution (i.e., db2shift-centos8).

Additional versions may be added depending on feedback from users. Place the program into its own directory and make sure that it has the execution bit set. You may also want to change the program name to db2shift.

```
mv db2shift-centos8 db2shift
chmod +x ./db2shift
```

Additional versions may be added depending on feedback from users.

The program will work in command mode for all environments. If you attempt to use the UI mode with the incorrect OS version, the screen may look distorted.



```

Enter•Accept and Cont
-----
inue ESC•Escape to Quit
          Disclaimer

          This software is provided "as is," with no warranties or guara
ntees      whatsoever. IBM expressly disclaims to the fullest extent permit
ted by    law all express, implied, statutory, and other warranties, guarant
ees,      or representations, including, without limitation, the warranties of
          merchantability, fitness for a particular purpose, and
          non-infringement of proprietary and intellectual property rights.
          Without limitation, IBM makes no warranty or guarantee that this
          application will be secure or error-free.

Use of Data
```

If your display looks like this, then you will have the wrong version installed. If you are running on a different platform than those listed above, please contact support. See Appendix B for details.

Program Download

The program is available from Passport Advantage or through the Technical Preview site:

- ibm.biz/c2cdownload
- <https://www.ibm.com/software/passportadvantage>

You will need an IBM userid to register and gain access to the program.

4

Db2 Shift Syntax

COMMAND LINE AND UI OVERVIEW

Chapter 4: Db2 Shift Syntax

The Db2 Shift command has two modes of operation:

- Command Line mode - A set of parameters are passed to the program and executed
- Interactive mode - An interactive User Interface is used to supply the parameters required to execute a command

The interactive mode provides the ability to generate all the commands and includes the following additional features:

- Ability to save scripts for future use
- Compare source and destination target databases to check for compatibility
- Review Log files for any shift errors
- Extensive help information for input fields and Db2 Shift scenarios

The interactive mode is invoked when the Db2 Shift command is executed with no parameters, or if either the following options are included on the command line:

- `--help` – Request detailed help information on running the Db2 Shift command
- `--mono` – Use a monochrome display format (White on Black) and non-Unicode character set when using the UI
- `--accept` – Accept license agreement (no license prompt)
- `--version` – Display Version information
- `--shiftpod`, `--shiftdb2`, `--clone`, `--deploypod`, `--deploydb2`, `--hadzpod`, `--hadzdb2` – Direct link to the Db2 Shift scenario found on the main menu
- `--logs` – View logs from last shift execution

Details of how to navigate the UI system are found in the User Interface Overview section.

A summary of the Db2 Shift command is found on the next page.

Syntax Summary

db2shift

--mode=[all,move,clone,apply_clone,sec_and_monitor,hadr_setup,
push_clone,pull_clone]

UI Mode Only

--help, --mono, --accept, --version

Client

--ssh,--local,--oc,--kubect1

Source Details

--source-database="sample"

--source-owner="db2inst1"

Destination Details

--dest-database="bludb"

--dest-owner="db2inst1"

--dest-server="c-demo-db2u-0" or "userid@ip.address"

--dest-namespace="db2u", --dest-project="db2u"

--overrides="instance_memory 3468896"

--dest-create-db

--mirror-path

Clone and Copy Details

--clone-dir="/tmp/clone"

--local-dir="/tmp/clone"

--dest-dir="/tmp/clone"

HADR Options

--hadr or -keep-rfw-pending

--source-hadr-server="some.server.com"

--dest-hadr-server="other.server.com"

--source-hadr-port=3700

Meta Data Generation

--blank-slate (or --new-settings)=[True|False]

--gen-settings

--verify-only

Synchronization Mode

--sync=[start_sync, rerun_sync, finish_sync]

Database Status

--online, --offline

Performance Options

--threads=[1..8]

--compression=[0..9]

Function Library Support

--exclude-functions

Details for each one of these options can be found in the reference chapter and are also discussed in each scenario section.

- Mode Option
- Target Client Instance
- Target Client POD
- Source Database
- Source Database Owner
- Destination Database
- Destination Owner
- Destination Server POD
- Destination Server Instance
- Destination Pod Namespace or Project
- Create Destination Database
- Mirror Paths
- Clone Directory
- Local Directory
- Target Directory
- HADR Setup
- HADR Source or Destination Server
- HADR Ports
- Metadata Generation
- Synchronization Options
- Online/Offline Mode
- Threading
- Compression
- Stored Procedures
- Overrides

Db2 Shift User Interface

Db2 Shift can be run either as a command line utility, or as an application with a menu system. The Db2 Shift menu system provides an easy-to-use interface for generating the appropriate shift commands and includes extensive help on the various parameters that need to be supplied.

This section will discuss the user-interface in detail and describe the additional features that are only found in the UI including:

- The ability to check the source and destinations to determine whether they are compatible
- View errors logs for diagnosing problems
- Ability to save commands as separate executable shell scripts

Enabling UI Mode

The Db2 Shift UI Mode is automatically invoked when the `db2shift` command is issued without any parameters. Requesting help with the `--help` flag will also display help panels using the UI system.

If your terminal session does not support color, the UI can be forced to the display the text as WHITE characters on a BLACK background. Use the `--mono` flag when using the `db2shift` command.

For terminals that do not show any color, please check the settings in the `TERM` environment variable.

```
echo $TERM
```

If the results do not show `xterm-256color` then you will need to check what terminal emulations are available with the following command:

```
toe -a
```

This will produce a list of terminal emulations. The preferred setting is `xterm-256color`, but other color options may also work.

Update the terminal emulation setting by issuing the command:

```
export TERM=xterm-256color
```

To make this change permanent on your system, edit the `~/.bashrc` configuration file in your instance home directory and add the above command to the end of the script. Save the file and then issue the following command to change the terminal settings.

```
source ~/.bashrc
```

If you find the display contains strange characters (white boxes instead of underscores for fields) like those on the following page, then your OS may not support many of the Unicode characters.

Chapter 4: Db2 Shift Syntax

```
Db2 Shift [/home/db2shift]
Shift to Db2U on OpenShift or Kubernetes
Source Details
Database Name : flights#
Instance Owner : db2inst1#.....

Destination Details
Target Location : # OpenShift # Kubernetes
Database Name : #.....
POD Project : #.....
POD Name : #.....

Metadata
: # Refresh # None # Settings Only # Verify
Sync Options : # None # Initialize # Refresh # Finalize
Move Options : # Database # Database/LDAP/DMC # Move Database for HADR
Exclude Routines : # Do not move external routines
Database Mode : # Online Move # Offline Move
Thread count : 4
Compression : 0
Overrides : #.....

Enter the name of the Destination database.
ESC-Quit #?Field Help #A-Analyze #X-Review and Execute
```

This is an indication that Unicode character set extensions have not been installed on the Linux distribution being used. The `--mono` flag will change the characters so that they are more readable in this environment. When using the UI, the following characters will be substituted for the graphical characters:

- Underscore (fields) with dots “...”
- Checkbox selection with asterisk “*” and unselected with dash “-“
- More (scrollbar) with “+” and previous with “-“

The following screen illustrates the difference in mono mode.

```
Db2 Shift [/home/c2cdb2/c2c]
Shift to Db2U on OpenShift or Kubernetes
Source Details
Database Name : flights
Instance Owner : george#.....

Destination Details
Target Location : - OpenShift * Kubernetes
Database Name : db2oltp
POD Project : db2u#.....+
POD Name : this-this-is-a-test.tosee#.....+

Metadata
: - Refresh * None - Settings Only - Verify
Sync Options : * None - Initialize - Refresh - Finalize
Move Options : - Database * Database/LDAP/DMC - Move Database for HADR
Exclude Routines : * Do not move external routines
Database Mode : - Online Move * Offline Move
Thread count : 1
Compression : 4
Overrides : #.....+

ESC-Quit ^F-Field Help ^A-Analyze ^X-Review and Execute
```


Menu System Overview

The Db2 Shift menu system provides access to all scenarios that are described in this documentation:

- Shift a Db2 database to OpenShift, Kubernetes or CP4D
- Shift a Db2 database to another Db2 instance
- Create a Cloned copy of the Db2 database for later deployment
- Deploy a clone into an OpenShift, Kubernetes or CP4D container
- Deploy a clone into another Db2 instance
- Initialize HADR between Source and Target POD
- Initialize HADR between Source and Target Instance
- Initialize DMC and LDAP Authentication for CP4D
- Copy Cloned databases to a POD

The various screens are described in the sections below. Some features that are specific to the UI are:

- Legal Disclaimer
- Help Information
- Analyze Database Configuration
- Review shift log files

Interacting with the UI

The Db2 Shift UI is based on a character-based terminal display, like VT100 or 3270 technologies. Using a character-based display format eliminates the need for a graphic display environment (GDM) and significantly reduces the memory requirements for the program.

When using a terminal-based UI, only keyboard entries can be used to navigate the screen (instead of mouse movements). There are some panels which will display URLs pointing to reference material. If your environment provides access to a mouse, you will be able to click on these links. However, mouse clicks within the UI have no effect.

The keys that are active within a screen are usually displayed at the bottom of the screen.



```
ESC•Quit ^F•Field Help ^D•Select Directory ^A•Analyze ^X•Review and Execute
```

The last three lines of all Db2 Shift panels will include a message line, a separator line, and a list of all active keys. In this example, there are five keys:

- ESC - Quit
- ^F - Field Help
- ^D - Select Directory
- ^A - Analyze

- ^X - Review and Execute

The ESCAPE key is used to exit out of the current panel and return to the previous panel. To completely exit out of the program at **any** time, use the CTRL+C combination. This key (break) will always stop execution of the UI and return you to the terminal prompt.

If a letter is provided as an "action" key, press that letter on the keyboard. There is no need to use the shift key. If the letter is preceded with a carat symbol ^, this represents the control (CTRL) key on the keyboard. In the above example, the ^F key requires that you hit the control key and the F key at the same time.

Some screens may use letters for commands – for example, the bottom of the main screen uses letters to switch screens:



Panels that require user input will need to use all the characters on the keyboard, so it is not possible to use single keys to invoke an action or move to another panel. To get around this restriction, the control key (CTRL) is prefixed to the character (^A) to tell the UI it is an action key rather than a regular input key.

The following keys are used for navigation on a panel:

- ESC – Return to previous screen or close message window
- DEL – Delete the current character and move text left
- Shift+DEL – Delete field from cursor to end of field
- BKSP – Delete the character left of the current character
- Up/Down – Move up or down to the next field on the screen
- Left/Right – Move left/right in the field and jump to the next field. Note that long fields will scroll automatically
- TAB – Tab to the next input field
- Shift+TAB – Tab backwards to the previous input field
- ENTER – Select a row, execute a command, or move down to next field
- Home – Cursor to the beginning of a field
- End – Cursor to the end of a field
- PGUP/PGDN – Scroll list up and down when your cursor is in the list. Note that scroll bar on the side indicates if more rows are available.
- (fn+)Shift+Up/(fn+)Shift+Down – Same as page up and page down if those keys are not available
- SPACEBAR – Select or deselect a list item or turn a value on or off

On a Mac (OSX), the function key needs to be used with the Shift+Up/Down arrow to simulate a Page Up or Page Down key.

Text Input

A text input field will contain underscores that represent the total length of the field.

```
POD Project      : db2u_.....
```

Any text entered in these fields will shift characters to the left (insert mode). Once text hits the end of the field it will no longer allow additional characters to be entered until you delete existing characters.

If you need to remove characters from the current character position to the end of the input field, use **Shift+DEL** to delete them.

Fields that only have one character will automatically replace the contents of the field rather than requiring the contents to be deleted.

Numeric Input

A numeric field will prevent the input of any characters other than valid numbers [0-9].

```
Source HADR Port : 3700_
```

Long Fields

Several fields require more space than what is available on the screen. These fields are either file or directory names, or server addresses.

```
Source HADR IP   : some.server.com.....+
```

These long fields will allow characters to roll off the end of the field and disappear. You can get to the end of the character string by using the **END** key or back to the beginning with the **HOME** key. You can usually tell that the field allows for larger text sizes because there is a tiny plus sign (+) at the end of the input line and the text will scroll past the end of the input field as you type.

Options

Option fields provide a way of selecting one option from a list of options.

```
Target Location : ● OpenShift ○ Kubernetes ○ Local
```

Use the **TAB** or right arrow to move between the values in the list. Only one option can be selected at a time! To select an option, make sure the option circle is highlighted and then press the **SPACEBAR** to select the option. The previous option will be de-selected, and the new option selected.

Check Boxes

A Checkbox is either on or off.

```
Exclude Routines :  Do not move external routines
```

Use the SPACEBAR to turn an option on or off.

Lists

There are three types of lists that are found in the UI:

- Read-only list
- Select One from the list (like Options)
- Select Multiple

Read-only lists are used for documentation and when selecting an item for immediate use (for instance, when selecting a directory).

```
Db2 Shift [/home/c2cdb2/c2c]
Select a directory or file from the list below.
▼ bin
  /boot
  /dev
  /etc
  /home
  /lib
  /lib64
  /media
  /mnt
  /opt
  /proc
  /root
  /run
  /sbin
  /srv
  /sys
  /tmp
ESC•Quit ^F•Help ^R•Root ^O•Home ^S•Select Directory/File Enter•Traverse
```

You can move up and down a list using the arrow keys. If the list requires a selection, you must position the cursor on the item you want selected, and then press the ENTER key.

Selection lists are like Option list where several items are shown, and one must be selected from the list.

```
Db2 Shift [/home/c2cdb2/c2c]

Review the settings below and select which ones should be overridden. Use ^?
to view details of the parameter and links to the documentation.
```

Parameter	Source	Destination
<input checked="" type="checkbox"/> EXT_PROC_COUNT	9	External procedures
<input type="checkbox"/> TBSP_DEVICE	/mnt/db2	Unsupported tablespaces
<input type="checkbox"/> TBSP_CONTAINER	/mnt/db2	Unsupported tablespaces
<input type="checkbox"/> TBSP_DIRECTORY	/mnt/db2	Unsupported tablespaces
<input type="checkbox"/> logarchmeth1	TSM	Unsupported value
<input type="checkbox"/> logarchmeth2	VENFOT	Unsupported value
<input type="checkbox"/> cur_eff_arch_lvl	V:10 R:5 M:5 F:0 I:0 SB	Automated migration
<input type="checkbox"/> cur_eff_code_lvl	V:10 R:5 M:5 F:0 I:0 SB	Automated migration
<input type="checkbox"/> encrypted_database	YES	Encrypted database
<input type="radio"/> repl_site_id	0	0
<input type="radio"/> authentication	SERVER	SERVER_ENCRYPT
<input type="radio"/> instance_memory	3468896	598630

```
^Q*Quit ^F*Field Help ^X*Generate Overrides
```

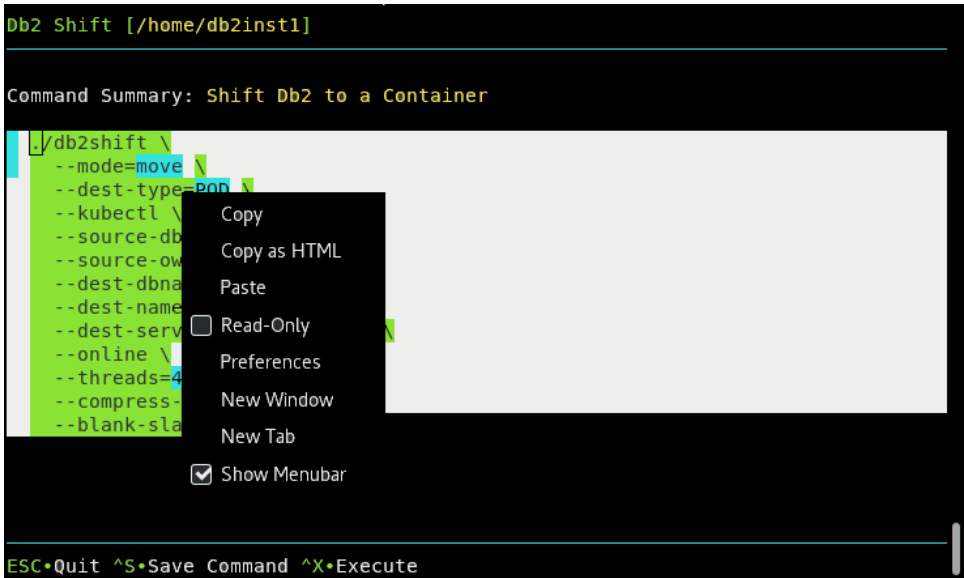
The example above includes two options that can be selected (circles) while the other items only have an underscore. Only those items with circles can be selected. Use the SPACEBAR to select one or more options in the list. A normal list only allows one item to be selected. Selecting an item will deselect any existing item and then select the item you were currently on.

Finally, lists may be larger than what is shown on the screen. The scrollbar indicators will be placed at the top of the list to indicate whether there are values above or below the current list items that are displayed.

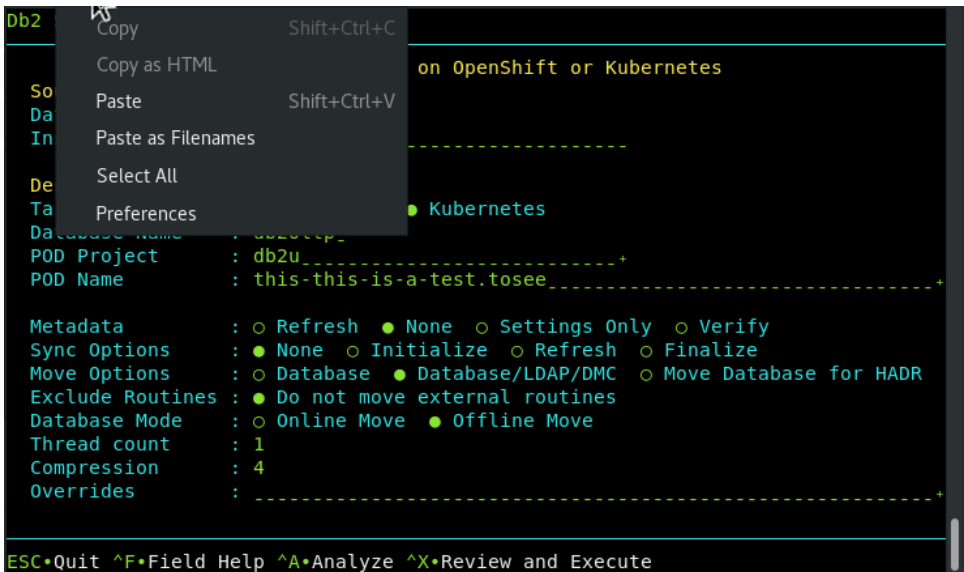
```
▲ The data used within this program is used only for the purposes of
▼ shifting databases between different environments. None of the control
data used within the application will be kept or transmitted to
```

Cut and Paste

If your system allows the use of a mouse, you can copy text from the Db2 Shift UI by selecting the text, using the right mouse button, and selecting Copy from the menu.



You are also able to paste items into a field positioning the cursor on the field and then selecting Paste from the terminal menu bar.



The other option for pasting a value is to use the key combination CTRL+SHIFT+V. This will paste the contents of the clipboard into the current field your cursor is positioned in.

Disclaimer

The Db2 Shift disclaimer is always displayed when starting the Db2 Shift command.

```
Db2 Shift [/home/c2cdb2/c2c]

Db2 Shift Disclaimer

Please review the disclaimers below and press ENTER to accept
or ESCAPE to cancel.

▼ Disclaimer

This software is provided "as is," with no warranties or guarantees
whatsoever. IBM expressly disclaims to the fullest extent permitted by
law all express, implied, statutory, and other warranties, guarantees,
or representations, including, without limitation, the warranties of
merchantability, fitness for a particular purpose, and
non-infringement of proprietary and intellectual property rights.
Without limitation, IBM makes no warranty or guarantee that this
application will be secure or error-free.

Use of Data

Enter•Accept and Continue ESC•Escape to Quit
```

By pressing the ENTER key, you are accepting the terms and conditions of the program. If you press ESCAPE, the program will immediately terminate. Details of the terms and condition are found in the Disclaimer section of this documentation.

If you prefer to skip the Disclaimer screen, use the --accept option on the Db2 Shift command line:

```
db2shift --accept
```

Help Panels

Help screens are created using read-only lists. The help information will be displayed with scroll bar indicators on the far lefthand side of the text:

```
Db2 Shift [/home/c2cdb2/c2c]
▲ ▣ • Create a clone of a database for later deployment
▼ ▣ • Extract the secrets from a Db2U pod on OpenShift
  ▣ • Check the local database for compatibility with a containerized
    environment

The sections below describe the various options that are available when
using the Db2 Shift program.

Db2 Containerization to OpenShift, Kubernetes, or CP4D

This menu is used to take an existing Db2 database on an on-premise
system, and containerize it to OpenShift, Kubernetes, or CP4D. The panel
requires the following information:

  ▣ Source Database details
  ▣ Destination location

ESC•Close

ESC•Quit H•Help K•Keyboard Help L•Legal S•Syntax Help
```

The up and down arrow indicate that there is text prior to this page and that there is text after this page. Use the Page Up and Page Down buttons (or (fn+)Shift-Up arrow and (fn+)Shift-Down arrow) to scroll through the help.

On most panels, pressing ^F (CTRL+F) on a field will display detailed information about that field. For instance, pressing ^F on the follow field:

```
Database Name : lights_
```

This will result in detailed information about the field being displayed on top of the existing panel.

```
Db2 Shift [/home/c2cdb2/c2c/dist]

Source Database
Database Name: -----
Syntax       : --source-database=""

The source database is the name of the database that you want to move to
the new location. Note that you can have the same or different database
name at the target. If you provide a different database name at the
target, the program will copy the database from the source and place it
on the target and use the existing name.

ESC•Quit

ESC•Quit ^F•Field Help ^A•Analyze ^X•Review and Execute
```


The help information can be dismissed by using the ESCAPE key. Note that arrows on the left side of the text will indicate if you need to use the scroll keys to view additional information.

Main Screen

The main Db2 Shift screen provides access to all the scenarios that have been discussed in the documentation.

```
Db2 Shift [/home/db2inst1]
-----
Place the cursor on the option you want and press Enter.
-----
Shift  Shift a Db2 database to OpenShift, Kubernetes or CP4D
      Shift a Db2 database to another Db2 instance
-----
Clone  Create a Cloned copy of the Db2 database for later deployment
      Deploy a clone into an OpenShift, Kubernetes or CP4D container
      Deploy a clone into another Db2 instance
      Copy a clone to or from a Pod
-----
HADR   Initialize HADR between Source and Target POD
      Initialize HADR between Source and Target Instance
      Initialize DMC and LDAP Authentication for CP4D
-----
Other  Analyze Local Database
      View Shift Log
-----
Press Enter to Select the Option
-----
ESC•Quit H•Help K•Keyboard Help L•Legal S•Syntax Help
```

Before describing the various options on the screen, there are some extra screens that can be accessed from the main screen:

- H – Db2 Shift Help
- K – Keyboard Help
- S – Syntax Details
- L – Legal Disclaimer

The Db2 Shift Help provides a general overview of the Db2 Shift program and details on every Db2 Shift scenario. The keyboard help provides a quick guide on how the keys work and the Syntax details provides information on every setting that Db2 Shift uses.

The Legal disclaimer provides the entire text of the legal documentation shipped with the Db2 Shift program. Odds are it is larger than all the code that was written for Db2 Shift!

To exit the Db2 Shift program, press ESCAPE from this screen. To select one of the options, use the up and down arrow keys to place the cursor on the option you want and then press the ENTER key. This will display the menu for that option.

The options themselves are divided into four sections:

- Shift – Shift a database from a local instance to a destination
- Clone – Clone a database and the deploy or copy to a destination
- HADR – Set up and start HADR between a source and target destination
- View – Options to analyze a local data or view current log file


Details of each one of these panels is described in the chapters that follow. Note that not every command line option will be display in the UI panel. For instance, the mode option to move a database to a pod is automatically generated by the UI so there is no requirement for this setting to be supplied.

In addition to using the menu system to navigate to the shift scenario you want to perform, you have the option of using the following keywords to go directly to a specific shift menu:

- --topod – Shift a Db2 database to a Kubernetes, OpenShift or CP4D pod
- --todb2 – Shift a Db2 database to another Db2 instance
- --clone – Clone a Db2 database
- --deploypod – Deploy a clone to a Kubernetes, OpenShift, or CP4D pod
- --deploydb2 – Deploy a clone to another Db2 instance
- --hadrapod – Set up connection between source and destination pod
- --hadrdb2 – Set up connection between source and Db2 instance
- --logs – View logs from last execution

Executing Commands

Each one of the UI panels has an option to Review and Execute a command. The key that is assigned to execute command is ^X.



```
ESC•Quit ^?•Field Help ^D•Select Directory ^A•Analyze ^X•Review and Execute
```

When the execute button is pressed, the Db2 Shift program will first determine if all the parameters have been properly filled in. In the event of a problem, an error message will be display and the cursor placed on the field that needs to be updated.

```
Db2 Shift [/home/c2cdb2/c2c/dist]
Shift to Db2U on OpenShift or Kubernetes

Source Details
Database Name      : █-----
Instance Owner    : db2inst1-----

Destination Details
Target Location   : ● OpenShift ○ Kubernetes
Database Name     : -----
POD Project       : -----+
POD Name          : -----+

Metadata          : ● Refresh ○ None ○ Settings Only ○ Verify
Sync Options      : ● None ○ Initialize ○ Refresh ○ Finalize
Move Options      : ● Database ○ Database/LDAP/DMC ○ Move Database for HADR
Exclude Routines : ○ Do not move external routines
Database Mode     : ● Online Move ○ Offline Move
Thread count      : 4
Compression       : 0
Overrides         : -----+
Error: Missing source database value

ESC•Quit ^F•Field Help ^A•Analyze ^X•Review and Execute
```

When the panel passes all checks, the Db2 Shift command is displayed on another screen.

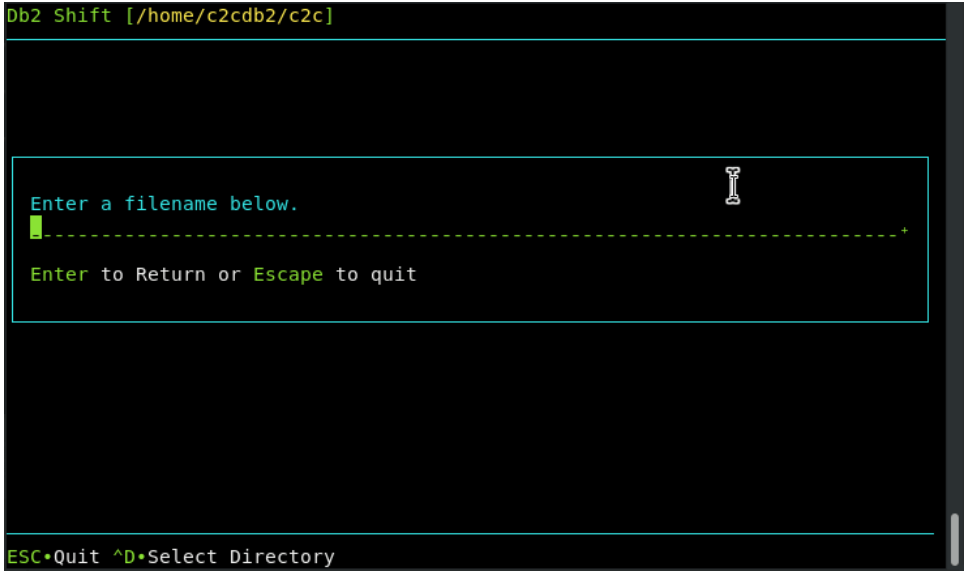
Summary Screen

```
Db2 Shift [/home/c2cdb2/c2c]
Command Summary: Shift Db2 to a Container

█/db2shift \
--source-dbname=flights \
--source-owner=db2inst1 \
--dest-dbname=db2oltp \
--dest-namespace=db2u \
--dest-server=c-demo-db2u-0 \
--dest-type=POD \
--oc \
--mode=move \
--hadr \
--online \
--threads=4 \
--compress-level=4 \
--exclude-functions \
--blank-slate=true

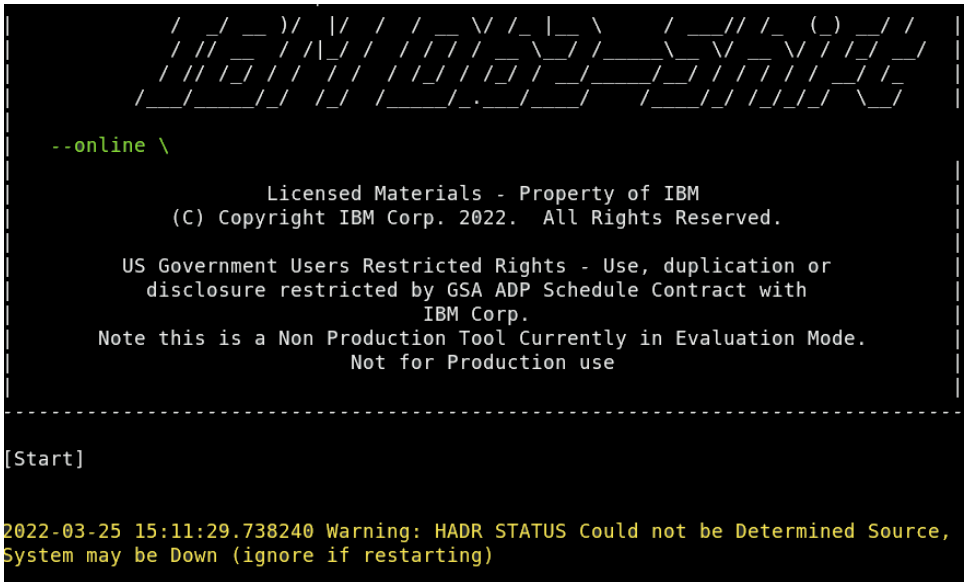
ESC•Quit ^S•Save Command ^X•Execute
```

If you have mouse control on your system, you will be able to select the text on this screen and copy it into a script or a terminal session. Alternatively, you can use the ^S Save Command to save the command to a file for future execution.



If you press the Execute ^X key on the review page, Db2 Shift will execute your command immediately. The output from the Db2 Shift process will be displayed on the screen (the UI will disappear).

Db2 Shift Execution



Various messages will be displayed during the execution of the Db2 Shift command with progress bars indicating the current step in the process. When the execution completes, the UI will be displayed with a success indicator and the contents of the log file.

```
[Start]
-----
                        Environment Diagnostics Check List
                        2022-04-13 12:53:31.407992
-----
Client Version:           Succeeded
Cluster Detection:       Succeeded
POD Detection:           Succeeded
RSYNC Detection:        Succeeded
Source User Check:      Succeeded

2022-04-13 12:53:31.707670 Info: Info: Environment Diagnostics Passed
-----
                        Performing Pre-Check and Initalization Procedures
                        2022-04-13 12:53:31.792969
-----
✓ Destination Exists
● Checking HADR status
```

Here is an example of a run that fails due to a connection to the Kubernetes cluster is unavailable.

```
Db2 Shift [/home/db2inst1]

Run Status: Failed
Error: Environment does not meet Requirements Please check log for more deta

▼ Diagnostic: Passed Arguments
--mode=move --dest-type=POD --kubectl --source-dbname=flights --source-
Diagnostic: [source      ] mv source.settings old_source.settings
Diagnostic: [source      ] mv destination.settings old_destination.settings
Diagnostic: [source      ] mv update.settings old_update.settings
Diagnostic: [source      ] cat ./status.cfg | grep -c normal
Diagnostic: [source      ] cat ./status.cfg | grep -c sync
Diagnostic: [source      ] kubectl version
Diagnostic: Client Version: version.Info

Environment Diagnostics Check List

Client Version:           Succeeded
Cluster Detection:       FAILED
Info: Error: No Active Cluster Connection Exists

ESC•Quit ^W•Wrap Text On/Off
```

The log file is displayed below the run status. You can scroll through this list to determine what steps failed during the Db2 Shift execution.

A successful run will also display the log file.

```
Db2 Shift [/home/db2inst1]

Run Status: Success
Transfer Completed Db2 Started in Mode SNAPSHOT

▼ Diagnostic: Passed Arguments
--mode=move --dest-type=POD --kubectl --source-dbname=flights --source-
Diagnostic: [source      ] mv source.settings old_source.settings
Diagnostic: [source      ] mv destination.settings old_destination.settings
Diagnostic: [source      ] mv update.settings old_update.settings
Diagnostic: [source      ] cat ./status.cfg | grep -c normal
Diagnostic: [source      ] cat ./status.cfg | grep -c sync
Diagnostic: [source      ] kubectl version
Diagnostic: Client Version: version.Info
Server Version: version.Info
Diagnostic: [source      ] kubectl get pods | grep c-demo-db2u-0
Diagnostic: kubectl exec c-demo-db2u-0 -q -- rsync --version
Diagnostic: [destination] /bin/bash -c kubectl exec c-demo-db2u-0 -q --
Diagnostic: [source      ] whoami
Diagnostic: db2inst1

ESC•Quit ^W•Wrap Text On/Off
```

The log records are truncated so that you only see 80 characters of the log file. If you want to see more details, the ^W (Wrap Text) will reformat the output so you will be able to see all the contents of the line.

```
Db2 Shift [/home/db2inst1]

Run Status: Success
Transfer Completed Db2 Started in Mode SNAPSHOT

▼ Diagnostic: Passed Arguments
--mode=move --dest-type=POD --kubectl --source-dbname=flights --source-
-- owner=db2inst1 --dest-dbname=db2oltp --dest-namespace=db2u --dest-
-- server=c-demo-db2u-0 --online --threads=4 --compress-level=4 --blank-
-- slate=true
Diagnostic: [source      ] mv source.settings old_source.settings
Diagnostic: [source      ] mv destination.settings old_destination.settings
Diagnostic: [source      ] mv update.settings old_update.settings
Diagnostic: [source      ] cat ./status.cfg | grep -c normal
Diagnostic: [source      ] cat ./status.cfg | grep -c sync
Diagnostic: [source      ] kubectl version
Diagnostic: Client Version: version.Info
-- GitVersion:"v1.23.0",
-- GitCommit:"ab69524f795c42094a6630298ff53f3c3ebab7f4",
-- GitTreeState:"clean", BuildDate:"2021-12-07T18:16:20Z",

ESC•Quit ^W•Wrap Text On/Off
```

Database Analysis

Shifting a database from an instance to a POD requires the Db2 Shift program to check that the source database meets certain criteria including:

- The source database must be 10.5, 11.1, or 11.5

- Only databases created with automatic storage are supported
- The system contains external procedures which are not in the standard Db2 library - these will need to be manually recreated and catalogued
- The LOGARCHMETH1/2 setting only supports DISK as a target in Db2u
- The database encryption keys will be moved to the new location, but if the target already has encrypted databases, then you will need to manually migrate the encryption key to the target location

This checking is done when the Db2 Shift program begins execution. Even though the database may meet these requirements, the source database environment may have certain settings which need to be present at the target location. By default, **all** database settings are moved to the target location. However, none of the instance settings are moved during the shift step unless you explicitly name them.

To test the compatibility of your source database to the target location, the Analyze ^A key can be used:

A screenshot of a terminal window showing a menu of options for the Db2 Shift program. The menu items are: ESC•Quit, ^?•Field Help, ^D•Select Directory, ^A•Analyze, and ^X•Review and Execute. The 'Analyze' option is highlighted with a green background.

The Analyze option is found in four of the Db2 Shift Scenarios:

- Shift to Pod
- Shift to Instance
- Deploy to Pod
- Deploy to Instance

Note that you cannot run an Analysis step against a database that does not currently exist on the target. This applies only to shift or clone operations that are being performed against a standard Db2 instance and you are requesting that the target database be created by the Db2 Shift command.

When the Analyze function is selected, the Db2 Shift program will gather information from the source and target databases and present a report containing the settings that are different between the environments.

```
Db2 Shift [/home/c2cdb2/c2c]

Review the settings below and select which ones should be overridden. Use ^?
to view details of the parameter and links to the documentation.
```

Parameter	Source	Destination
█ EXT_PROC_COUNT	9	External procedures
_ TBSP_DEVICE	/mnt/db2	Unsupported tablespaces
_ TBSP_CONTAINER	/mnt/db2	Unsupported tablespaces
_ TBSP_DIRECTORY	/mnt/db2	Unsupported tablespaces
_ logarchmeth1	TSM	Unsupported value
_ logarchmeth2	VENFOT	Unsupported value
_ cur_eff_arch_lvl	V:10 R:5 M:5 F:0 I:0 SB	Automated migration
_ cur_eff_code_lvl	V:10 R:5 M:5 F:0 I:0 SB	Automated migration
_ encrypted_database	YES	Encrypted database
○ repl_site_id	0	0
○ authentication	SERVER	SERVER_ENCRYPT
○ instance_memory	3468896	598630

```
^Q•Quit ^F•Field Help ^X•Generate Overrides
```

This example shows many of the errors that can be reported by the Analysis step. Those items in red will stop a shift from occurring, while those in yellow are features which might cause an issue when the database is started in the target location. Details of the setting are available by pressing ^F while the cursor is on the line of the configuration parameter (next page).

```
Db2 Shift [/home/c2cdb2/c2c]

Review the settings below and select which ones should be overridden. Use ^?
to view details of the parameter and links to the documentation.
```

Parameter	Source	Destination
EXT_PROC_COUNT		
The system contains external procedures which are not in the standard Db2 library. You will need to manually recreate and catalog them.		
ESC•Close		
_ cur_eff_code_lvl	V:10 R:5 M:5 F:0 I:0 SB	Automated migration
_ encrypted_database	YES	Encrypted database
○ repl_site_id	0	0
○ authentication	SERVER	SERVER_ENCRYPT
○ instance_memory	3468896	598630

```
^Q•Quit ^F•Field Help ^X•Generate Overrides
```

Some fields will have additional help available through a web link.


```
Db2 Shift [/home/c2cdb2/c2c]

Review the settings below and select which ones should be overridden. Use ^?
to view details of the parameter and links to the documentation.

Parameter                Source                Destination

INSTANCE_MEMORY

Specifies the maximum amount of memory that can be allocated for a
database partition
Link: instance_memory

ESC+Close

_ encrypted_database     YES                  Encrypted database
o repl_site_id           0                   0
o authentication         SERVER              SERVER_ENCRYPT
o instance_memory        3468896             598630

Select which values should be placed into an override file.

^Q+Quit ^F+Field Help ^X+Generate Overrides
```

If you have access to a mouse, you will be able to click on the link in the field help display and have a web page display with more details on the parameter.

Overriding Parameters

The items that have circles beside them are values which can be overridden during the shift process. For instance, one of the settings that should change is the instance memory value. The source system currently shows 3.5M pages of memory being allocated, while the target system only provides 600K pages of memory. The database at the target system will end up being memory constrained and may suffer performance degradation because of the lack of memory.

To update the target value with the source value, the cursor must be placed on the setting line and the SPACEBAR pressed to select the item.

Once you have selected the settings to override, the ^X key is used to take all selected items in the list and place them into the overrides settings of the Shift or Deploy panel.

```
Overrides      : instance_memory 3468896 .....
```

Local Analysis

You have the option of checking whether a local database is suitable for a shift operation. This option is available at the bottom of the main menu:

```
Other    Analyze Local Database
         View Shift Log
```

When you select the Analyze Local Database option, the following screen will be displayed:

```
Db2 Shift [/home/db2inst1]

Use this panel to check if the current database (existing settings) or
another local database are suitable to be shifted to a Db2u environment.

Enter a database name below or leave blank to use existing settings.
|-----
Esc•Quit Enter•Analyze

ESC•Quit
```

Enter the name of a local database into this field and hit Enter. The program will analyze the database compared to an empty Db2u container. Note that this analysis is done against a sample installation so many of the settings that are flagged may not be applicable to a real deployment. The analysis will highlight any incompatibilities in the source database, so any warnings or errors that are reported will need to be investigated.

5

Db2 Shift Scenarios

DETAILS ON HOW TO SHIFT A DATABASE

Chapter 5: Shift Scenarios

There are 8 scenarios covered in this section. Each scenario represents a typical use of the Db2 Shift program. The scenarios can be summarized as:

- Shift a Db2 database to OpenShift, Kubernetes or CP4D
- Shift a Db2 database to another Db2 instance
- Create a Cloned copy of the Db2 database for later deployment
- Deploy a clone into an OpenShift, Kubernetes or CP4D container
- Deploy a clone into another Db2 instance
- Initialize HADR between Source and Target POD
- Initialize HADR between Source and Target Instance
- Initialize DMC and LDAP Authentication for CP4D
- Copy Cloned Databases to a POD

All scenarios assume that you are currently connected to the instance that has the Db2 database and have authenticated to OpenShift or Kubernetes before issuing the command.

Db2 Shift Command

The Db2 shift command is invoked with either of the following formats:

- `./db2shift` (if in local directory)
- `db2shift` (if installed in a PATH directory)

The program has three modes of operation:

- If the program is started without any parameters, it will go into full screen display mode and prompt you for the action you wish to take. You can use this prompt mode to generate any Db2 Shift command.
- If the `--help` option is used, the program will display a menu of help topics on the use of the Db2 Shift program.
- Otherwise, the Db2 Shift program will attempt to run the shift based on the settings provided.

The following sections describe the different Db2 Shift scenarios and what settings are required to run each one. Each scenario includes information on what fields are required and those that are optional. More details on the settings are discussed in the chapters dedicated to these scenarios. Use this section to determine what settings are required before attempting to do a shift operation.

Db2 Containerization to OpenShift, Kubernetes, or Cloud Pak for Data

Containerizing an existing Db2 database on an on-premises system to OpenShift, Kubernetes, or CP4D is the most common scenario for using Db2 Shift. To move a Db2 database to a POD, you will require the following information:

- Source Database details
- Destination location
- Type of Containerization environment
- Shift Options

The target of the Db2 Shift operation can be OpenShift, a Kubernetes cluster, or Cloud Pak for Data.

Required Options

```
--mode=[move | all]
--dest-type=POD
--oc or --kubectl
--source-dbname=flights
--source-owner=db2inst1
--dest-dbname=db2oltp
--dest-server=c-demo-db2u-0
```

Optional Settings

```
--dest-namespace=db2u
--hadr or --keep-rfw-pending
--online or --offline
--threads=4
--compress-level=0
--exclude-functions
--sync=[start_sync, rerun_sync, finish_sync]
--blank-slate(--new-settings)=[true|false]
--gen-settings
--verify
```

Db2 Shift to another Db2 Instance

This format of the Db2 Shift will take an existing Db2 database on an on-premises system and shift it to another traditional Db2 system hosted on another on-premises server or cloud virtual machine. This does not containerize Db2! The Db2 Shift command requires the following information:

- Source Database details

- Destination location
- Shift Options

The Db2 Shift program assumes that you are currently connected to the instance that has the Db2 database and have ssh connectivity to the target server.

Required Options

```
--mode=move
--ssh
--dest-type=OTHER
--source-dbname=flights
--source-owner=db2inst1
--dest-dbname=db2oltp
--dest-server=db2inst1@some.server.com
```

Optional Settings

```
--dest-create-db
--mirror-path
--hadr or --keep-rfw-pending
--online or --offline
--threads=4
--compress-level=0
--exclude-functions
--sync=[start_sync, rerun_sync, finish_sync]
--blank-slate(--new-settings)=[true|false]
--gen-settings
--verify
```

Clone an Existing Database

The Db2 Shift clone option is used to take an existing Db2 database that is currently on-premises and clone it into a directory. This cloned database can then be transported to another server and be deployed at that location.

The advantage of cloning is that the destination does not need to be connected to the source location and the deployment of the clone can be done at a more convenient time.

The Db2 Shift program requires the following information:

- Source Database details
- Clone Options

The destination details are not required to clone a database.

Required Options

```
--mode=clone
--source-dbname=flights
--source-owner=db2inst1
--clone-dir=/tmp/cache
```

Optional Settings

```
--online or --offline
--threads=4
--exclude-functions
--sync=[start_sync, rerun_sync, finish_sync]
--blank-slate(--new-settings)=[true|false]
--gen-settings
--verify
```

Clone Deployment to OpenShift, Kubernetes, or CP4D

This Db2 Shift option will take a database clone and deploy it into a Db2u pod running on OpenShift, Kubernetes or CP4D.

The panel requires the following information:

- Database name and clone location
- Destination POD details
- Type of Containerization environment
- Shift Options

Required Options

```
--mode=apply_clone
--dest-type=POD
--oc or --kubect1 or --local
--source-dbname=flights
--source-owner=db2inst1
--clone-dir=/tmp/cache
--dest-dbname=db2oltp
--dest-server=c-demo-db2u-0
```

Optional Settings

```
--dest-namespace=db2u
--hadr or --keep-rfw-pending
--online or --offline
--threads=4
--compress-level=0
--exclude-functions
--sync=[start_sync, rerun_sync, finish_sync]
```

```
--blank-slate(--new-settings)=[true|false]
--gen-settings
--verify
```

Clone Deployment to Another Db2 Instance

This Db2 Shift option will take a database clone and deploy it into another Db2 instance running natively (in any environment including Cloud VMs).

The panel requires the following information:

- Database name and clone location
- Destination instance details
- Shift Options

Required Options

```
--mode=apply_clone
--dest-type=OTHER
--ssh or --local
--source-dbname=flights
--source-owner=db2inst1
--clone-dir=/tmp/cache
--dest-dbname=db2oltp
--dest-owner=db2inst1
--dest-server=db2inst1@some.server.com
```

Optional Settings

```
--dest-create-db
--hadr or --keep-rfw-pending
--online or --offline
--threads=4
--compress-level=0
--exclude-functions
--sync=[start_sync, rerun_sync, finish_sync]
--blank-slate(--new-settings)=[true|false]
--gen-settings
--verify
```

Initialize HADR between Source and Target POD

This Db2 Shift option will take a source and destination (POD) database and start the HADR service between them. The Db2u pod must have been created with the `--hadr` setting during the shift step. The panel requires the following:

- The source database name and server
- The destination POD and server details

Required Options

```
--mode=hadr_setup
--dest-type=POD
--oc or --kubectl
--source-dbname=flights
--source-hadr-host=some.server.com
--source-hadr-port=3700
--dest-server=c-demo-db2u-0
--dest-hadr-host=oc.server.com
--dest-hadr-port=3700
```

Optional Settings

```
--dest-namespace=db2u
```

Initialize HADR between Source and Target Instance

This menu is like the previous one where the HADR service is setup between the source Db2 database and another Db2 instance. The Db2 database on the target system must have been created with one of the following settings during the shift step.

The panel requires the following information:

- The source database name and server
- The destination server details

Required Options

```
--mode=hadr_setup
--dest-type=OTHER
--ssh
--source-dbname=flights
--source-hadr-host=some.server.com
--source-hadr-port=3700
--dest-server=db2inst1@other.server.com
--dest-hadr-host=oc.server.com
--dest-hadr-port=3700
```

Initialize DMC and LDAP Authentication for CP4D

When shifting a database to a CP4D platform or LDAP-secured environment, the Db2 Shift program will automatically add the appropriate userids to the LDAP service and reset the Data Management Console (DMC) so that it recognizes the database.

If you choose to shift a database and set it up as an HADR secondary, the LDAP and DMC setup cannot be performed until after you have converted the secondary POD into the primary server. Only when the POD is the primary server can it be initialized to support LDAP and DMC services on IBM Cloud Pak for Data.

Required Options

```
--mode=sec_and_monitor
--dest-type=POD
--oc or --kubect1
--source-owner=db2inst1
--dest-dbname=flights
--dest-server=c-demo-db2u-0
```

Optional Settings

```
--dest-namespace=db2u
```

Clone Copy

This Db2 Shift command provides a feature that allows a user to copy an existing database clone copy to a POD, or to retrieve a database clone from a POD.

Once a database clone has been generated, the copy can be moved to any location and then deployed locally. This option provides a convenient way of copying the database using Db2 Shift without having to use OpenShift or Kubernetes commands.

The panel requires the following information:

- Type of copy (From Source to Target or Target to Source)
- Source cloned database directory
- Target cloned database directory
- The destination POD and server details

Required Options

```
--mode=push_clone (Source to Target)
--mode=pull_clone (Target to Source)
--dest-type=POD
--oc or --kubectl
--dest-server=c-demo-db2u-0
--local-dir=/tmp/cache
--dest-dir=/tmp/cache
```

Optional Settings

```
--dest-namespace=db2u
```

6

Containerize Db2 to OpenShift, Kubernetes or CP4D

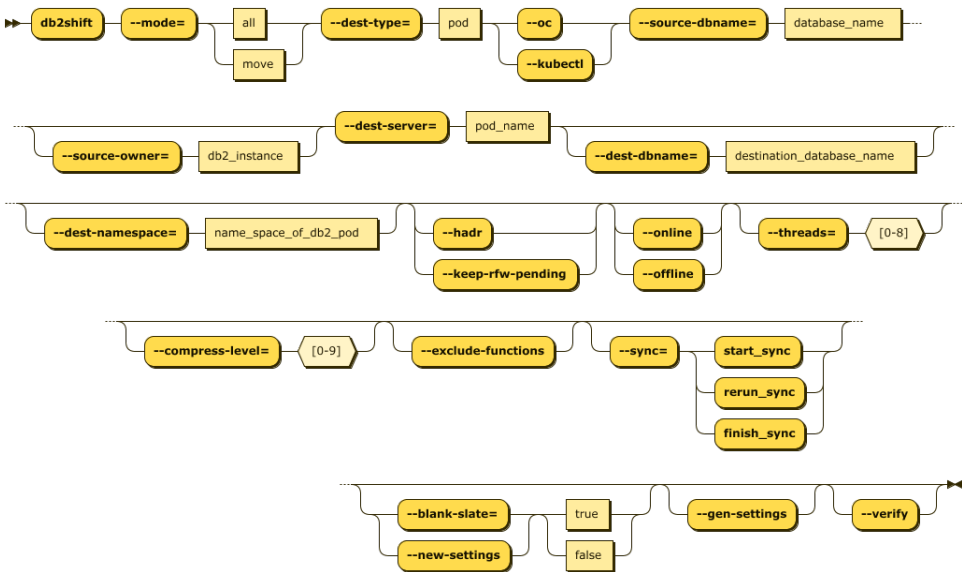
SHIFTING AN INSTANCE TO A POD

Chapter 6: Db2 Containerization to OpenShift, Kubernetes, or Cloud Pak for Data

Containerizing an existing Db2 database on an on-premises system to OpenShift, Kubernetes, or CP4D is the most common scenario for using Db2 Shift. To move a Db2 database to a POD you will require the following information:

- Source Database details
- Destination location
- Type of Containerization environment
- Shift Options

The target of the Db2 Shift operation can be OpenShift, a Kubernetes cluster, or Cloud Pak for Data.



Required Options

- ```

--mode=[move | all]
--dest-type=POD
--oc or --kubect1
--source-dbname=flights
--source-owner=db2inst1
--dest-dbname=db2oltp
--dest-server=c-demo-db2u-0

```

## Optional Settings

```
--dest-namespace=db2u
--hadr or --keep-rfw-pending
--online or --offline
--threads=4
--compress-level=4
--exclude-functions
--sync=[start_sync, rerun_sync, finish_sync]
--blank-slate(--new-settings)=[true|false]
--gen-settings
--verify
```

The panel that provides this capability:

```
Db2 Shift [/home/c2cdb2/c2c]
 Shift to Db2U on OpenShift or Kubernetes
Source Details
Database Name : flights_
Instance Owner : db2inst1_-----
Destination Details
Target Location : OpenShift Kubernetes
Database Name : db2oltp_
POD Project : db2u_-----+
POD Name : c-demo-db2u-0_-----+
Metadata : Refresh None Settings Only Verify
Sync Options : None Initialize Refresh Finalize
Move Options : Database Database/LDAP/DMC Move Database for HADR
Exclude Routines : Do not move external routines
Database Mode : Online Move Offline Move
Thread count : 0
Compression : 4
Overrides : -----+
ESC•Quit ^?•Field Help ^A•Analyze ^X•Review and Execute
```

To view this panel directly, use the syntax:

- `db2shift --shiftpod`

## Mode Option

The MODE option determines what steps the Db2 Shift program will take to move your database to the new location. The following are the values that move can have.

Syntax: `--mode=[move,all]`

```
Move Options : Database Database/LDAP/DMC Move Database for HADR
```

## Move Database Only

Syntax: `--mode=move`

```
Move Options : ● Database ○ Database/LDAP/DMC ○ Move Database for HADR
```

The MOVE option will take a copy of your Db2 database and copy it into the target Db2U container or instance. No other processing is done with the database.

## Move Database and Set up LDAP and DMC Settings

Syntax: `--mode=all`

```
Move Options : ○ Database ● Database/LDAP/DMC ○ Move Database for HADR
```

If you choose the ALL option, the database will be moved, and the program will also apply any security and console settings that are required. This is equivalent to running `mode=move` and `mode=sec_and_monitor` one after another. The ALL option is necessary in CP4D environments where the LDAP settings need to be updated and the DMC console needs to be reset to recognize the new database.

## Target Client (Instance to POD)

Syntax: `--oc`, `--kubect1`

```
Target Location : ● OpenShift ○ Kubernetes
```

The pod client for a deploy (clone) operation must be supplied as part of the Db2 Shift command. Only one of the following clients must be used:

- `--oc` OpenShift Destination
- `--kubect1` Kubernetes Destination

If the client is Kubernetes (`--kubect1`) or OpenShift (`--oc`), the program requires that the appropriate `kubect1` or `oc` client has been installed locally and that the namespace or project has already been specified.

## Source Database

Syntax: `--source-database=""`

```
Database Name : 📁lights_
```

The source database is the name of the database that you want to move to the new location. Note that you can have the same or different database name at the target. If you provide a different database name at the target, the program will copy the database from the source and place it on the target and use the existing name.

### Alert!

If you are shifting a database to Cloud Pak for Data, and your INSTANCE userid is not db2inst1, you must execute the following SQL commands from a userid that has SECADM authority:

```
GRANT SECADM ON DATABASE TO USER db2inst1
GRANT DBADM ON DATABASE TO USER db2inst1
```

The db2inst1 userid does not need to exist in the Operating system to grant these privileges to the userid. This requirement does not apply to other Db2 Shift environments. If db2inst1 is not defined as a SECADM and DBADM user in the database, the Data Management Console feature of CP4D will not be able to access the database nor will it be able to monitor it.

## Source or Instance Owner

Syntax: `--source-owner="instance name"`

```
Instance Owner : db2inst1_.....
```

The Db2 Shift program assumes that the current userid you are logged into is the owner of the instance. This is necessary due to the requirement to access the underlying files that are used by Db2. If you supply the source owner value, Db2 Shift will double check that you are working in the correct instance and the settings files are consistent.

## Destination Database

Syntax: `--dest-database=""`

```
Database Name : lights_
```

The destination database name can be the same as the source database, or a completely different name. Make sure that if you are creating a database with a different name that it doesn't currently exist on your target system. Otherwise, the target database will be deleted!

## Destination Server (POD)

Syntax: `--dest-server="pod_name"`

```
POD Name : c-demo-db2u-0_.....
```

For deployments to OpenShift, Kubernetes, or CP4D, you must supply the name of the POD that Db2U is running in. The OpenShift or Kubernetes client should be used to connect to the target namespace or project before issuing the Db2 Shift command.



## Stored Procedures and Functions

Syntax: `--exclude-functions`

```
Exclude Routines : Do not move external routines
```

By default, the Db2 Shift command will automatically move all external stored procedures and functions that are found in the `$HOME/sqllib/function` path to the new destination. All SQLPL and PL/SQL routines are moved as part of the database move, so there is no migration required for those routines.

Use the exclude flag to prevent any external functions being moved to the destination.

## Destination Pod Namespace or Project

Syntax: `--dest-namespace=""`, `--dest-project=""`

```
POD Project : db2u_-----
```

In Kubernetes deployments, the location of a pod is associated with a namespace, while in OpenShift deployments, the pod is associated with a project.

When authenticating to a Kubernetes or OpenShift environment, it is recommended that the local client be connected to the project or namespace that the Db2U pod is running in.

If you do not supply a namespace or project value, the Db2 Shift program will assume that you are already connected to that project. If this is not the case, the program will stop with an error when it attempts to find the pod.

To have Db2 Shift connect to the appropriate project or namespace, supply the value of the namespace or project using this option.

## HADR Setup

Syntax: `--hadr` or `--keep-rfw-pending`

```
Move Options : Database Database/LDAP/DMC Move Database for HADR
```

When HADR mode is selected, the database will be copied over to the target location and initialized as an HADR secondary. The database can now connect to the primary database as an HADR pair and participate in failover or read-only applications.

This option cannot be used if your database needs to be migrated or if the database needs to be renamed.

## Synchronization Options

Syntax: `--sync=[start_sync, rerun_sync, finish_sync]`

```
Sync Options : ● None ○ Initialize ○ Refresh ○ Finalize
```

The Db2 Shift program has two methods of moving the database to the destination. The traditional method is to take one pass at the database and move everything at once. During the finishing process, the program will briefly suspend the database (depending on settings) and perform a final refresh. This last step will pick up any database objects that may have changed over the course of the move.

The total amount of time the database is suspended is directly related to how much information has changed during the shift process. If you have large number of changes, the final refresh may be too long an outage. To minimize this outage, you can use the synchronization options provided.

The default setting is do a complete shift operation without intermediate sync operations.

To initialize the synchronization option, the first Db2 Shift command will use the sync option.

Syntax: `--sync=start_sync`

```
Sync Options : ○ None ● Initialize ○ Refresh ○ Finalize
```

This initial step will instruct Db2 Shift to copy all the required database objects to the target system. Once the copies are complete, the program will end processing and leave the target system in an incomplete state. During this process the source database is operational, and it will not be suspended.

When there is an appropriate moment, the database movement can be finalized with the `finish_sync` option.

Syntax: `--sync=finish_sync`

```
Sync Options : ○ None ○ Initialize ○ Refresh ● Finalize
```

The `finish_sync` option will do one final pass against the source database and then it will finalize the database movement on the destination site. During the last pass the finish process will suspend the database to get a consistent database environment. Once this step completes, the destination database will be available.

If the source system has high update volumes, there may be a need to do multiple sync operations to minimize the finalize step. The Db2 Shift command will need to be told that it is syncing the database again, but not to start from scratch. The control files generated by the `--sync=start_sync` option will allow

Db2 Shift to move database objects that are new or have changed since the initial synchronization request. The command to do the sync and only refresh the database objects requires the use of the `rerun_sync` option.

Syntax: `--sync=rerun_sync`

```
Sync Options : None Initialize Refresh Finalize
```

The `rerun_sync` option indicates to the Db2 Shift program to start syncing process to look for delta changes only.

The `--sync=rerun_sync` process can be run repeatedly until the number of changes between runs is minimized. When an appropriate timeslot is available, the shift can be finalized by using the `finish_sync` option.

Syntax: `--sync=finish_sync`

```
Sync Options : None Initialize Refresh Finalize
```

The `finish_sync` option will perform the final pass against the source database and complete the shift process as before.

In summary, the standard shift operation will complete in one step (no `--sync` option is used). The `--sync=start_sync` option allows you to gradually move a database over time. For a gradual database shift, use the `start_sync` option on the first run. This will move an initial set of database files to the target. Then use the `rerun_sync` option on subsequent runs to copy any files that may have been added or changed to the database. When you are ready to finalize the shift, use the `finish_sync` option to gather any remaining files and complete the shift operation on the target system.

## Metadata Generation

Syntax: `--blank-slate=[true|false], --gen-settings, --verify-only`

```
Metadata : Refresh None Settings Only Verify
```

The Db2 Shift command generates metadata that is used during the shift process. This metadata is key to determining which objects need to be moved from a source to destination as well as validating that the source can be successfully moved.

Generating metadata requires access to the source and destination systems. If for some reason the connection to the source or destination is unavailable, the existing metadata files can still be used. In most cases you will not need to adjust these settings unless you have encountered a shift error.

## Blank Slate

Syntax: `--blank-slate=true`

```
Metadata : ● Refresh ○ None ○ Settings Only ○ Verify
```

The `--blank-slate` option determines whether the existing metadata is refreshed. The default is value for `--blank-slate` is `true` which will display in the GUI as Refresh. The `--blank-slate` option will delete any existing metadata files and recreate them on your system during the shift operation. The default option is `true` which will result in new source and destination settings being generated.

If you are importing settings files from other systems, or if you need to rerun the shift process without regenerating the files, use the `--blank-slate=false` option. When Db2 Shift executes it will use the existing metadata in the working directory and attempt to use those settings.

Syntax: `--blank-slate=false`

```
Metadata : ○ Refresh ● None ○ Settings Only ○ Verify
```

One scenario that involves the use of `--blank-slate=false` occurs when a shift operation fails at the target OC/Kubernetes pod because of a communication error. The database at the destination will be left in an inconsistent state and must be rebuilt by the shift process. The settings for the destination database can no longer be retrieved because the database cannot be started. Because of this reason, you must use the existing destination settings that were generated when you first ran the Db2 Shift command.

## Generate Settings

Syntax: `--blank-slate=[true|false], --gen-settings`

```
Metadata : ○ Refresh ○ None ● Settings Only ○ Verify
```

The `--gen-settings` (Generate Settings) option is used in conjunction with `--blank-slate`. The use of `--gen-settings` will prevent Db2 Shift from continuing execution after the metadata files have been created.

## Verification

Syntax: `--verify-only`

```
Metadata : ○ Refresh ○ None ○ Settings Only ● Verify
```

The `--verify-only` option will generate the metadata files and check the connectivity and all settings and then stop execution. If `--verify-only`

completes successfully, the Db2 Shift command will be able to execute the shift process.

## Online or Offline Move

Syntax: `--online`, `--offline`

```
Database Mode : ● Online Move ○ Offline Move
```

Db2 Shift provides two options when dealing with the state of a database during the shift process:

- `--online` - database is online while the shift is taking place
- `--offline` - database has been shut down

By default, the Db2 Shift command assumes you will use online mode and will suspend the database while it completes the last scan of the files. During this period, the database will not complete any insert, update, or delete transactions. This will result in a consistent database at the destination, but some transactions will not have been committed to the database. If the database at the destination must be identical to the source database, then you must completely shut down the database and choose the `--offline` option.

During the last step of the shift process, Db2 Shift will search for any updates that may have been applied to the database after the initial copying was done. To ensure the integrity of the data being copied, the database will be placed into a WRITE SUSPEND mode. This last step should only take a few seconds and the database will be WRITE RESUMED when the final copy is done.

When the database is suspended, all read activities will continue. New connections will not be permitted, but any applications that are currently running will be allowed to continue. Those transactions that are updating records will be temporarily "paused" while the copying is done. Once the copy is complete, a suspended application will finish their transaction.

The period that the database is suspended or stopped is dependent on the changes that have occurred in the database from the time the shift operation started to the time when copying of the data is complete. The changes that have occurred during this period needs to be captured during the final step. It is during this step that the database must be suspended or stopped. During the initial scan the database will remain completely online and will not be impacted by the shift utility from a transaction perspective. However, since there is a large amount of disk reads taking place, it may impact buffer pool read performance.

The offline mode must be used when shifting a database that requires a migration from an older release of Db2. This option must be specified when shifting Db2 versions 10.5 or 11.1.

To use offline mode, you must run Db2 Shift against the source and target using:

- Shift command with `--verify-only`  
This step will generate a copy of the control files needed
- Db2 Shift operation using the offline mode `--offline`

If you do not create the control files beforehand, the Db2 Shift operation will not have the necessary control files to run the shift.

## Performance: Threading

Syntax: `--threads=[1...8]`

```
Thread count : 4
```

The copy phase of the Db2 Shift program can use multiple threads to transmit data to a destination. This setting allows you to increase the parallelism up to 8 threads. As you increase the number of threads, the amount of data being transmitted increases, at the expense of greater CPU usage and network congestion. The default value is 4 which strikes a balance between overhead and network performance.

## Performance: Compression

Syntax: `--compression=[0...9]`

```
Compression : 4
```

RSYNC compresses the data during the transfer process to allow for faster movement of data. The amount of compression can be adjusted from 0 to 9 with 0 turning off compression and values between 1 and 9 increasing the amount of compression applied to the data. Higher compression values will result in more CPU usage and may not significantly reduce the size of the data stream.

A value of 4 has been found to be a good compromise between compression overhead and data size on slow networks (< 1Gb/s). For high-speed networks, a value of 0 is recommended unless there is a requirement to reduce network traffic.

# 7

## Db2 Shift Instance to Instance

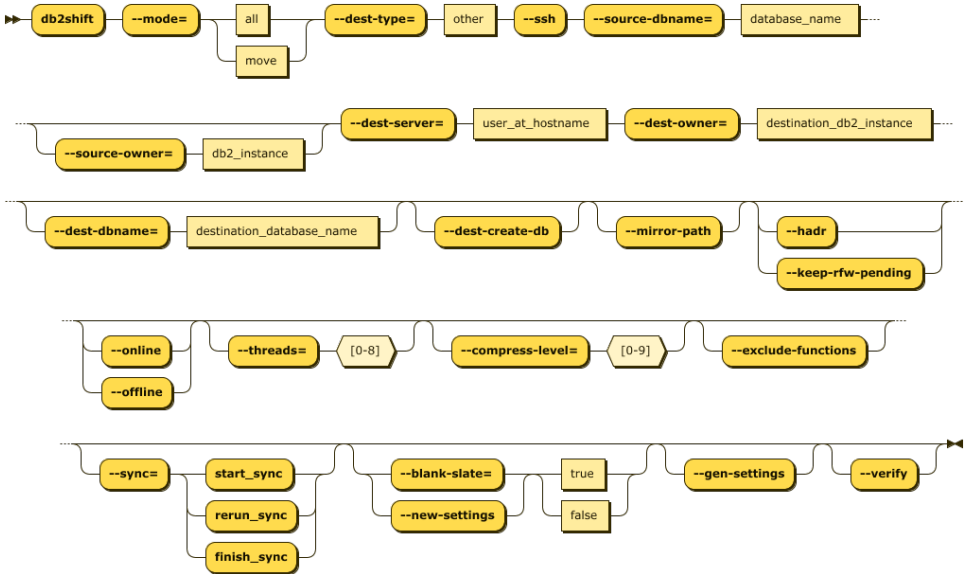
SHIFTING TO ANOTHER INSTANCE

## Chapter 7: Db2 Shift to Another Db2 Instance

This format of the Db2 Shift will take an existing Db2 database on an on-premises system and shift it to another traditional Db2 system hosted on another on-premises server or cloud virtual machine. This does not containerize Db2! The Db2 Shift command requires the following information:

- Source Database details
- Destination location
- Shift Options

The Db2 Shift program assumes that you are currently connected to the instance that has the Db2 database and have ssh connectivity to the target server.



### Required Options

- ```

--mode=move
--ssh
--dest-type=OTHER
--source-database=flights
--source-owner=db2inst1
--dest-database=db2oltp
--dest-server=db2inst1@some.server.com
    
```


Optional Settings

```
--hadr or --keep-rfw-pending
--online or --offline
--dest-create-db
--mirror-path
--threads=4
--compress-level=4
--exclude-functions
--sync=[start_sync, rerun_sync, finish_sync]
--blank-slate(--new-settings)=[true|false]
--gen-settings
--verify
```

The panel that provides this capability:

```
Db2 Shift [/home/c2cdb2/c2c]

Shift Db2 to another Db2 instance

Source Details
Database Name      : -----
Instance Owner    : -----

Destination Details
Database Name      : -----    Create Database    Use Mirrored Paths
Instance Owner    : -----
Server Address     : -----+

Metadata          :  Refresh    None    Settings Only    Verify
Sync Options      :  None    Initialize    Refresh    Finalize
HADR Option       :  Move Database for HADR
Exclude Routines  :  Do not move external routines
Database Mode     :  Online Move    Offline Move
Thread Count      : 4
Compression       : 0
Overrides         : -----+

ESC•Quit ^F•Field Help ^A•Analyze ^X•Review and Execute
```

To view this panel directly, use the syntax:

- `db2shift --shiftdb2`

Mode Option

Syntax: `--mode=move`

The MODE option determines what steps the Db2 Shift program will take to move your database to the new location. The only valid option is move when shifting a database between two Db2 instances.

Target Client (Instance to Instance)

Syntax: `--ssh`

The client for a shift or deploy (clone) operation must be supplied as part of the Db2 Shift command. If the target is a remote Db2 instance (`--ssh`), Db2 Shift expects that a passwordless ssh environment has been established between the source and target servers.

Source Database

Syntax: `--source-database=""`

```
Database Name      : lights_
```

The source database is the name of the database that you want to move to the new location. Note that you can have the same or different database name at the target. If you provide a different database name at the target, the program will copy the database from the source and place it on the target and use the existing name.

Source or Instance Owner

Syntax: `--source-owner="instance name"`

```
Instance Owner     : db2inst1_
```

The Db2 Shift program assumes that the current userid you are logged into is the owner of the instance. This is necessary due to the requirement to access the underlying files that are used by Db2. If you supply the source owner value, Db2 Shift will double check that you are working in the correct instance and the settings files are consistent.

Destination Database

Syntax: `--dest-database=""`

```
Database Name      : lights_
```

The destination database name can be the same as the source database, or a completely different name. Make sure that if you are creating a database with a different name that it doesn't currently exist on your target system. Otherwise, the target database will be deleted!

Force Destination Database Creation

Syntax: `--dest-create-db`

```
Database Name      : _____  o Create Database  o Use Mirrored Paths
```

During a shift or clone operation to a traditional Db2 INSTANCE, the Db2 Shift program will check whether the target database exists. If it does not, the Db2 Shift operation will fail. If the Force Database Creation setting is on, Db2 Shift will attempt to create the database first and then run the shift operation.

Note: Db2 Shift will create the database in a directory structure under `/db2_portable` on the target system.

Mirror Path

Syntax: `--mirror-path`

```
Database Name      : _____  o Create Database  o Use Mirrored Paths
```

Mirror path is a valid switch for Applying a Clone or moving a database to a standard db2 instance.

It will take all the paths, except the default database path, and use them as the destination paths on the target instance. It assumes that those paths exist already, or it can create them. The database names and instance owners must also be the same.

If used in combination with `--dest-create-db`, the database that is created, rather than being created under the `/home/$DB2INST/db2_portable` directory, will be created on the same path as the source system. If using this option into a path (even if it is the default database path) you may receive an error if the database configuration directory number e.g., `SQL00001` does not match across source and target systems. In this case you should manually create the target database on a different path. This is a limitation of the Db2 relocation utility.

When using this option, you must be cautious that the target paths on the destination are on paths that the instance owner has permissions to create the underlying directory structures, and that you have free space to create the database.

Destination Server (Instance)

Syntax: `--dest-server=userid@ip.address`

```
Server Address : some.server.com_-----
```

For destinations that are traditional Db2 instances, you must provide the userid and the IP address or symbolic name of the destination server. You only use an IP address for shifting into a traditional Db2 instance.

The format of the parameter is `userid@address` when using the Db2 Shift command line.

When connecting to a remote instance, the Db2 Shift program expects that a passwordless ssh environment has been established between the source and target servers. The `--ssh` flag must be used in conjunction with this parameter.

Stored Procedures and Functions

Syntax: `--exclude-functions`

```
Exclude Routines :  Do not move external routines
```

By default, the Db2 Shift command will automatically move all external stored procedures and functions that are found in the `$HOME/sqllib/function` path to the new destination. All SQLPL and PL/SQL routines are moved as part of the database move, so there is no migration required for those routines.

Use the exclude flag to prevent any external functions being moved to the destination.

HADR Setup

Syntax: `--hadr` or `--keep-rfw-pending`

```
HADR Option :  Move Database for HADR
```

When HADR option is selected, the database will be copied over to the target location and initialized as an HADR secondary. The database can now connect to the primary database as an HADR pair and participate in failover or read-only applications.

This option cannot be used if your database needs to be migrated or if the database needs to be renamed.

Synchronization Options

Syntax: `--sync=[start_sync, rerun_sync, finish_sync]`

Sync Options : None Initialize Refresh Finalize

The Db2 Shift program has two methods of moving the database to the destination. The traditional method is to take one pass at the database and move everything at once. During the finishing process, the program will briefly suspend the database (depending on settings) and perform a final refresh. This last step will pick up any database objects that may have changed over the course of the move.

The total amount of time the database is suspended is directly related to how much information has changed during the shift process. If you have large number of changes, the final refresh may too long an outage. To minimize this outage, you can use the synchronization options provided.

The default setting is do a complete shift operation without intermediate sync operations.

To initialize the synchronization option, the first Db2 Shift command will use the sync option.

Syntax: `--sync=start_sync`

Sync Options : None Initialize Refresh Finalize

This initial step will instruct Db2 Shift to copy all the required database objects to the target system. Once the copies are complete, the program will end processing and leave the target system in an incomplete state. During this process the source database is operational, and it will **not be suspended**.

When there is an appropriate moment, the database movement can be finalized with the `finish_sync` option.

Syntax: `--sync=finish_sync`

Sync Options : None Initialize Refresh Finalize

The `finish_sync` option will do one final pass against the source database and then it will finalize the database movement on the destination site. During the last pass the finish process will suspend the database to get a consistent database environment. Once this step completes, the destination database will be available.

If the source system has high update volumes, there may be a need to do multiple sync operations to minimize the finalize step. The Db2 Shift command will need to be told that it is syncing the database again, but not to start from scratch. The control files generated by the `--sync=start_sync` option will allow

Db2 Shift to move database objects that are new or have changed since the initial synchronization request. The command to do the sync and only refresh the database objects requires the use of the `rerun_sync` option.

Syntax: `--sync=rerun_sync`

```
Sync Options      :  None  Initialize  Refresh  Finalize
```

The `rerun_sync` option indicates to the Db2 Shift program to start syncing process to look for delta changes only.

The `--sync=rerun_sync` process can be run repeatedly until the number of changes between runs is minimized. When an appropriate timeslot is available, the shift can be finalized by using the `finish_sync` option.

Syntax: `--sync=finish_sync`

```
Sync Options      :  None  Initialize  Refresh  Finalize
```

The `finish_sync` option will perform the final pass against the source database and complete the shift process as before.

In summary, the standard shift operation will complete in one step (no `--sync` option is used). The `--sync=start_sync` option allows you to gradually move a database over time. For a gradual database shift, use the `start_sync` option on the first run. This will move an initial set of database files to the target. Then use the `rerun_sync` option on subsequent runs to copy any files that may have been added or changed to the database. When you are ready to finalize the shift, use the `finish_sync` option to gather any remaining files and complete the shift operation on the target system.

Metadata Generation

Syntax: `--blank-slate=[true|false], --gen-settings, --verify-only`

```
Metadata          :  Refresh  None  Settings Only  Verify
```

The Db2 Shift command generates metadata that is used during the shift process. This metadata is key to determining which objects need to be moved from a source to destination as well as validating that the source can be successfully moved.

Generating metadata requires access to the source and destination systems. If for some reason the connection to the source or destination is unavailable, the existing metadata files can still be used. In most cases you will not need to adjust these settings unless you have encountered a shift error.

Blank Slate

Syntax: `--blank-slate=true`

Metadata : Refresh None Settings Only Verify

The `--blank-slate` option determines whether the existing metadata is refreshed. The default is value for `--blank-slate` is `true` which will display in the GUI as Refresh. The `--blank-slate` option will delete any existing metadata files and recreate them on your system during the shift operation. The default option is `true` which will result in new source and destination settings being generated.

If you are importing settings files from other systems, or if you need to rerun the shift process without regenerating the files, use the `--blank-slate=false` option. When Db2 Shift executes it will use the existing metadata in the working directory and attempt to use those settings.

Syntax: `--blank-slate=false`

Metadata : Refresh None Settings Only Verify

One scenario that involves the use of `--blank-slate=false` occurs when a shift operation fails at the target OC/Kubernetes pod because of a communication error. The database at the destination will be left in an inconsistent state and must be rebuilt by the shift process. The settings for the destination database can no longer be retrieved because the database cannot be started. Because of this reason, you must use the existing destination settings that were generated when you first ran the Db2 Shift command.

Generate Settings

Syntax: `--blank-slate=true, --gen-settings`

Metadata : Refresh None Settings Only Verify

The `--gen-settings` (Generate Settings) option is used in conjunction with `--blank-slate`. The use of `--gen-settings` will prevent Db2 Shift from continuing execution after the metadata files have been created.

Verification

Syntax: `--verify-only`

Metadata : Refresh None Settings Only Verify

The `--verify-only` option will generate the metadata files and check the connectivity and all settings and then stop execution. If `--verify-only`

completes successfully, the Db2 Shift command will be able to execute the shift process.

Online or Offline Move

Syntax: `--online`, `--offline`

```
Database Mode      : ● Online Move ○ Offline Move
```

Db2 Shift provides two options when dealing with the state of a database during the shift process:

- `--online` - database is online while the shift is taking place
- `--offline` - database has been shut down

By default, the Db2 Shift command assumes you will use online mode and will suspend the database while it completes the last scan of the files. During this period, the database will not complete any insert, update, or delete transactions. This will result in a consistent database at the destination, but some transactions will not have been committed to the database. If the database at the destination must be identical to the source database, then you must completely shut down the database and choose the `--offline` option.

During the last step of the shift process, Db2 Shift will search for any updates that may have been applied to the database after the initial copying was done. To ensure the integrity of the data being copied, the database will be placed into a WRITE SUSPEND mode. This last step should only take a few seconds and the database will be WRITE RESUMED when the final copy is done.

When the database is suspended, all read activities will continue. New connections will not be permitted, but any applications that are currently running will be allowed to continue. Those transactions that are updating records will be temporarily "paused" while the copying is done. Once the copy is complete, a suspended application will finish their transaction.

The period that the database is suspended or stopped is dependent on the changes that have occurred in the database from the time the shift operation started to the time when copying of the data is complete. The changes that have occurred during this period needs to be captured during the final step. It is during this step that the database must be suspended or stopped. During the initial scan the database will remain completely online and will not be impacted by the shift utility from a transaction perspective. However, since there is a large amount of disk reads taking place, it may impact buffer pool read performance.

The offline mode must be used when shifting a database that requires a migration from an older release of Db2. This option must be specified when shifting Db2 versions 10.5 or 11.1.

To use offline mode, you must run Db2 Shift against the source and target using:

Chapter 7: Shifting Db2 to another Db2 Instance

- Shift command with `--verify-only`
This step will generate a copy of the control files needed
- Db2 Shift operation using the offline mode `--offline`

If you do not create the control files beforehand, the Db2 Shift operation will not have the necessary control files to run the shift.

Performance: Threading

Syntax: `--threads=[1...8]`

```
Thread count      : 4
```

The copy phase of the Db2 Shift program can use multiple threads to transmit data to a destination. This setting allows you to increase the parallelism up to 8 threads. As you increase the number of threads, the amount of data being transmitted increases, at the expense of greater CPU usage and network congestion. The default value is 4 which strikes a balance between overhead and network performance.

Performance: Compression

Syntax: `--compression=[0...9]`

```
Compression      : 4
```

RSYNC compresses the data during the transfer process to allow for faster movement of data. The amount of compression can be adjusted from 0 to 9 with 0 turning off compression and values between 1 and 9 increasing the amount of compression applied to the data. Higher compression values will result in more CPU usage and may not significantly reduce the size of the data stream.

A value of 4 has been found to be a good compromise between compression overhead and data size on slow networks (< 1Gb/s). For high-speed networks, a value of 0 is recommended unless there is a requirement to reduce network traffic.

8

Clone an Existing Database

SHIFTING A DATABASE TO AIR GAPPED
ENVIRONMENTS

Chapter 8: Clone an Existing Database

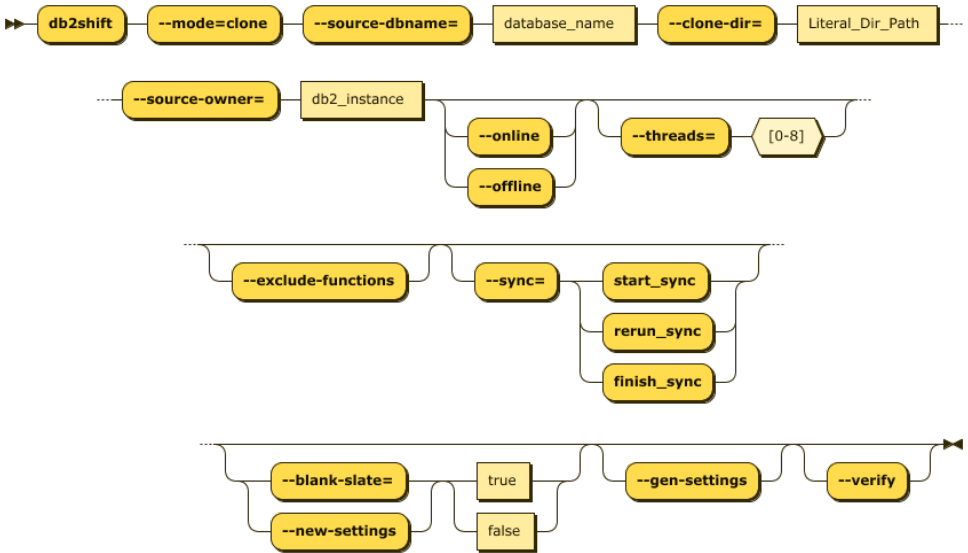
The Db2 Shift clone option is used to take an existing Db2 database that is currently on-premises and clone it into a directory. This cloned database can then be transported to another server and be deployed at that location.

The advantage of cloning is that the destination does not need to be connected to the source location and the deployment of the clone can be done at a more convenient time.

The Db2 Shift program requires the following information:

- Source Database details
- Clone Options

The destination details are not required to clone a database.



Required Options

```

--mode=clone
--source-dbname=flights
--source-owner=db2inst1
--clone-dir=/tmp/cache
  
```

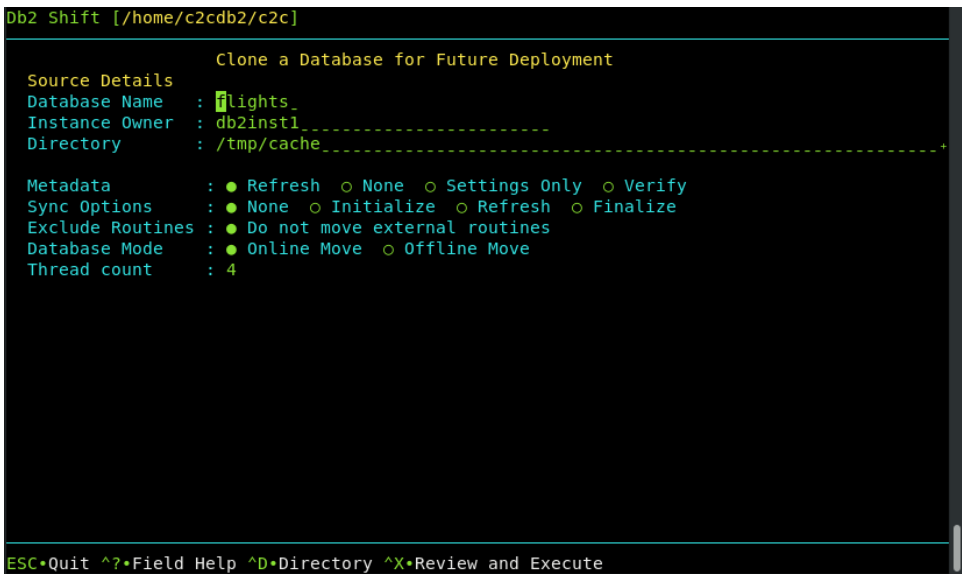
Optional Settings

```

--online or --offline
--threads=4
--exclude-functions
--sync=[start_sync, rerun_sync, finish_sync]
  
```

```
--blank-slate(--new-settings)=[true|false]
--gen-settings
--verify
```

The panel that provides this capability:



```
Db2 Shift [/home/c2cdb2/c2c]
Clone a Database for Future Deployment
Source Details
Database Name : lights_
Instance Owner : db2inst1
Directory : /tmp/cache
Metadata : ● Refresh ○ None ○ Settings Only ○ Verify
Sync Options : ● None ○ Initialize ○ Refresh ○ Finalize
Exclude Routines : ● Do not move external routines
Database Mode : ● Online Move ○ Offline Move
Thread count : 4
ESC•Quit ^?•Field Help ^D•Directory ^X•Review and Execute
```

To view this panel directly, use the syntax:

- db2shift --clone

Mode Option

Syntax: --mode=clone

The MODE option determines what steps the Db2 Shift program will take to move your database to the new location.

The CLONE option can be used to create a copy of database to be moved to another location and then deployed. This feature is useful in situations where the target system cannot be reached through the network due to firewall issues.

Source Database

Syntax: --source-database=" "



```
Database Name : lights_
```

The source database is the name of the database that you want to move to the new location. Note that you can have the same or different database name at the target. If you provide a different database name at the target, the program

will copy the database from the source and place it on the target and use the existing name.

Alert!

If you are going to use this cloned copy to shift the database to Cloud Pak for Data, and your INSTANCE userid is **not** db2inst1, you must execute the following SQL commands from a userid that has SECADM authority:

```
GRANT SECADM ON DATABASE TO USER db2inst1
GRANT DBADM ON DATABASE TO USER db2inst1
```

The db2inst1 userid does not need to exist in the Operating system to grant these privileges to the userid. This requirement does not apply to other Db2 Shift environments. If db2inst1 is not defined as a SECADM and DBADM user in the database, the Data Management Console feature of CP4D will not be able to access the database nor will it be able to monitor it.

Source or Instance Owner

Syntax: `--source-owner="instance name"`

```
Instance Owner : db2inst1_-----
```

The Db2 Shift program assumes that the current userid you are logged into is the owner of the instance. This is necessary due to the requirement to access the underlying files that are used by Db2. If you supply the source owner value, Db2 Shift will double check that you are working in the correct instance and the settings files are consistent.

Clone Directory

Syntax: `--clone-dir=""`

```
Directory : /tmp/cache_-----
```

The cloned copy of the database will be placed into the directory that is specified by this field. When the database is cloned, the contents of the directory can be moved to a new location and the `--mode=apply_clone` option used to shift the contents of the clone into the Db2 database.

Stored Procedures and Functions

Syntax: `--exclude-functions`

```
Exclude Routines : o Do not move external routines
```

By default, the Db2 Shift command will automatically move all external stored procedures and functions that are found in the `$HOME/sqlllib/function` path to

the new destination. All SQLPL and PL/SQL routines are moved as part of the database move, so there is no migration required for those routines.

Use the exclude flag to prevent any external functions being moved to the destination.

Synchronization Options

Syntax: `--sync=[start_sync, rerun_sync, finish_sync]`

```
Sync Options      : ● None ○ Initialize ○ Refresh ○ Finalize
```

The Db2 Shift program has two methods of moving the database to the destination. The traditional method is to take one pass at the database and move everything at once. During the finishing process, the program will briefly suspend the database (depending on settings) and perform a final refresh. This last step will pick up any database objects that may have changed over the course of the move.

The total amount of time the database is suspended is directly related to how much information has changed during the shift process. If you have large number of changes, the final refresh may too long an outage. To minimize this outage, you can use the synchronization options provided.

The default setting is do a complete shift operation without intermediate sync operations.

To initialize the synchronization option, the first Db2 Shift command will use the sync option.

Syntax: `--sync=start_sync`

```
Sync Options      : ○ None ● Initialize ○ Refresh ○ Finalize
```

This initial step will instruct Db2 Shift to copy all the required database objects to the target system. Once the copies are complete, the program will end processing and leave the target system in an incomplete state. During this process the source database is operational, and it will **not be suspended**.

When there is an appropriate moment, the database movement can be finalized with the `finish_sync` option.

Syntax: `--sync=finish_sync`

```
Sync Options      : ○ None ○ Initialize ○ Refresh ● Finalize
```

The `finish_sync` option will do one final pass against the source database and then it will finalize the database movement on the destination site. During the last pass the finish process will suspend the database to get a consistent

database environment. Once this step completes, the destination database will be available.

If the source system has high update volumes, there may be a need to do multiple sync operations to minimize the finalize step. The Db2 Shift command will need to be told that it is syncing the database again, but not to start from scratch. The control files generated by the `--sync=start_sync` option will allow Db2 Shift to move database objects that are new or have changed since the initial synchronization request. The command to do the sync and only refresh the database objects requires the use of the `rerun_sync` option.

Syntax: `--sync=rerun_sync`

```
Sync Options      :  None  Initialize  Refresh  Finalize
```

The `rerun_sync` option indicates to the Db2 Shift program to start syncing process to look for delta changes only.

The `--sync=rerun_sync` process can be run repeatedly until the number of changes between runs is minimized. When an appropriate timeslot is available, the shift can be finalized by using the `finish_sync` option.

Syntax: `--sync=finish_sync`

```
Sync Options      :  None  Initialize  Refresh  Finalize
```

The `finish_sync` option will perform the final pass against the source database and complete the shift process as before.

In summary, the standard shift operation will complete in one step (no `--sync` option is used). The `--sync=start_sync` option allows you to gradually move a database over time. For a gradual database shift, use the `start_sync` option on the first run. This will move an initial set of database files to the target. Then use the `rerun_sync` option on subsequent runs to copy any files that may have been added or changed to the database. When you are ready to finalize the shift, use the `finish_sync` option to gather any remaining files and complete the shift operation on the target system.

Metadata Generation

Syntax: `--blank-slate=[true|false], --gen-settings, --verify-only`

```
Metadata          :  Refresh  None  Settings Only  Verify
```

The Db2 Shift command generates metadata that is used during the shift process. This metadata is key to determining which objects need to be moved from a source to destination as well as validating that the source can be successfully moved.

Generating metadata requires access to the source and destination systems. If for some reason the connection to the source or destination is unavailable, the existing metadata files can still be used. In most cases you will not need to adjust these settings unless you have encountered a shift error.

Blank Slate

Syntax: `--blank-slate=true`

Metadata : ● Refresh ○ None ○ Settings Only ○ Verify

The `--blank-slate` option determines whether the existing metadata is refreshed. The default value for `--blank-slate` is `true` which will display in the GUI as Refresh. The `--blank-slate` option will delete any existing metadata files and recreate them on your system during the shift operation. The default option is `true` which will result in new source and destination settings being generated.

If you are importing settings files from other systems, or if you need to rerun the shift process without regenerating the files, use the `--blank-slate=false` option. When Db2 Shift executes it will use the existing metadata in the working directory and attempt to use those settings.

Syntax: `--blank-slate=false`

Metadata : ○ Refresh ● None ○ Settings Only ○ Verify

One scenario that involves the use of `--blank-slate=false` occurs when a shift operation fails at the target OC/Kubernetes pod because of a communication error. The database at the destination will be left in an inconsistent state and must be rebuilt by the shift process. The settings for the destination database can no longer be retrieved because the database cannot be started. Because of this reason, you must use the existing destination settings that were generated when you first ran the Db2 Shift command.

Generate Settings

Syntax: `--blank-slate=true, --gen-settings`

Metadata : ○ Refresh ○ None ● Settings Only ○ Verify

The `--gen-settings` (Generate Settings) option is used in conjunction with `--blank-slate`. The use of `--gen-settings` will prevent Db2 Shift from continuing execution after the metadata files have been created.

Verification

Syntax: `--verify-only`

```
Metadata      :  Refresh  None  Settings Only  Verify
```

The `--verify-only` option will generate the metadata files and check the connectivity and all settings and then stop execution. If `--verify-only` completes successfully, the Db2 Shift command will be able to execute the shift process.

Online or Offline Move

Syntax: `--online, --offline`

```
Database Mode :  Online Move  Offline Move
```

Db2 Shift provides two options when dealing with the state of a database during the shift process:

- `--online` - database is online while the shift is taking place
- `--offline` - database has been shut down

By default, the Db2 Shift command assumes you will use online mode and will suspend the database while it completes the last scan of the files. During this period, the database will not complete any insert, update, or delete transactions. This will result in a consistent database at the destination, but some transactions will not have been committed to the database. If the database at the destination must be identical to the source database, then you must completely shut down the database and choose the `--offline` option.

During the last step of the shift process, Db2 Shift will search for any updates that may have been applied to the database after the initial copying was done. To ensure the integrity of the data being copied, the database will be placed into a WRITE SUSPEND mode. This last step should only take a few seconds and the database will be WRITE RESUMED when the final copy is done.

When the database is suspended, all read activities will continue. New connections will not be permitted, but any applications that are currently running will be allowed to continue. Those transactions that are updating records will be temporarily "paused" while the copying is done. Once the copy is complete, a suspended application will finish their transaction.

The period that the database is suspended or stopped is dependent on the changes that have occurred in the database from the time the shift operation started to the time when copying of the data is complete. The changes that have occurred during this period needs to be captured during the final step. It is during this step that the database must be suspended or stopped. During the initial scan the database will remain completely online and will not be impacted

by the shift utility from a transaction perspective. However, since there is a large amount of disk reads taking place, it may impact buffer pool read performance.

The offline mode must be used when shifting a database that requires a migration from an older release of Db2. This option must be specified when shifting Db2 versions 10.5 or 11.1.

To use offline mode, you must run Db2 Shift against the source and target using:

- Shift command with `--verify-only`
This step will generate a copy of the control files needed
- Db2 Shift operation using the offline mode `--offline`

If you do not create the control files beforehand, the Db2 Shift operation will not have the necessary control files to run the shift.

Performance: Threading

Syntax: `--threads=[0..8]`

```
Thread count      : 4
```

The copy phase of the Db2 Shift program can use multiple threads to transmit data to a destination. This setting allows you to increase the parallelism up to 8 threads. As you increase the number of threads, the amount of data being transmitted increases, at the expense of greater CPU usage and network congestion. The default value is 4 which strikes a balance between overhead and network performance.

9

Deploying a Clone to OpenShift, Kubernetes or CP4D

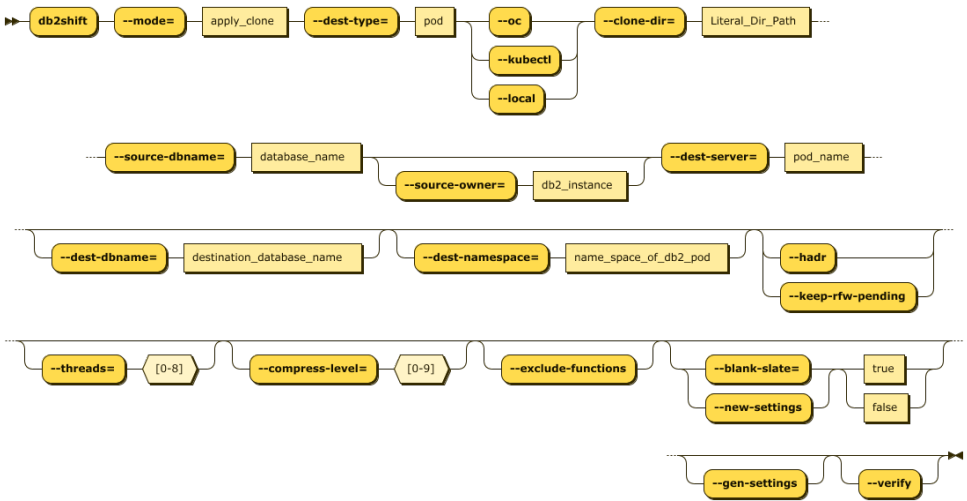
DEPLOYING A CLONE TO A POD

Chapter 9: Clone Deployment to OpenShift, Kubernetes, or CP4D

This Db2 Shift option will take a database clone and deploy it into a Db2u pod running on OpenShift, Kubernetes or CP4D.

The panel requires the following information:

- Database name and clone location
- Destination POD details
- Type of Containerization environment
- Shift Options



Required Options

- mode=apply_clone
- dest-type=POD
- oc or --kubectl or --local
- source-dbname=flights
- source-owner=db2inst1
- clone-dir=/tmp/cache
- dest-dbname=db2oltp
- dest-server=c-demo-db2u-0

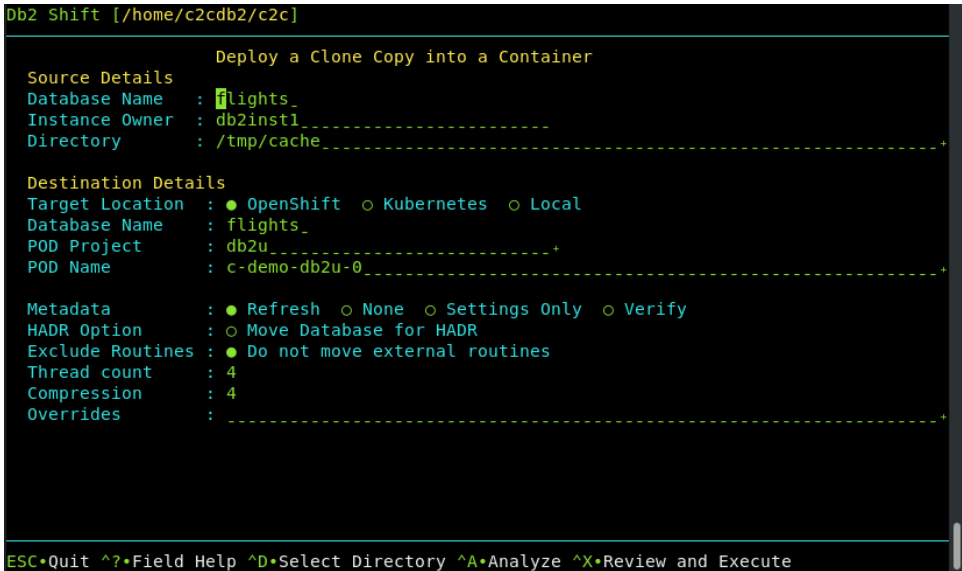
Optional Settings

- dest-namespace=db2u
- hadr or --keep-rfw-pending
- online or --offline
- threads=4

Chapter 9: Deploying a Clone to a POD

```
--compress-level=4
--exclude-functions
--sync=[start_sync, rerun_sync, finish_sync]
--blank-slate(--new-settings)=[true|false]
--gen-settings
--verify
```

The panel that provides this capability:



```
Db2 Shift [/home/c2cdb2/c2c]
Deploy a Clone Copy into a Container

Source Details
Database Name : flights_
Instance Owner : db2inst1
Directory : /tmp/cache

Destination Details
Target Location : ● OpenShift ○ Kubernetes ○ Local
Database Name : flights_
POD Project : db2u
POD Name : c-demo-db2u-0

Metadata : ● Refresh ○ None ○ Settings Only ○ Verify
HADR Option : ○ Move Database for HADR
Exclude Routines : ● Do not move external routines
Thread count : 4
Compression : 4
Overrides :

ESC•Quit ^?•Field Help ^D•Select Directory ^A•Analyze ^X•Review and Execute
```

To view this panel directly, use the syntax:

- `db2shift --deploypod`

Mode Option

Syntax: `--mode=apply_clone`

The MODE option determines what steps the Db2 Shift program will take to move your database to the new location.

The APPLY_CLONE option will take an existing CLONE database and shift it to a POD or a Db2 instance. There is no need to be connected to the original database because all the control information is contained within the CLONES image.

Target Client (Instance to POD)

Syntax: `--oc`, `--kubect1`, `--local`

```
Target Location : ● OpenShift ○ Kubernetes ○ Local
```

The pod client for a deploy (clone) operation must be supplied as part of the Db2 Shift command. Only one of the following clients must be used:

- `--oc` OpenShift Destination
- `--kubect1` Kubernetes Destination
- `--local` Copy of the cloned database is inside the pod

If the client is Kubernetes (`--kubect1`) or OpenShift (`--oc`), the program requires that the appropriate `kubect1` or `oc` client has been installed locally and that the namespace or project has already been specified.

The `--local` option is used when the cloned copy of the database has already been copied inside the POD that it is being shifted to.

Source Database

Syntax: `--source-database=""`

```
Database Name : lights_
```

The source database is the name of the database that you have in the cloned copy. The deploy option double checks that you have the proper database in the clone.

Note that you can have the same or different database name at the target. If you provide a different database name at the target, the program will copy the clone and place it on the target and use the existing name.

Alert!

If you are shifting a database to Cloud Pak for Data, then the cloned database must have `db2inst1` as the INSTANCE userid. See the section on Cloning a database to make sure that the proper authorities have been granted to the `db2inst1` userid.

Source or Instance Owner

Syntax: `--source-owner="instance name"`

```
Instance Owner : db2inst1_.....
```

The Db2 Shift checks that the cloned copy was created with the instance owner provided with this setting.

Clone Directory

Syntax: `--clone-dir=""`

```
Directory      : /tmp/cache_-----+
```

The cloned copy of the database will be retrieved from the directory that is specified by this field.

Destination Database

Syntax: `--dest-database=""`

```
Database Name  : flights_-----+
```

The destination database name can be the same as the source database, or a completely different name. Make sure that if you are creating a database with a different name that it doesn't currently exist on your target system. Otherwise, the target database will be deleted!

Destination Server (POD)

Syntax: `--dest-server="pod_name"`

```
POD Project    : db2u_-----+
```

For deployments to OpenShift, Kubernetes, or CP4D, you must supply the name of the POD that Db2U is running in. The OpenShift or Kubernetes client should be used to connect to the target namespace or project before issuing the Db2 Shift command.

Destination Pod Namespace or Project

Syntax: `--dest-namespace="", --dest-project=""`

```
POD Name      : c-demo-db2u-0_-----+
```

In Kubernetes deployments, the location of a pod is associated with a namespace, while in OpenShift deployments, the pod is associated with a project.

When authenticating to a Kubernetes or OpenShift environment, it is recommended that the local client be connected to the project or namespace that the Db2U pod is running in.

If you do not supply a namespace or project value, the Db2 Shift program will assume that you are already connected to that project. If this is not the case, the program will stop with an error when it attempts to find the pod.

To have Db2 Shift connect to the appropriate project or namespace, supply the value of the namespace or project using this option.

Stored Procedures and Functions

Syntax: `--exclude-functions`

```
Exclude Routines :  Do not move external routines
```

By default, the Db2 Shift command will automatically move all external stored procedures and functions that are found in the `$HOME/sqllib/function` path to the new destination. All SQLPL and PL/SQL routines are moved as part of the database move, so there is no migration required for those routines.

Use the exclude flag to prevent any external functions being moved to the destination.

HADR Setup

Syntax: `--hadr` or `--keep-rfw-pending`

```
Move Options :  Database  Database/LDAP/DMC  Move Database for HADR
```

When HADR mode is selected, the database will be copied over to the target location and initialized as an HADR secondary. The database can now connect to the primary database as an HADR pair and participate in failover or read-only applications.

This option cannot be used if your database needs to be migrated or if the database needs to be renamed.

Synchronization Options

Syntax: `--sync=[start_sync, rerun_sync, finish_sync]`

```
Sync Options :  None  Initialize  Refresh  Finalize
```

The Db2 Shift program has two methods of moving the database to the destination. The traditional method is to take one pass at the database and move everything at once. During the finishing process, the program will briefly suspend the database (depending on settings) and perform a final refresh. This last step will pick up any database objects that may have changed over the course of the move.

The total amount of time the database is suspended is directly related to how much information has changed during the shift process. If you have large number of changes, the final refresh may too long an outage. To minimize this outage, you can use the synchronization options provided.

The default setting is do a complete shift operation without intermediate sync operations.

To initialize the synchronization option, the first Db2 Shift command will use the sync option.

Syntax: `--sync=start_sync`

```
Sync Options      :  None  Initialize  Refresh  Finalize
```

This initial step will instruct Db2 Shift to copy all the required database objects to the target system. Once the copies are complete, the program will end processing and leave the target system in an incomplete state. During this process the source database is operational, and it will not be suspended.

When there is an appropriate moment, the database movement can be finalized with the `finish_sync` option.

Syntax: `--sync=finish_sync`

```
Sync Options      :  None  Initialize  Refresh  Finalize
```

The `finish_sync` option will do one final pass against the source database and then it will finalize the database movement on the destination site. During the last pass the finish process will suspend the database to get a consistent database environment. Once this step completes, the destination database will be available.

If the source system has high update volumes, there may be a need to do multiple sync operations to minimize the finalize step. The Db2 Shift command will need to be told that it is syncing the database again, but not to start from scratch. The control files generated by the `--sync=start_sync` option will allow Db2 Shift to move database objects that are new or have changed since the initial synchronization request. The command to do the sync and only refresh the database objects requires the use of the `rerun_sync` option.

Syntax: `--sync=rerun_sync`

```
Sync Options      :  None  Initialize  Refresh  Finalize
```

The `rerun_sync` option indicates to the Db2 Shift program to start syncing process to look for delta changes only.

The `--sync=rerun_sync` process can be run repeatedly until the number of changes between runs is minimized. When an appropriate timeslot is available, the shift can be finalized by using the `finish_sync` option.

Syntax: `--sync=finish_sync`

Sync Options : None Initialize Refresh Finalize

The `finish_sync` option will perform the final pass against the source database and complete the shift process as before.

In summary, the standard shift operation will complete in one step (no `--sync` option is used). The `--sync=start_sync` option allows you to gradually move a database over time. For a gradual database shift, use the `start_sync` option on the first run. This will move an initial set of database files to the target. Then use the `rerun_sync` option on subsequent runs to copy any files that may have been added or changed to the database. When you are ready to finalize the shift, use the `finish_sync` option to gather any remaining files and complete the shift operation on the target system.

Metadata Generation

Syntax: `--blank-slate=[true|false]`, `--gen-settings`, `--verify-only`

Metadata : Refresh None Settings Only Verify

The Db2 Shift command generates metadata that is used during the shift process. This metadata is key to determining which objects need to be moved from a source to destination as well as validating that the source can be successfully moved.

Generating metadata requires access to the source and destination systems. If for some reason the connection to the source or destination is unavailable, the existing metadata files can still be used. In most cases you will not need to adjust these settings unless you have encountered a shift error.

Blank Slate

Syntax: `--blank-slate=true`

Metadata : Refresh None Settings Only Verify

The `--blank-slate` option determines whether the existing metadata is refreshed. The default value for `--blank-slate` is `true` which will display in the GUI as Refresh. The `--blank-slate` option will delete any existing metadata files and recreate them on your system during the shift operation. The default option is `true` which will result in new source and destination settings being generated.

If you are importing settings files from other systems, or if you need to rerun the shift process without regenerating the files, use the `--blank-slate=false`

option. When Db2 Shift executes it will use the existing metadata in the working directory and attempt to use those settings.

Syntax: `--blank-slate=false`

```
Metadata      :  Refresh  None  Settings Only  Verify
```

One scenario that involves the use of `--blank-slate=false` occurs when a shift operation fails at the target OC/Kubernetes pod because of a communication error. The database at the destination will be left in an inconsistent state and must be rebuilt by the shift process. The settings for the destination database can no longer be retrieved because the database cannot be started. Because of this reason, you must use the existing destination settings that were generated when you first ran the Db2 Shift command.

Generate Settings

Syntax: `--blank-slate=true, --gen-settings`

```
Metadata      :  Refresh  None  Settings Only  Verify
```

The `--gen-settings` (Generate Settings) option is used in conjunction with `--blank-slate`. The use of `--gen-settings` will prevent Db2 Shift from continuing execution after the metadata files have been created.

Verification

Syntax: `--verify-only`

```
Metadata      :  Refresh  None  Settings Only  Verify
```

The `--verify-only` option will generate the metadata files and check the connectivity and all settings and then stop execution. If `--verify-only` completes successfully, the Db2 Shift command will be able to execute the shift process.

Online or Offline Move

Syntax: `--online, --offline`

```
Database Mode :  Online Move  Offline Move
```

Db2 Shift provides two options when dealing with the state of a database during the shift process:

- `--online` - database is online while the shift is taking place
- `--offline` - database has been shut down

By default, the Db2 Shift command assumes you will use online mode and will suspend the database while it completes the last scan of the files. During this period, the database will not complete any insert, update, or delete transactions. This will result in a consistent database at the destination, but some transactions will not have been committed to the database. If the database at the destination must be identical to the source database, then you must completely shut down the database and choose the `--offline` option.

During the last step of the shift process, Db2 Shift will search for any updates that may have been applied to the database after the initial copying was done. To ensure the integrity of the data being copied, the database will be placed into a WRITE SUSPEND mode. This last step should only take a few seconds and the database will be WRITE RESUMED when the final copy is done.

When the database is suspended, all read activities will continue. New connections will not be permitted, but any applications that are currently running will be allowed to continue. Those transactions that are updating records will be temporarily "paused" while the copying is done. Once the copy is complete, a suspended application will finish their transaction.

The period that the database is suspended or stopped is dependent on the changes that have occurred in the database from the time the shift operation started to the time when copying of the data is complete. The changes that have occurred during this period needs to be captured during the final step. It is during this step that the database must be suspended or stopped. During the initial scan the database will remain completely online and will not be impacted by the shift utility from a transaction perspective. However, since there is a large amount of disk reads taking place, it may impact buffer pool read performance.

The offline mode must be used when shifting a database that requires a migration from an older release of Db2. This option must be specified when shifting Db2 versions 10.5 or 11.1.

To use offline mode, you must run Db2 Shift against the source and target using:

- Shift command with `--verify-only`
This step will generate a copy of the control files needed
- Db2 Shift operation using the offline mode `--offline`

If you do not create the control files beforehand, the Db2 Shift operation will not have the necessary control files to run the shift.

Performance: Threading

Syntax: `--threads=[0..8]`

```
Thread count      : 4
```

The copy phase of the Db2 Shift program can use multiple threads to transmit data to a destination. This setting allows you to increase the parallelism up to 8 threads. As you increase the number of threads, the amount of data being transmitted increases, at the expense of greater CPU usage and network congestion. The default value is 4 which strikes a balance between overhead and network performance.

Performance: Compression

Syntax: `--compression=[0..9]`

```
Compression      : 4
```

RSYNC compresses the data during the transfer process to allow for faster movement of data. The amount of compression can be adjusted from 0 to 9 with 0 turning off compression and values between 1 and 9 increasing the amount of compression applied to the data. Higher compression values will result in more CPU usage and may not significantly reduce the size of the data stream.

A value of 4 has been found to be a good compromise between compression overhead and data size on slow networks (< 1Gb/s). For high-speed networks, a value of 0 is recommended unless there is a requirement to reduce network traffic.

10

Deploying a Clone to another Db2 Instance

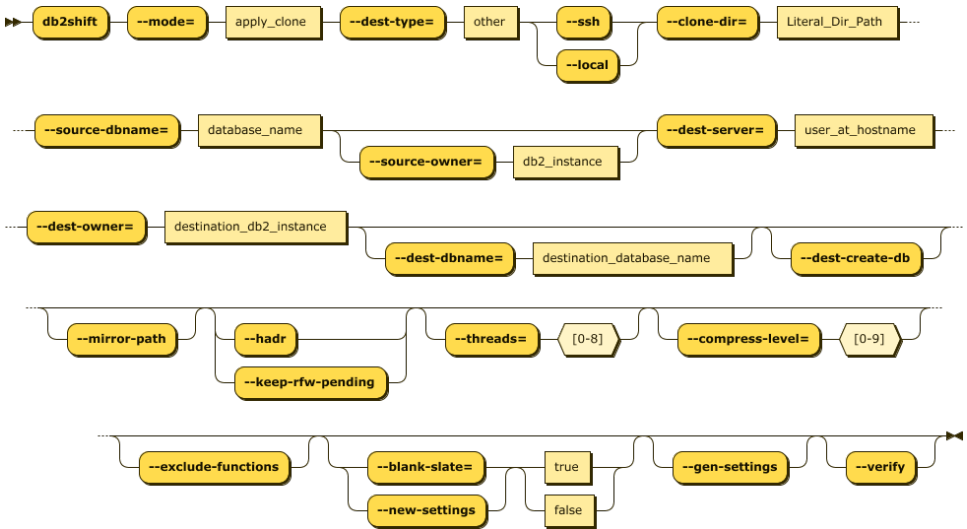
DEPLOY INSTANCE TO INSTANCE

Chapter 10: Clone Deployment to Another Db2 Instance

This Db2 Shift option will take a database clone and deploy it into another Db2 instance running natively (in any environment including Cloud VMs).

The panel requires the following information:

- Database name and clone location
- Destination instance details
- Shift Options



Required Options

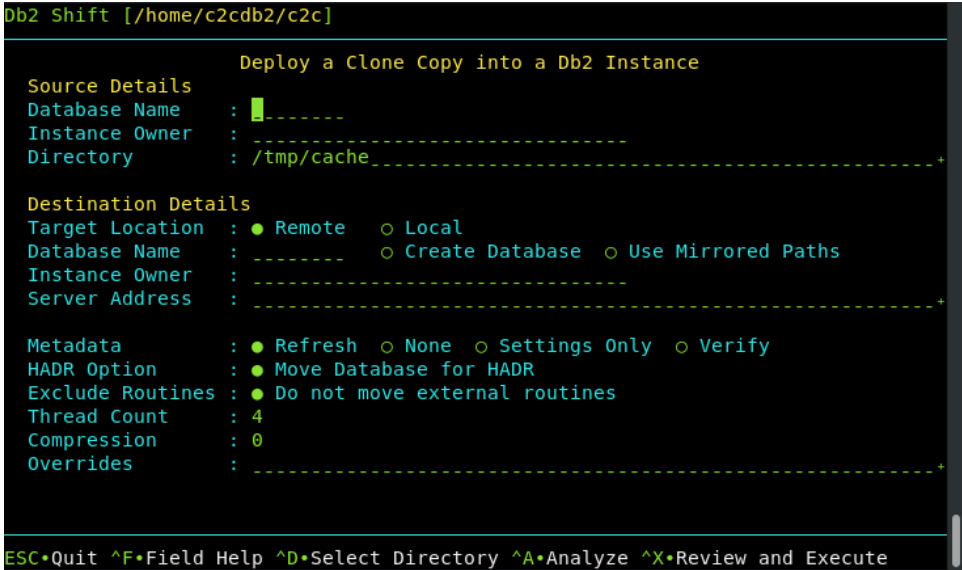
```
--mode=apply_clone
--dest-type=OTHER
--ssh or --local
--source-dbname=flights
--source-owner=db2inst1
--clone-dir=/tmp/cache
--dest-dbname=db2oltp
--dest-owner=db2inst1
--dest-server=db2inst1@some.server.com
```

Optional Settings

```
--dest-create-db
--hadr or --keep-rfw-pending
--online or --offline
```

```
--threads=4
--compress-level=4
--exclude-functions
--sync=[start_sync, rerun_sync, finish_sync]
--blank-slate(--new-settings)=[true|false]
--gen-settings
--verify
```

The panel that provides this capability:



To view this panel directly, use the syntax:

- `db2shift --deploydb2`

Mode Option

Syntax: `--mode=apply_clone`

The `MODE` option determines what steps the Db2 Shift program will take to move your database to the new location.

The `APPLY_CLONE` option will take an existing `CLONE` database and shift it to a `POD` or a `Db2` instance. There is no need to be connected to the original database because all the control information is contained within the `CLONES` image.

Target Client (Instance to Instance)

Syntax: `--ssh`, `--local`

```
Target Location : ● Remote ○ Local
```

The client for a deploy (clone) operation must be supplied as part of the Db2 Shift command. Only one of the following targets must be used:

- `--ssh` Remote Db2 Instance
- `--local` Db2 Instance

If the target is a remote Db2 instance (`--ssh`), Db2 Shift expects that a passwordless ssh environment has been established between the source and target servers.

The `--local` option only applies to Clones (database copies) that are being deployed onto the same server that the Db2 Shift command is running on. The ability to shift a database locally (i.e., make a copy of a Db2 database on the same instance) is currently disabled due to some restrictions with `db2relocatdb`.

Source Database

Syntax: `--source-database=""`

```
Database Name : lights_
```

The source database is the name of the database that you have in the cloned copy. The deploy option double checks that you have the proper database in the clone.

Note that you can have the same or different database name at the target. If you provide a different database name at the target, the program will copy the clone and place it on the target and use the existing name.

Source or Instance Owner

Syntax: `--source-owner="instance name"`

```
Instance Owner : db2inst1_.....
```

The Db2 Shift checks that the cloned copy was created with the instance owner provided with this setting.

Clone Directory

Syntax: `--clone-dir=""`

```
Directory      : /tmp/cache_-----
```

The cloned copy of the database will be retrieved from the directory that is specified by this field.

Destination Database

Syntax: `--dest-database=""`

```
Database Name  : flights_
```

The destination database name can be the same as the source database, or a completely different name. Make sure that if you are creating a database with a different name that it doesn't currently exist on your target system. Otherwise, the target database will be deleted!

Force Destination Database Creation

Syntax: `--dest-create-db`

```
Database Name  : flights_ ● Force Database Creation
```

During a shift or clone operation to a traditional Db2 INSTANCE, the Db2 Shift program will check whether the target database exists. If it does not, the Db2 Shift operation will fail. If the Force Database Creation setting is on, Db2 Shift will attempt to create the database first and then run the shift operation.

Note: Db2 Shift will create the database in a directory structure under `/db2_portable` on the target system.

Mirror Path

Syntax: `--mirror-path`

```
Database Name  : ----- ○ Create Database ○ Use Mirrored Paths
```

Mirror path is a valid switch for Applying a Clone or moving a database to a standard db2 instance.

It will take all the paths, except the default database path, and use them as the destination paths on the target instance. It assumes that those paths exist already, or it can create them. The database names and instance owners must also be the same.

If used in combination with `--dest-create-db`, the database that is created, rather than being created under the `/home/$DB2INST/db2_portable` directory, will be created on the same path as the source system. If using this option into a path (even if it is the default database path) you may receive an error if the database configuration directory number e.g., `SQL00001` does not match across source and target systems. In this case you should manually create the target database on a different path. This is a limitation of the Db2 relocation utility.

When using this option, you must be cautious that the target paths on the destination are on paths that the instance owner has permissions to create the underlying directory structures, and that you have free space to create the database.

Destination Server (Instance)

Syntax: `--dest-server=userid@ip.address`

```
Server Address : some.server.com_.....
```

For destinations that are traditional Db2 instances, you must provide the `userid` and the IP address or symbolic name of the destination server. You only use an IP address for shifting into a traditional Db2 instance.

The format of the parameter is `userid@address` when using the Db2 Shift command line.

When connecting to a remote instance, the Db2 Shift program expects that a passwordless ssh environment has been established between the source and target servers. The `--ssh` flag must be used in conjunction with this parameter.

Stored Procedures and Functions

Syntax: `--exclude-functions`

```
Exclude Routines : o Do not move external routines
```

By default, the Db2 Shift command will automatically move all external stored procedures and functions that are found in the `$HOME/sqlllib/function` path to the new destination. All SQLPL and PL/SQL routines are moved as part of the database move, so there is no migration required for those routines.

Use the `exclude` flag to prevent any external functions being moved to the destination.

HADR Setup

Syntax: `--hadr` or `--keep-rfw-pending`

HADR Option : ● Move Database for HADR

When HADR mode is selected, the database will be copied over to the target location and initialized as an HADR secondary. The database can now connect to the primary database as an HADR pair and participate in failover or read-only applications.

This option cannot be used if your database needs to be migrated or if the database needs to be renamed.

Synchronization Options

Syntax: `--sync=[start_sync, rerun_sync, finish_sync]`

Sync Options : ● None ○ Initialize ○ Refresh ○ Finalize

The Db2 Shift program has two methods of moving the database to the destination. The traditional method is to take one pass at the database and move everything at once. During the finishing process, the program will briefly suspend the database (depending on settings) and perform a final refresh. This last step will pick up any database objects that may have changed over the course of the move.

The total amount of time the database is suspended is directly related to how much information has changed during the shift process. If you have large number of changes, the final refresh may too long an outage. To minimize this outage, you can use the synchronization options provided.

The default setting is do a complete shift operation without intermediate sync operations. To initialize the synchronization option, the first Db2 Shift command will use the `sync` option.

Syntax: `--sync=start_sync`

Sync Options : ○ None ● Initialize ○ Refresh ○ Finalize

This initial step will instruct Db2 Shift to copy all the required database objects to the target system. Once the copies are complete, the program will end processing and leave the target system in an incomplete state. During this process the source database is operational, and it will **not be suspended**.

When there is an appropriate moment, the database movement can be finalized with the `finish_sync` option.

Syntax: --sync=finish_sync

```
Sync Options      :  None  Initialize  Refresh  Finalize
```

The `finish_sync` option will do one final pass against the source database and then it will finalize the database movement on the destination site. During the last pass the finish process will suspend the database to get a consistent database environment. Once this step completes, the destination database will be available.

If the source system has high update volumes, there may be a need to do multiple sync operations to minimize the finalize step. The Db2 Shift command will need to be told that it is syncing the database again, but not to start from scratch. The control files generated by the `--sync=start_sync` option will allow Db2 Shift to move database objects that are new or have changed since the initial synchronization request. The command to do the sync and only refresh the database objects requires the use of the `rerun_sync` option.

Syntax: --sync=rerun_sync

```
Sync Options      :  None  Initialize  Refresh  Finalize
```

The `rerun_sync` option indicates to the Db2 Shift program to start syncing process to look for delta changes only.

The `--sync=rerun_sync` process can be run repeatedly until the number of changes between runs is minimized. When an appropriate timeslot is available, the shift can be finalized by using the `finish_sync` option.

Syntax: --sync=finish_sync

```
Sync Options      :  None  Initialize  Refresh  Finalize
```

The `finish_sync` option will perform the final pass against the source database and complete the shift process as before.

In summary, the standard shift operation will complete in one step (no `--sync` option is used). The `--sync=start_sync` option allows you to gradually move a database over time. For a gradual database shift, use the `start_sync` option on the first run. This will move an initial set of database files to the target. Then use the `rerun_sync` option on subsequent runs to copy any files that may have been added or changed to the database. When you are ready to finalize the shift, use the `finish_sync` option to gather any remaining files and complete the shift operation on the target system.

Metadata Generation

Syntax: `--blank-slate=[true|false], --gen-settings, --verify-only`

Metadata : Refresh None Settings Only Verify

The Db2 Shift command generates metadata that is used during the shift process. This metadata is key to determining which objects need to be moved from a source to destination as well as validating that the source can be successfully moved.

Generating metadata requires access to the source and destination systems. If for some reason the connection to the source or destination is unavailable, the existing metadata files can still be used. In most cases you will not need to adjust these settings unless you have encountered a shift error.

Blank Slate

Syntax: `--blank-slate=true`

Metadata : Refresh None Settings Only Verify

The `--blank-slate` option determines whether the existing metadata is refreshed. The default value for `--blank-slate` is `true` which will display in the GUI as Refresh. The `--blank-slate` option will delete any existing metadata files and recreate them on your system during the shift operation. The default option is `true` which will result in new source and destination settings being generated.

If you are importing settings files from other systems, or if you need to rerun the shift process without regenerating the files, use the `--blank-slate=false` option. When Db2 Shift executes it will use the existing metadata in the working directory and attempt to use those settings.

Syntax: `--blank-slate=false`

Metadata : Refresh None Settings Only Verify

One scenario that involves the use of `--blank-slate=false` occurs when a shift operation fails at the target OC/Kubernetes pod because of a communication error. The database at the destination will be left in an inconsistent state and must be rebuilt by the shift process. The settings for the destination database can no longer be retrieved because the database cannot be started. Because of this reason, you must use the existing destination settings that were generated when you first ran the Db2 Shift command.

Generate Settings

Syntax: `--blank-slate=true, --gen-settings`

```
Metadata :  Refresh  None  Settings Only  Verify
```

The `--gen-settings` (Generate Settings) option is used in conjunction with `--blank-slate`. The use of `--gen-settings` will prevent Db2 Shift from continuing execution after the metadata files have been created.

Verification

Syntax: `--verify-only`

```
Metadata :  Refresh  None  Settings Only  Verify
```

The `--verify-only` option will generate the metadata files and check the connectivity and all settings and then stop execution. If `--verify-only` completes successfully, the Db2 Shift command will be able to execute the shift process.

Online or Offline Move

Syntax: `--online, --offline`

```
Database Mode :  Online Move  Offline Move
```

Db2 Shift provides two options when dealing with the state of a database during the shift process:

- `--online` - database is online while the shift is taking place
- `--offline` - database has been shut down

By default, the Db2 Shift command assumes you will use online mode and will suspend the database while it completes the last scan of the files. During this period, the database will not complete any insert, update, or delete transactions. This will result in a consistent database at the destination, but some transactions will not have been committed to the database. If the database at the destination must be identical to the source database, then you must completely shut down the database and choose the `--offline` option.

During the last step of the shift process, Db2 Shift will search for any updates that may have been applied to the database after the initial copying was done. To ensure the integrity of the data being copied, the database will be placed into a WRITE SUSPEND mode. This last step should only take a few seconds and the database will be WRITE RESUMED when the final copy is done.

When the database is suspended, all read activities will continue. New connections will not be permitted, but any applications that are currently

running will be allowed to continue. Those transactions that are updating records will be temporarily "paused" while the copying is done. Once the copy is complete, a suspended application will finish their transaction.

The period that the database is suspended or stopped is dependent on the changes that have occurred in the database from the time the shift operation started to the time when copying of the data is complete. The changes that have occurred during this period needs to be captured during the final step. It is during this step that the database must be suspended or stopped. During the initial scan the database will remain completely online and will not be impacted by the shift utility from a transaction perspective. However, since there is a large amount of disk reads taking place, it may impact buffer pool read performance.

The offline mode must be used when shifting a database that requires a migration from an older release of Db2. This option must be specified when shifting Db2 versions 10.5 or 11.1.

To use offline mode, you must run Db2 Shift against the source and target using:

- Shift command with `--verify-only`
This step will generate a copy of the control files needed
- Db2 Shift operation using the offline mode `--offline`

If you do not create the control files beforehand, the Db2 Shift operation will not have the necessary control files to run the shift.

Performance: Threading

Syntax: `--threads=[0...8]`

```
Thread count      : 4
```

The copy phase of the Db2 Shift program can use multiple threads to transmit data to a destination. This setting allows you to increase the parallelism up to 8 threads. As you increase the number of threads, the amount of data being transmitted increases, at the expense of greater CPU usage and network congestion. The default value is 4 which strikes a balance between overhead and network performance.

Performance: Compression

Syntax: `--compression=[0...9]`

```
Compression      : 4
```

RSYNC compresses the data during the transfer process to allow for faster movement of data. The amount of compression can be adjusted from 0 to 9 with

0 turning off compression and values between 1 and 9 increasing the amount of compression applied to the data. Higher compression values will result in more CPU usage and may not significantly reduce the size of the data stream.

A value of 4 has been found to be a good compromise between compression overhead and data size on slow networks (< 1Gb/s). For high-speed networks, a value of 0 is recommended unless there is a requirement to reduce network traffic.

11

HADR Initialization

SOURCE AND TARGET POD

Chapter 11: Initialize HADR between Source and Target POD

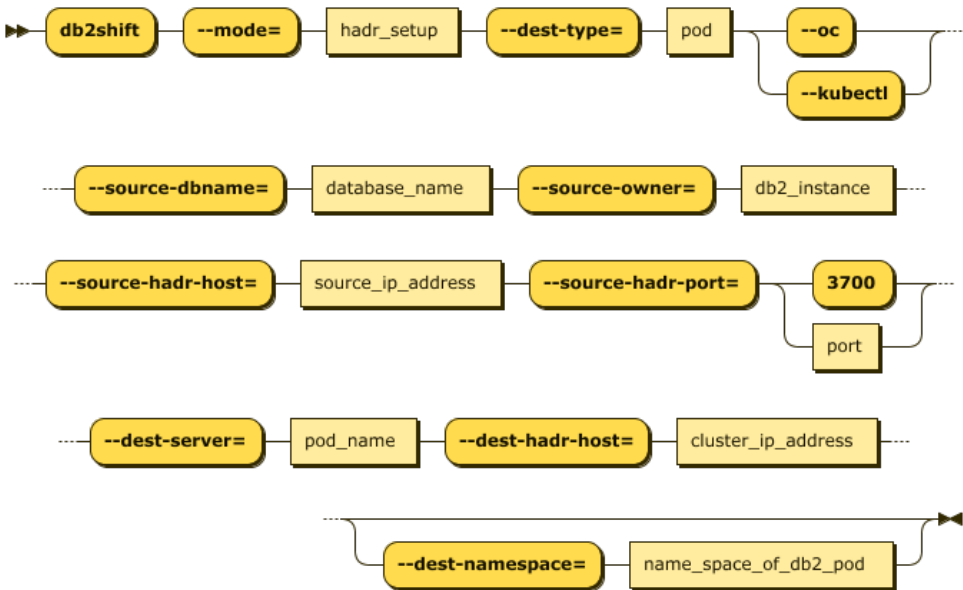
This Db2 Shift option will take a source and destination (POD) database and start the HADR service between them. The Db2u pod must have been created with the following setting during the shift step.

Syntax: `--hadr` or `--keep-rlw-pending`

Move Options : Database Database/LDAP/DMC Move Database for HADR

The panel requires the following information:

- The source database name and server
- The destination POD and server details



Required Options

- `--mode=hadr_setup`
- `--dest-type=POD`
- `--oc` or `--kubectl`
- `--source-dbname=flights`
- `--source-hadr-host=some.server.com`
- `--source-hadr-port=3700`
- `--dest-server=c-demo-db2u-0`
- `--dest-hadr-host=oc.server.com`

Optional Settings

```
--dest-namespace=db2u
```

The panel that provides this capability:

```
Db2 Shift [/home/c2cdb2/c2c/dist]

                Initialize HADR Source and Destination Databases (POD)

Source Details
Database Name   : █-----
Source HADR IP  : -----+
Source HADR Port : 3700_

Destination Details
Target Location : ● OpenShift ○ Kubernetes
POD Project     : -----+
POD Name        : -----+
Target HADR IP  : -----+
```

ESC•Quit ^F•Field Help ^X•Review and Execute

To view this panel directly, use the syntax:

- `db2shift --hadrxpod`

Mode Option

Syntax: `--mode=hadr_setup`

The HADR option is used to initialize and start HADR between a source and target server. This step would be run after the database has been shifted to the new location. This option is not used for the initial HADR setup of the target system. The target database must be created with the `--hadrx` option for it be placed into the correct mode for HADR communication. The `--hadrx` is enabled for pods with the Move Database for HADR option in the menu.

```
Move Options      : ○ Database ○ Database/LDAP/DMC ● Move Database for HADR
```

Target Client (Instance to POD)

Syntax: `--oc`, `--kubect1`

```
Target Location : ● OpenShift ○ Kubernetes
```

The pod client for a deploy (clone) operation must be supplied as part of the Db2 Shift command. Only one of the following clients must be used:

- `--oc` OpenShift Destination
- `--kubect1` Kubernetes Destination

If the client is Kubernetes (`--kubect1`) or OpenShift (`--oc`), the program requires that the appropriate `kubect1` or `oc` client has been installed locally and that the namespace or project has already been specified.

Source Database

Syntax: `--source-database=""`

```
Database Name : lights_
```

The source database is the name of the database that you want to setup HADR with on the source and destination servers. Note that the database name must be same at both locations.

Alert!

If you are shifting a database to Cloud Pak for Data, then the cloned database must have `db2inst1` as the INSTANCE userid. See the section on Cloning a database to make sure that the proper authorities have been granted to the `db2inst1` userid.

HADR Source and Destination Server

Syntax: `--source-hadr-server=""`, `--dest-hadr-server=""`

```
Source HADR IP : some.server.com.....*
```

```
Target HADR IP : other.server.com.....*
```

For HADR setup, the Db2 Shift command requires the IP or symbolic name of the source server that will be used in an HADR setup. This server is referred to as the primary server.

If the destination server is a standard Db2 instance, use the IP address of the server. If the destination target is an OpenShift or Kubernetes cluster, use the address of the load balancer.

HADR ports

Syntax: `--source-hadr-port=#, --dest-hadr-port=#`

```
Source HADR Port : 3700_
```

HADR communicates over a port which is different than the Db2 instance. You must supply the source and destination port numbers that Db2 will communicate between the HADR servers. The default port number is 3700 for HADR communications but verify the value. The target port number is automatically generated for you.

Destination Server (POD)

Syntax: `--dest-server="pod_name"`

```
POD Name : c-demo-db2u-0_-----+
```

For deployments to OpenShift, Kubernetes, or CP4D, you must supply the name of the POD that Db2U is running in. The OpenShift or Kubernetes client should be used to connect to the target namespace or project before issuing the Db2 Shift command.

Destination Pod Namespace or Project

Syntax: `--dest-namespace="", --dest-project=""`

```
POD Project : db2u_-----
```

In Kubernetes deployments, the location of a pod is associated with a namespace, while in OpenShift deployments, the pod is associated with a project.

When authenticating to a Kubernetes or OpenShift environment, it is recommended that the local client be connected to the project or namespace that the Db2U pod is running in.

If you do not supply a namespace or project value, the Db2 Shift program will assume that you are already connected to that project. If this is not the case, the program will stop with an error when it attempts to find the pod.

To have Db2 Shift connect to the appropriate project or namespace, supply the value of the namespace or project using this option.

12

HADR Initialization

DB2 TO DB2 INSTANCES

Chapter 12: Initialize HADR between Source and Target Instance

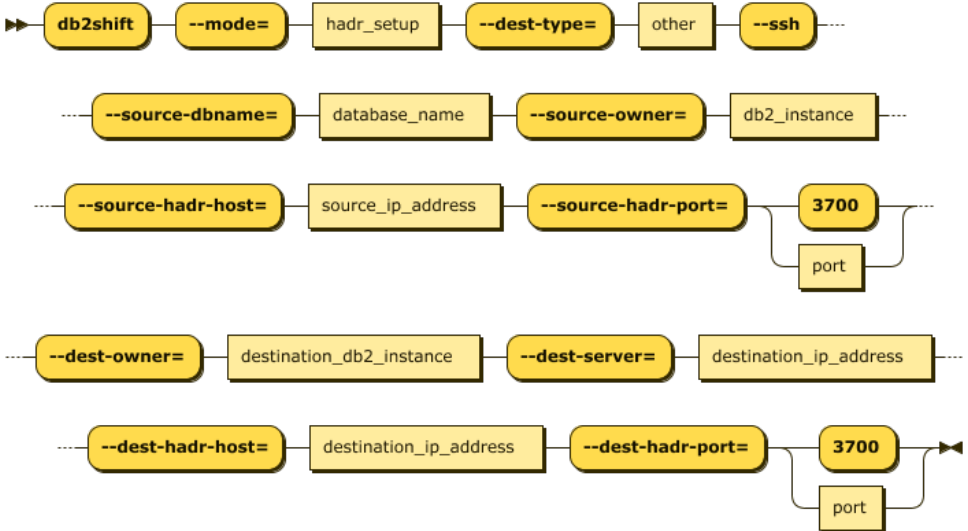
This menu is like the previous one where the HADR service is setup between the source Db2 database and another Db2 instance. The Db2 database on the target system must have been created with one of the following settings during the shift step.

Syntax: `--hadr` or `--keep-rlw-pending`

HADR Option : ● Move Database for HADR

The panel requires the following information:

- The source database name and server
- The destination server details



Required Options

```
--mode=hadr_setup
--dest-type=OTHER
--ssh
--source-dbname=flights
--source-hadr-host=some.server.com
--source-hadr-port=3700
--dest-server=db2inst1@other.server.com
--dest-hadr-host=oc.server.com
--dest-hadr-port=3700
```


The panel that provides this capability:

```
Db2 Shift [/home/c2cdb2/c2c]
Initialize HADR Source and Destination Databases
Source Details
Database Name      : flights_
Source HADR IP    : some.server.com_-----+
Source HADR Port  : 3700_
Destination Details
Instance Owner   : db2inst1_-----
Target HADR IP   : other.server.com_-----+
Target HADR Port : 3700_
ESC+Quit ^?+Field Help ^X+Review and Execute
```

To view this panel directly, use the syntax:

- `db2shift --hadrd2`

Mode Option

Syntax: `--mode=hadr_setup`

The HADR option is used to initialize and start HADR between a source and target server. This step would be run after the database has been shifted to the new location. This option is not used for the initial HADR setup of the target system. The target database must be created with the `--hadri` option for it be placed into the correct mode for HADR communication. The `--hadri` is enabled for pods with the HADR option in the menu.

```
HADR Option      : ● Move Database for HADR
```

Target Client (Instance to Instance)

Syntax: `--ssh`

The client for a deploy (clone) operation must be supplied as part of the Db2 Shift command. If the target is a remote Db2 instance (`--ssh`), Db2 Shift expects that a passwordless ssh environment has been established between the source and target servers.

Source Database

Syntax: `--source-database=""`

```
Database Name : lights_
```

The source database is the name of the database that you want to setup HADR with on the source and destination servers. Note that the database name must be same at both locations.

Destination Server (Instance)

Syntax: `--dest-server=userid@ip.address`

```
Instance Owner : db2inst1_
```

```
Target HADR IP : other.server.com
```

For destinations that are traditional Db2 instances, you must provide the userid and the IP address or symbolic name of the destination server. You only use an IP address for shifting into a traditional Db2 instance.

The format of the parameter is `userid@address` when using the Db2 Shift command line. When using the UI, this field is generated automatically by combining the destination instance owner name with the destination server address.

When connecting to a remote instance, the Db2 Shift program expects that a passwordless ssh environment has been established between the source and target servers. The `--ssh` flag must be used in conjunction with this parameter.

HADR Source and Destination Server

Syntax: `--source-hadr-server=""`, `--dest-hadr-server=""`

```
Source HADR IP : some.server.com
```

```
Target HADR IP : other.server.com
```

For HADR setup, the Db2 Shift command requires the IP or symbolic name of the source and destination server that will be used in an HADR setup.

HADR ports

Syntax: `--source-hadr-port=#, --dest-hadr-port=#`

```
Source HADR Port : 3700_
```

```
Target HADR Port : 3700_
```

HADR communicates over a port which is different than the Db2 instance. You must supply the source and destination port numbers that Db2 will communicate between the HADR servers. The default port number is 3700 for HADR communications but verify the value. The target port number will also be required.

13

LDAP and DMC Initialization

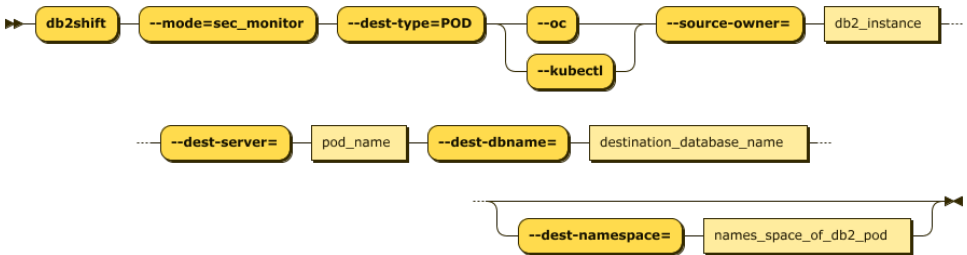
UPDATING DMC IN CP4D
ENVIRONMENTS

Chapter 13: Initialize DMC and LDAP Authentication for CP4D

This Db2 Shift option only applies when the destination is a POD on K8s or OpenShift.

When shifting a database to a CP4D platform or LDAP-secured environment, the Db2 Shift program will automatically add the appropriate userids to the LDAP service and reset the Data Management Console (DMC) so that it recognizes the database.

If you choose to shift a database to CP4D, and set it up as an HADR secondary, the LDAP and DMC setup cannot be performed until after you have converted the secondary POD into the primary database. Only when the POD is the primary database can it be initialized to support LDAP and DMC services on IBM Cloud Pak for Data.



Required Options

```
--mode=sec_and_monitor
--dest-type=POD
--oc or --kubectl
--source-owner=db2inst1
--dest-dbname=flights
--dest-server=c-demo-db2u-0
```

Optional Settings

```
--dest-namespace=db2u
```

This Db2 Shift option only applies when the destination is a POD on K8s or OpenShift. If an associated LDAP repository exists, the `db2inst1` user is added. If the IBM Data management console is deployed with a Cloud Pak for Data system, it resets the console and applies the relevant DDL and privileges for monitors.

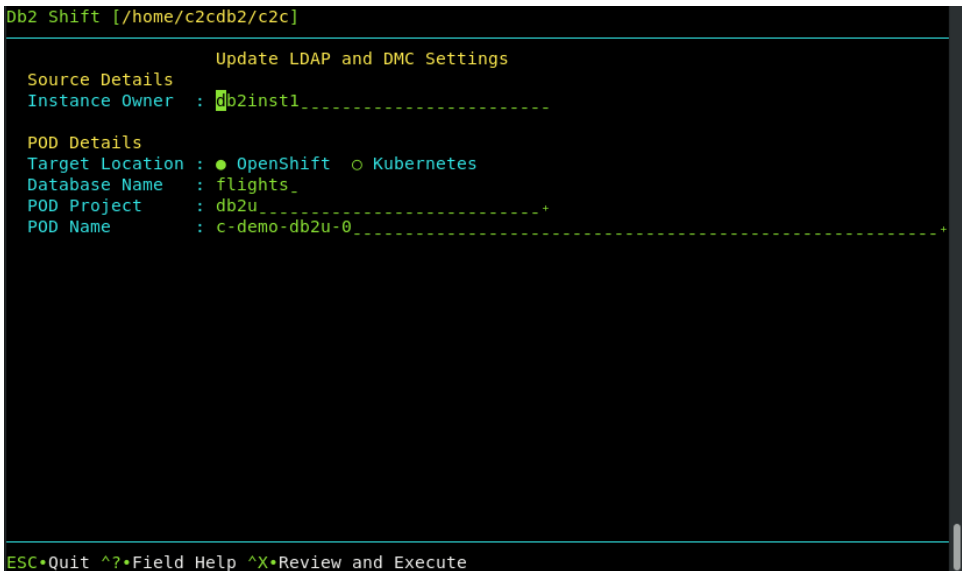
Alert!

If you are creating a database on Cloud Pak for Data, and your source INSTANCE userid is **not** db2inst1, you must execute the following SQL commands from a userid that has SECADM authority:

```
GRANT SECADM ON DATABASE TO USER db2inst1
GRANT DBADM ON DATABASE TO USER db2inst1
```

The db2inst1 userid does not need to exist in the Operating system to grant these privileges to the userid. This requirement does not apply to other Db2 Shift environments. If db2inst1 is not defined as a SECADM and DBADM user in the database, the Data Management Console feature of CP4D will not be able to access the database nor will it be able to monitor it.

The panel that provides this capability:



```
Db2 Shift [/home/c2cdb2/c2c]
Update LDAP and DMC Settings
Source Details
Instance Owner : db2inst1_-----
POD Details
Target Location : ● OpenShift ○ Kubernetes
Database Name  : flights_
POD Project    : db2u_-----+
POD Name       : c-demo-db2u-0_-----+
ESC•Quit ^?•Field Help ^X•Review and Execute
```

Mode Option

Syntax: `--mode=sec_and_monitor`

This setting only applies when the destination is a POD on K8s or OpenShift. If an associated LDAP repository exists, the db2inst1 user is added. If the IBM Data management console is deployed with a Cloud Pak for Data system, it resets the console and applies the relevant DDL and privileges for monitors.

Target Client

Syntax: `--oc, --kubect1`

```
Target Location : ● OpenShift ○ Kubernetes
```

The pod client for the security and monitoring update operation must be supplied as part of the Db2 Shift command. Only one of the following clients must be used:

- `--oc` OpenShift Destination
- `--kubect1` Kubernetes Destination

If the client is Kubernetes (`--kubect1`) or OpenShift (`--oc`), the program requires that the appropriate `kubect1` or `oc` client has been installed locally and that the namespace or project has already been specified.

Source or Instance Owner

Syntax: `--source-owner="instance name"`

```
Instance Owner : db2inst1_-----
```

The Db2 Shift program will double check that the instance owner matches that of the target system when applying the LDAP and DMC settings.

Destination Database

Syntax: `--dest-database=""`

```
Database Name : Lights_
```

The destination database must be supplied so that the updates to the security settings and Data Management Console control tables can be properly inserted.

Destination Server (POD)

Syntax: `--dest-server="pod_name"`

```
POD Name : c-demo-db2u-0_-----*
```

For deployments to OpenShift, Kubernetes, or CP4D, you must supply the name of the POD that Db2U is running in. The OpenShift or Kubernetes client should be used to connect to the target namespace or project before issuing the Db2 Shift command.

Destination Pod Namespace or Project

Syntax: `--dest-namespace=""`, `--dest-project=""`

```
POD Project      : db2u_-----
```

In Kubernetes deployments, the location of a pod is associated with a namespace, while in OpenShift deployments, the pod is associated with a project.

When authenticating to a Kubernetes or OpenShift environment, it is recommended that the local client be connected to the project or namespace that the Db2U pod is running in.

If you do not supply a namespace or project value, the Db2 Shift program will assume that you are already connected to that project. If this is not the case, the program will stop with an error when it attempts to find the pod.

To have Db2 Shift connect to the appropriate project or namespace, supply the value of the namespace or project using this option.

14

Clone Copy

COPYING CLONE IMAGES TO PODS

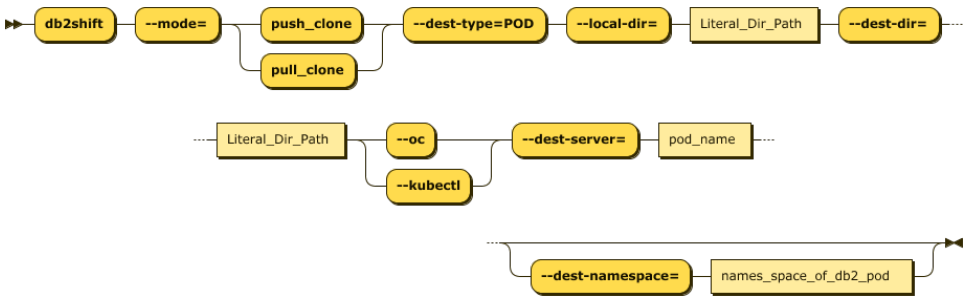
Chapter 14: Clone Copy

This Db2 Shift command provides a feature that allows a user to copy an existing database clone copy to a POD, or to retrieve a database clone from a POD.

Once a database clone has been generated, the copy can be moved to any location and then deployed locally. This option provides a convenient way of copying the database using Db2 Shift without having to use OpenShift or Kubernetes commands.

The panel requires the following information:

- Type of copy (From Source to Target or Target to Source)
- Source cloned database directory
- Target cloned database directory
- The destination POD and server details



Required Options

```

--mode=push_clone (Source to Target)
--mode=pull_clone (Target to Source)
--dest-type=POD
--oc or --kubect1
--dest-server=c-demo-db2u-0
--local-dir=/tmp/cache
--dest-dir=/tmp/cache
  
```

Optional Settings

```

--dest-namespace=db2u
  
```

The panel that provides this capability:

```
Db2 Shift [/home/c2cdb2/c2c]
Copy or Retrieve Database Clone
Copy Type      :  Copy Clone to Target Pod  Copy Target Clone to Local
Source Details
Directory      : /tmp/cache_-----+
POD Details
Directory      : /tmp/cache_-----+
Target Location :  OpenShift  Kubernetes
POD Project    : -----+
POD Name       : -----+
```

ESC•Quit ^?•Field Help ^X•Review and Execute

Mode Option

Syntax: `--mode=push_clone, --mode=pull_clone`

```
Copy Type      :  Copy Clone to Target Pod  Copy Target Clone to Local
```

The Db2 Shift will copy a cloned database to a pod or retrieve the contents of a cloned database back to the local instance.

Source Clone Directory

Syntax: `--local-dir=""`

```
Directory      : /tmp/cache_-----
```

The cloned copy of the database will be retrieved or stored in this location depending on what type of copy operation is being performance.

Target Clone Directory

Syntax: `--dest-dir=""`

```
Directory      : /tmp/cache_-----
```

The cloned copy of the database will be placed into the target directory that is specified by this field. If you are pulling a copy (`--mode=pull_clone`), this field contains the cloned database that is going to move to the local directory.

Target Client

Syntax: `--oc`, `--kubect1`

```
Target Location : ● OpenShift ○ Kubernetes
```

The pod client for a copy operation must be supplied as part of the Db2 Shift command. Only one of the following clients must be used:

- `--oc` OpenShift Destination
- `--kubect1` Kubernetes Destination

If the client is Kubernetes (`--kubect1`) or OpenShift (`--oc`), the program requires that the appropriate `kubect1` or `oc` client has been installed locally and that the namespace or project has already been specified.

Destination Server (POD)

Syntax: `--dest-server="pod_name"`

```
POD Name : c-demo-db2u-0_-----*
```

For deployments to OpenShift, Kubernetes, or CP4D, you must supply the name of the POD that Db2U is running in. The OpenShift or Kubernetes client should be used to connect to the target namespace or project before issuing the Db2 Shift command.

Destination Pod Namespace or Project

Syntax: `--dest-namespace=""`, `--dest-project=""`

```
POD Project : db2u_-----
```

In Kubernetes deployments, the location of a pod is associated with a namespace, while in OpenShift deployments, the pod is associated with a project.

When authenticating to a Kubernetes or OpenShift environment, it is recommended that the local client be connected to the project or namespace that the Db2U pod is running in.

If you do not supply a namespace or project value, the Db2 Shift program will assume that you are already connected to that project. If this is not the case, the program will stop with an error when it attempts to find the pod.

To have Db2 Shift connect to the appropriate project or namespace, supply the value of the namespace or project using this option.

15

Best Practices

HINTS, TIPS, AND RECOMMENDED
SETTINGS

Chapter 15: Best Practices

There are several best practices that we recommend when using the Db2 Shift utility. Many of these recommendations are evolving and updates will be added to this document on a regular basis.

Database Migration

The Db2 Shift command will migrate a database from Version 10.5 and 11.1 to the latest version of Db2 (11.5). Databases that are at older release levels will need to be migrated to 10.5 or 11.1 before using the Db2 Shift utility.

When performing a Db2 Shift on an older release, the database must be shut down or DEACTIVATED before a shift can be performed. All Db2 upgrades require that the database be in a fully consistent state for the upgrade to take place. The utility will fail if the database was in a WRITE SUSPEND state.

If the Db2 Shift utility is run against a database which requires migration, it will check to see if `--offline` or `--online` was supplied in the command line or in the user interface. If `--offline` was set, the Db2 Shift utility will shift the files assuming the database is offline. If the database was not offline, the upgrade at the target will fail because the control information in the database files will not be consistent.

If `--online` was selected, and a migration is required, Db2 will force the database into a DEACTIVATED state. This will cause a database outage and potentially rollback transactions that were in progress. Make sure that the database does not have any workloads running against it if you decide to shift when using the `--online` mode. Note that having a database in DEACTIVATED state is not a guarantee that no one can connect to the database.

The other option to reduce outage time for a database is to use the clone option. A clone copy can be used for an upgrade and the database can be write suspended when using this mode of operation.

Make sure to prune your log archive files to minimize the amount of data that is transmitted during the shift step:

```
db2 prune history yyyyymmdd
```

In summary, when shifting a database that needs an upgrade, using one of the following techniques:

- Shut down the database and use the `--offline` flag
- Use the `--online` flag and let Db2 DEACTIVATE the database
- Clone the database (and deploy later) using the `--online` flag

Online Versus Offline Mode

Db2 Shift provides two options when dealing with the state of a database during the shift process:

- `--online` mode – the database is online while the shift is taking place
- `--offline` mode – the database has been shutdown

Online Mode

Online mode is the default value for all shifts. While this can be used for shifting databases that require an upgrade, the recommendation is to use `--offline` mode instead.

When shifting a database in `--online` mode, the file copy step will not affect the status of the database. Workloads can continue to run while the copying is taking place. Note that the buffer pools may be affected by the reading process so there may be lower cache utilization while the utility is running.

During the last step of the shift process, Db2 Shift will search for any updates that may have been applied to the database after the initial copying was done. To ensure the integrity of the data being copied, the database will be placed into a `WRITE SUSPEND` mode. This last step should only take a few seconds and the database will be `WRITE RESUMED` when the final copy is done.

When the database is suspended, all read activities will continue. New connections will not be permitted, but any applications that are currently running will be allowed to continue. Those transactions that are updating records will be temporarily "paused" while the copying is done.

Once the copy is complete, a suspended application will finish their transaction. Online mode provides the least disruption to an active database but is not recommended for databases that being upgrade.

Offline Mode

Offline mode should be used for database migrations. If it is not used, the Db2 Shift utility will `DEACTIVATE` the database before continuing.

Offline mode indicates to Db2 Shift that the database has either been `DEACTIVATED` or the instance has been shut down. If you want to use offline mode, you must first gather the control information while the database is online. To do this, either through the command, or the UI, select `--blank-slate=true --verify`.

The `--blank-slate=true` option will gather the source and target control information that is required to do a shift. If the source database was already offline, this information would not available.

Adding the `--verify` option will check that all the settings are correct between the source and target and stop execution (i.e., the shift will not occur).

After your shut down the source database you would use the option `--blank-slate=false`. This will force Db2 Shift to use the existing control information that you generated in the previous step. At this point the database will be a consistent state and the upgrade step will proceed at the target location.

Instance Ownership

If you are moving a database that was created with an instance owner that is **not** found on the target system, you must make sure to add the instance owner at the target into the current database **before** shifting the database.

Example: Database CQW was created by instance owner `db2inst2`. This database is then cloned to another instance that has `db2inst1` as the instance owner. When `db2inst1` attempts to access the database, they will be unable to manage it because they do not have DBADM or SECADM privileges on the tables created by `db2inst2`.

There are two ways to solve this. One is to create the userid `db2inst2` at the destination location for a traditional instance. You would then be able to connect to the database as `db2inst2` and grant DBADM and SECADM back to `db2inst1`. However, this approach will not work with Db2U and CP4D installations because there is no mechanism available within the POD to create a new OS-level userid.

The second (and easier) method is to add the destination instance owner to the source database.

```
GRANT SECADM ON DATABASE TO USER db2inst1
GRANT DBADM ON DATABASE TO USER db2inst1
```

The `db2inst1` userid does not need to exist in the Operating system to grant these privileges to the userid. Once the database is shifted to the new location, the instance owner `db2inst1` will have full access to the database.

Target Db2 Environment

The Db2 Shift program can clone an existing Db2 database into two environments:

- Db2U Pod running on OpenShift, Kubernetes, Cloud Pak for Data
- Db2 Instance on premise, on a virtual machine (on premise or Cloud)

Cloning into a Containerized Environment

For instances where you are shifting to a Db2U POD, the POD must already exist, and the database must have been created. Depending on what you plan to do with the target database, the database name must be:

- The same as the SOURCE database if:
 - You are connecting the source and target with HADR
 - You are cloning the database into Cloud Pak for Data
 - You want to keep the same database name
- Any valid database name

If you move a database (i.e., SAMPLE) into a Db2U POD database called DB2OLTP, the contents of the database SAMPLE will end up in DB2OLTP. In other words, the SAMPLE database is shifted and renamed to DB2OLTP. If you want to have the same database name, then you must create the database with the same name.

Cloning into a Traditional Db2 Instance

When cloning a Db2 database into a traditional instance, the Db2 database can already exist, or you can ask that the database be generated for you at shift time. The Db2 instance must be available and running on the server.

Syntax: `--dest-create-db`

```
Database Name      : _____  ○ Create Database  ○ Use Mirrored Paths
```

To create the database at shift time, make sure to select the **Create Database** setting in the UI menu or use `--dest-db-create` flag in the command line. The Db2 Shift program will attempt to create the database in a directory structure under `/db2_portable` on the target system.

When cloning into a traditional Db2 instance, similar rules apply to the database name. The database name must be:

- The same as the SOURCE database if:
 - You are connecting the source and target with HADR
 - You want to keep the same database name
- Any valid database name

HADR Considerations

The HADR option can be used to create a standby version of the primary database for either HA capabilities, or for syncing the source and target until the decision is made to move all database workloads to the standby.

Syntax: `--mode=hadr_setup`

```
Move Options      : ○ Database ○ Database/LDAP/DMC ● Move Database for HADR
```

```
HADR Option      : ● Move Database for HADR
```

The requirements for creating an HADR pair are:

- The databases must both be at the same level (Upgrade not supported)
- The primary must have logging enabled and a full backup generated before shifting

```
update db config for trading using logarchmeth1
    DISK:/home/db2inst1/logfiles
update db config for trading using logindexbuild on
connect reset
backup database trading to /dev/null
```

- Set Read on Standby if required

```
db2set DB2_HADR_ROS=ON
db2stop force
db2start
```

- Open ports on the primary (3700) and on the secondary only if it is a traditional instance

The steps for creating an HADR primary and standby pair with Db2 Shift are:

- Run a shift (or clone) to an instance or POD while the database is online
 - The target database must have the same database name – Db2 Shift will not run if they have different names
- Run the HADR setup for POD or Instance

At this point the two databases will be in NEARSYNC mode and will be in an HA configuration until you decide to turn the standby into the new primary.

HADR to CP4D POD

If you choose to use HADR to connect to a standby Db2 running on CP4D, you will need to use the LDAP/DMC utility **after** you have switched the CP4D Db2 POD into the primary.

Syntax: `--mode=sec_and_monitor`

At this point you must enable the LDAP connection to CP4D as well as register the database to the Data Management Console on CP4D. While the database was running as a standby database, the LDAP and DMC connections settings could not be updated because the database was in read-only mode and needed to be identical to the primary database.

After the switch to primary status, you are now able to register the database to CP4D and be able to manage it from the DMC console.

HADR using a Clone Copy

You can use a Clone copy of a database to create an HADR pair. The Clone copy must have been created with the following pre-requisites:

- The databases must both be at the same level (Upgrade not supported)
- The primary must have logging enabled and a full backup generated before generating the Clone
- The primary was online during the cloning process

Once you create the Standby database (Deploy to Instance), you must copy the following files from the target server (where you ran the Db2 Shift deploy command) and copy them to the source system where you ran the original Db2 Shift clone command:

- `source.settings`
- `destination.settings`

At the source system, run the HADR initialization step for a POD or Instance.

When the HADR initialization is run after a standard shift operation, the settings files would have been generated at the same time. Since you are using a Clone copy, there is no information available about the destination server, so these files need to be made available to Db2 Shift for the HADR command to work.

Database Storage Relocation

When shifting a database to a POD or to another Db2 instance, the storage paths will need to be modified for Db2 to run. The `dft_paths.cfg` file contains information on where the Db2 Shift utility will place database objects on the destination server. If the `dft_paths.cfg` file does not already exist in the directory where the Db2 Shift utility is running, the file will be created during the first execution of the utility.

The `dft_paths.cfg` file contains 2 sets of paths, one for DB2u Pod/Container deployments and the other for standard Instance installs classified as `server-type other`.

You can alter the `type other` paths in the file as long as you do not change the `DFT_STORAGE_PATH` variable. The `{}` characters in a path signifies where `$HOME`

will be substituted into the string. If the `{}` characters are removed, the absolute path will be used.

A copy of the file is shown below. The format of the file is:

```
PATH | Directory | type
```

Where:

- Path Type – The type of path that the directory represents
- Directory – The directory that will be used for the path type
- Target – pod for Db2U deployments and other for Db2 instance deployments

```
MIRRORLOG_PATH|/mnt/bludata0/db2/mirrorlog|pod
FAILARCHIVE_PATH|/mnt/bludata0/db2/failarchive|pod
LOGARCHMETH1|/mnt/bludata0/db2/archive_log|pod
LOGARCHMETH2|/mnt/bludata0/db2/archive_log2|pod
OVERFLOWLOG_PATH|/mnt/bludata0/db2/overflowlog|pod
DFT_STORAGE_PATH|/mnt/bludata0/db2/databases/|pod
OTHER_STORAGE_PATH|/mnt/bludata0/db2/databases/SPATH|pod
OTHER_LOG_PATH|/mnt/bludata0/db2/dblogs|pod
MIRRORLOG_PATH|{}/db2_portable/mirror_logs/mirrorlog|other
FAILARCHIVE_PATH|{}/db2_portable/failarchive|other
LOGARCHMETH1|{}/db2_portable/archive_log|other
LOGARCHMETH2|{}/db2_portable/archive_log_mirror|other
OVERFLOWLOG_PATH|{}/db2_portable/overflow_logs/overflowlog|other
DFT_STORAGE_PATH|{}/other
OTHER_LOG_PATH|{}/db2_portable/primary_logs/dblogs|other
NEW_DB_ON|/db2_portable/databases|other
OTHER_STORAGE_PATH|{}/db2_portable/alt_store_paths/SPATH|other
```

You must update this file **prior** to running a shift operation. The Db2 Shift utility will use the existing `dft_paths.cfg` file to relocate the directories into the target location.

Note: The `dft_path.cfg` is setup to transfer multiple databases and their paths under the `/$HOME/db2_portable` directory.

The following paths, as defined in `dft_paths.cfg` can be modified using the mount utility on Linux to point them to a different directory.

```
MIRRORLOG_PATH|{}/db2_portable/mirror_logs/
FAILARCHIVE_PATH|{}/db2_portable/failarchive/
LOGARCHMETH1|{}/db2_portable/archive_log/
LOGARCHMETH2|{}/db2_portable/archive_log_mirror/
OVERFLOWLOG_PATH|{}/db2_portable/overflow_logs/
OTHER_LOG_PATH|{}/db2_portable/primary_logs/
```

Note that the mount path depends on the path type e.g., `MIRRORLOG_PATH` is not necessarily at the end of the path defined in the default config. That is because of the unique naming requirement for paths and the assumption that paths need to be multi-tenant capable.

For instance:

- Mirror log path:
MIRRORLOG_PATH = `{}/db2_portable/mirror_logs/mirrorlog`
- Mount path for db2inst1 alternate mirrored logs volume:
`/home/db2inst1/db2_portable/mirror_logs`

If we were to change the config file simply to point at the mirror logs path using an alternate absolute path, your config file entry would look like:

```
MIRRORLOG_PATH|/mnt/mirror_logs/mirrorlog|other
```

For your absolute path you would need to create the directory prior to the shift :

```
mkdir /mnt/mirror_logs
```

Note: The Db2 Shift tool creates the `dft_paths.cfg` on start-up if it does not already exist. If you need to recreate the file with default values, simply delete the file. If the file already exists in the directory, it will not be recreated by the tool.

Note: DFT_STORAGE_PATH and NEW_DB_ON paths should not be changed. NEW_DB_ON controls where any database created by the tool goes on the target instance, and this path is always relative to the \$HOME directory for the instance. If you want the database created elsewhere, manually create it first before running the tool. The other path changes, as per above, will be observed.

For alternate Storage paths you can mount the volume at a lower level, but you must know how many paths will be included. This can be determined by looking at the `new0.cfg` file after a generate settings run.

Syntax: `--blank-slate=true, --gen-settings`



The `--gen-settings` (Generate Settings) option is used in conjunction with `--blank-slate`. The use of `--gen-settings` will prevent Db2 Shift from continuing execution after the metadata files have been created. Storage can also be set up by MLN or NODE.

```
OTHER_STORAGE_PATH|{}/db2_portable/alt_store_paths/SPATH[1..N]/NODE000X
```

For Example:

```
/home/db2inst2/db2_portable/alt_store_paths/SPATH1/db2inst2/NODE0000  
/home/db2inst2/db2_portable/alt_store_paths/SPATH1/db2inst2/NODE0001  
/home/db2inst2/db2_portable/alt_store_paths/SPATH1/db2inst2/NODE0002  
/home/db2inst2/db2_portable/alt_store_paths/SPATH1/db2inst2/NODE0003  
/home/db2inst2/db2_portable/alt_store_paths/SPATH2/db2inst2/NODE0000  
/home/db2inst2/db2_portable/alt_store_paths/SPATH2/db2inst2/NODE0001  
/home/db2inst2/db2_portable/alt_store_paths/SPATH2/db2inst2/NODE0002  
/home/db2inst2/db2_portable/alt_store_paths/SPATH2/db2inst2/NODE0003  
/home/db2inst2/db2_portable/alt_store_paths/SPATH3/db2inst2/NODE0000  
/home/db2inst2/db2_portable/alt_store_paths/SPATH3/db2inst2/NODE0001
```

Chapter 15: Best Practices

```
/home/db2inst2/db2_portable/alt_store_paths/SPATH3/db2inst2/NODE0002  
/home/db2inst2/db2_portable/alt_store_paths/SPATH3/db2inst2/NODE0003
```

You will have to pre-create the paths at the target server using this pattern prior to shifting the database into it.

If you apply the mount at a lower point in the directory structure, it will likely be deleted in the Db2 Shift process.

Note: No changes to the `dft_paths.cfg` are supported for moves to containers which are identified with the `pod` keyword at the end of a control line.

Force Database Creation and Mirror Paths

The Db2 Shift utility provides an option to create a database at the target location (Instance Only). The option is available on the Shift to Instance and Deploy to Instance menus.

Syntax: `--dest-create-db --mirror-path`

```
Database Name      : _____   Create Database   Use Mirrored Paths
```

The following describes how Db2 Shift handles the creation of databases at the target instance:

1. If no options are set:
 - a. If the target database does not exist, an error message will be returned, and processing stops
 - b. Otherwise, the contents of the target database will be replaced with the source database using the target's home database directory
2. If `--dest-create-db` is set:
 - a. If the target database exists, it will be dropped!
 - b. The target database is created using the path `/$HOME/db2_portable`
3. If `--mirror-path` is set:
 - a. If the target database does not exist, an error message will be returned, and processing stops
 - b. Otherwise, the contents of the target database will be replaced with the source database using the same paths as the source database
4. If `--dest-create-db` and `--mirror-path` is set:
 - a. If the target database exists, it will be dropped!
 - b. The target database is created, using the same paths as the source database

Scenarios 3 and 4 require special care in their usage and are discussed below.

Mirror Path (Only)

When a request is made to use an existing database using `--mirror-path`, the following must be true:

- The directory structure used by the source database must be replicated at the target database
- The instance owner and database name must be identical
- The target instance owner must have appropriate permissions to read/write to these directories

The shift or deploy operation will fail if a directory structure is not found at the target site.

Example: Database TRADING was created at the source using the following syntax:

```
CREATE DATABASE TRADING ON /path1, /path2, /path3
```

For the `--mirror-path` option work correctly, the destination database must be created using the identical paths:

```
CREATE DATABASE TRADING ON /path1, /path2, /path3
```

You cannot change the name of the database when using this technique.

Mirror Path and Force Database Creation

It is possible for the Db2 Shift Utility to create a database using mirror paths, but there are a few limitations when using this feature. When Db2 Shift creates the target database, it will take all the paths, except the default database path, and use them as the destination paths on the target instance. The paths must already exist, or the Db2 Shift utility must be able to create the paths.

If the generated database configuration directory number (e.g., SQL00001) does not match across source and target systems, an error will occur. In this case you should manually create the target database on a different path. This is a limitation of the Db2 relocation utility.

A simple example highlights the problem. Running a `db2 list db directory` on your source system may result in 4 databases being displayed: SAMPLE, HR, ACCOUNTS, and BANKING. The target instance currently has no databases. When attempting to shift ACCOUNTS to the target system using `--dest-create-db` and `--mirror-path`, the shift will fail because ACCOUNTS is using SQL00003 at the source and it will be created as SQL00001 at the target.

The scenario that best suits the `--dest-create-db` and `--mirror-path` option is a Db2 instance that contains only 1 database that is being shifted to an empty Db2 instance.

Encrypted Databases

The Db2 Shift utility can only shift encryption keys that are available locally to the database. If your database uses an enterprise key manager, then you will need to register that key manager at the target pod or instance.

If you are shifting to another Db2 instance or pod that contains multiple databases, you cannot shift the key file. During a shift, the target key file is completely replaced which would remove any keys being used by existing databases at the target location. If you want to shift the database to the new location, you will have to extract the current key and reimport it into the target key manager.

Database, Instance, and Environment Settings

The Db2 Shift utility will maintain all database settings when the database is shifted to a different Db2 instance or pod. However, the Instance and Environment settings are not moved.

The Db2 Shift provides an option to override the target instance settings. If a particular setting needs to be updated, you can provide the value as part of the UI or command line. An example of a setting that you may want to update during a shift is the `INSTANCE_MEMORY` and the `INTRA_PARALLEL` settings.

You should ensure that any change you make at the instance level are reflected in the Db2U POD configuration in the event the Db2U code is replaced.

Environment settings are not handled by the Db2 Shift program. You must ensure that the instance or Db2U POD have the appropriate environment variables set before you shift the database. For instance, if you are shifting a Db2 database that has Oracle compatibility turned on, you must create an empty Db2 database with this setting turned ON before shifting the original database. There is no way to set the compatibility flag after the database has been created.

Threads and Compression

One area that is still evolving is the use of threads and compression when shifting a database into a cloud environment. The Db2 Shift tool has been set to use default settings that give the best performance in networks that have 1Gb/s throughput. These settings will need to be adjusted based on your network and machine characteristics.

The following are some general observations from early testing.

Threads

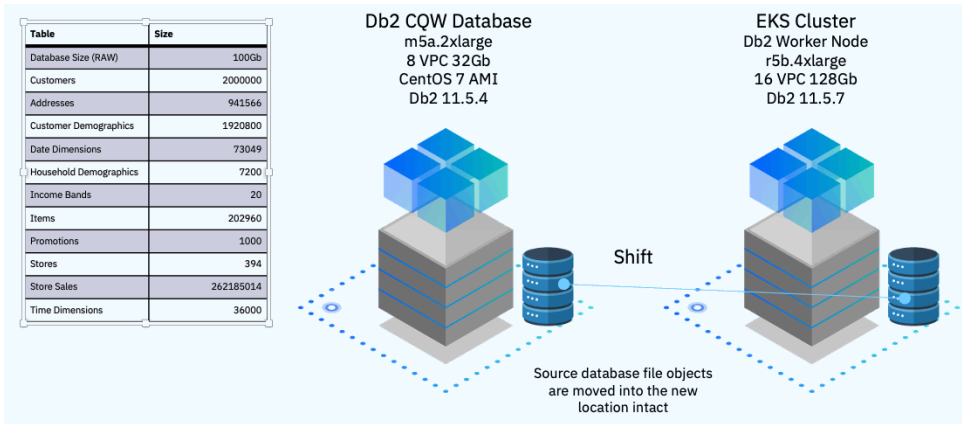
Using a higher number of threads is useful in situations where the Db2 database has multiple paths defined for a storage group. If the database was created with a single storage path, the threads are competing for I/O on the same device.

Increasing the number of threads will not necessarily result in more throughput. Testing has shown that 4 threads is a good starting point for parallelism, with small incremental benefits as you increase the number. You need to balance the CPU usage by the Shift utility and the impact on workloads running on the server.

If the Db2 Shift command is being used for gradual shifts (initial, refresh), the number of threads should be set to 1 to reduce the overhead on the server. Once the final shift is run you may want to increase the value to minimize the suspend time of the database.

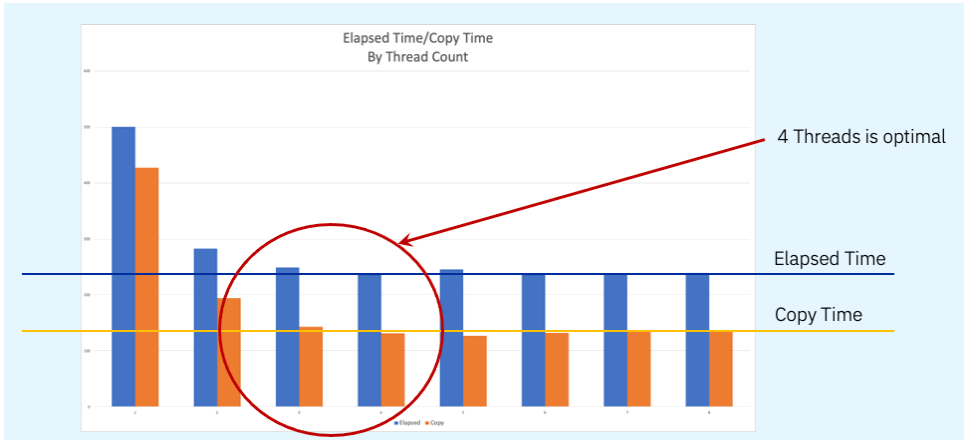
If there is no contention on the server then the maximum thread count should be 1 less than the number of VPCs of the server or 8, whichever is less. For a cloud instance that is defined with 4 cores, 8 VPCs (SMTx2), the thread count should be a maximum of 7 (8 VPCs - 1). For larger servers, a maximum of 8 threads can be used.

Test Scenario



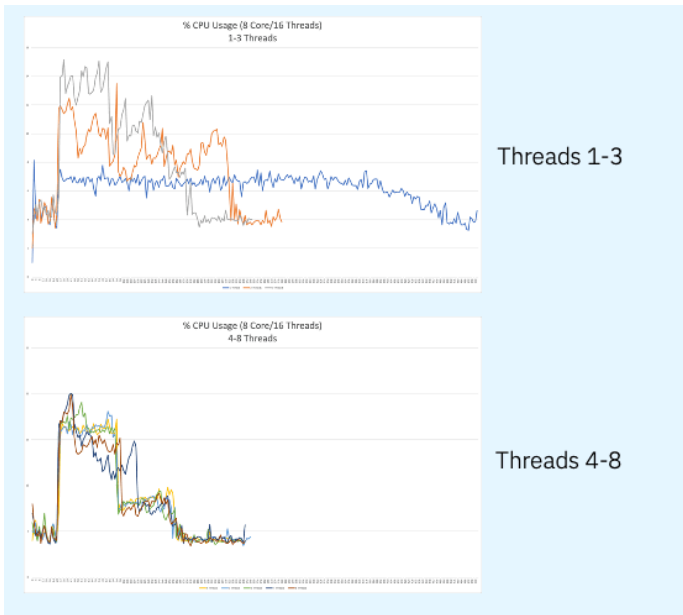
The Db2 Shift command was run from a standard EC2 instance running Db2 on Linux and shifting the database to an EKS cluster with Db2U installed. The database size was approximately 50Gb with 260+M rows of transactional data. The elapsed times of the shift are plotted against the number of threads used for the rsync process.

Chapter 15: Best Practices



The best elapsed time was 237 seconds (131s copy time only). Increasing the number of threads did not make a difference to the total elapsed time.

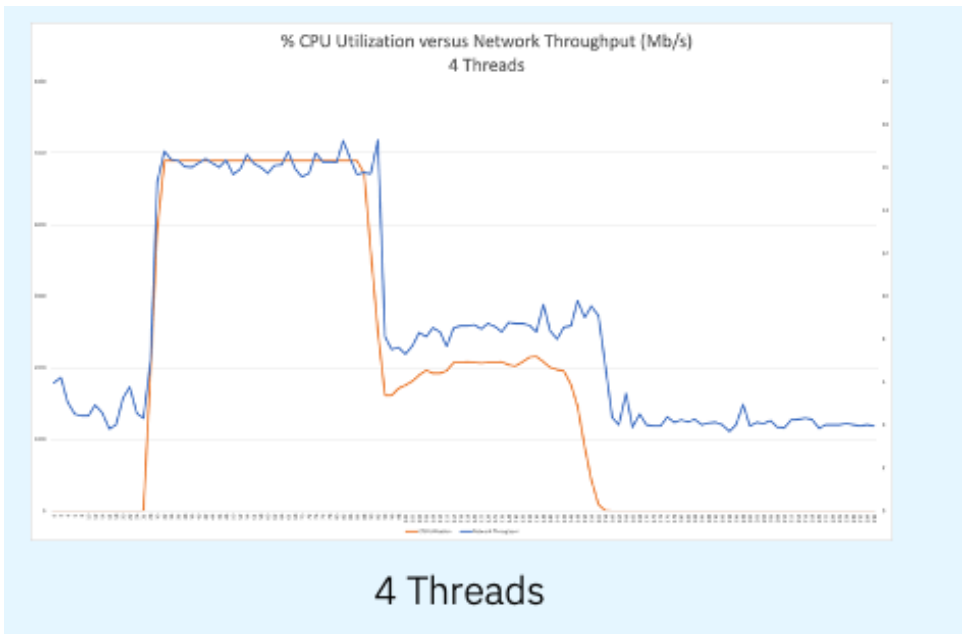
Examining the CPU usage, the average utilization of the threads was almost identical between 4 and 8 threads.



The network throughput appears to track with the CPU usage.



Overlaying the 4 thread CPU utilization and Network throughput demonstrates that the CPUs are busy transmitting as much data onto the network as possible.



The results showed that the maximum throughput was capped by the network limit of 5Gb/s. The network throughput was:

- 1 thread was 1240 Mb/sec
- 4 threads were 4890 Mb/sec

A CPU thread has a limit on how much data it can push onto the network. By running a test on a single thread, you can determine how many cores you can effectively use during a Shift run. Dividing the network capacity by the single core performance will determine the optimal number of threads to use.

Example:

- Throughput of one thread is approximately 1.2Gb/s
- Network limit is 5Gb/s
- $5/1.2$ is approximately 4 threads

This result can also be used to determine your total copy time:

- $\text{Database Size}/(\text{Network Limit}/8) = \text{elapsed time}$
- $50 \text{ GB}/(5\text{Gbs}/8) = 50\text{GB}/(.625\text{GBs}) = 80\text{s}$ with ideal conditions
- Tests results were $50\text{GB}/(4.89\text{Gbs}/8) = 128\text{s}$ ideal (131 observed)

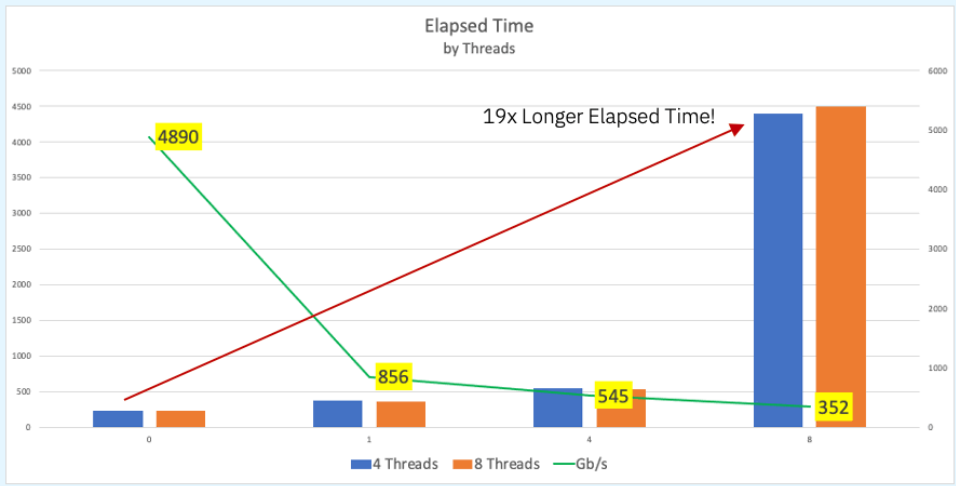
Compression

Compression will significantly reduce throughput if you are CPU constrained. If the database is encrypted, uses compression, or has a high ratio of numeric values over character values, adding compression will not help throughput. In some cases, this may significantly reduce throughput.

The recommendation is to use compression only if you have a very slow network (<100Mbps). For these types of situations, the compression may result in less data being transferred which will reduce the load on the network. Aside from this situation, leaving compression at 0 is the best strategy for initial Db2 Shift tests.

Test Scenario

The identical configuration for the thread tests was used to determine network usage and elapsed time for different compression values.



The chart compares 4 different shifts with 4 or 8 threads (4 was optimal) and compression settings of 0, 1, 4, and 8. The numbers that are highlighted in yellow represent the maximum network throughput that was achieved with the different levels of compression and threads. For all tests, the difference between 4 and 8 threads was negligible. However, the reduction in network throughput and elapsed time increased dramatically as the compression value was increased.

The conclusion from these tests shows a compression value between 1 and 4 may be useful in environments where network throughput is less than 1Gb/s. For networks with higher throughput, a compression value of zero should be used.

A

Options Reference

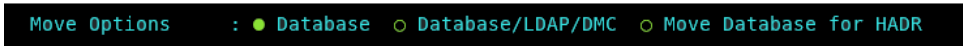
SYNTAX SUMMARY

Appendix A: Options Reference

The following section includes all the parameters that are used by the Db2 Shift program. Each section will display the syntax for that option, along with an image of the field in the UI that you would complete or select.

Mode Option

```
--mode=[all,move,clone,apply_clone,sec_and_monitor,  
hadr_setup,push_clone,pull_clone]
```

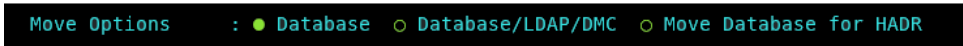


The MODE option determines what steps the Db2 Shift program will take to move your database to the new location. When using the UI to generate the scripts, the mode option may not be displayed since it is explicitly created for you. For instance, when cloning a database, the `--mode=clone` will have been generated as part of the command.

The following are the values that mode option can have.

Move Database Only

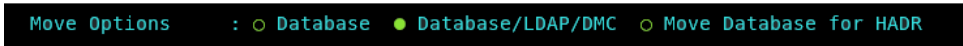
Syntax: `--mode=move`



The MOVE option will take a copy of your Db2 database and copy it into the target Db2U container or instance. No other processing is done with the database.

Move Database and Set up LDAP and DMC Settings

Syntax: `--mode=all`



If you choose the ALL option, the database will be moved, and the program will also apply any security and console settings that are required. This is equivalent to running `mode=move` and `mode=sec_and_monitor` one after another. The ALL option is necessary in CP4D environments where the LDAP settings need to be updated and the DMC console needs to be reset to recognize the new database.

This option is not applicable for Db2 instance to Db2 instance moves.

Appendix A: Options Reference

Clone a Database

Syntax: `--mode=clone`

The CLONE option can be used to create a copy of database to be moved to another location and then deployed. This feature is useful in situations where the target system cannot be reached through the network due to firewall issues.

Apply a Clone to a POD or Instance

Syntax: `--mode=apply_clone`

The APPLY_CLONE option will take an existing CLONE database and shift it to a POD or a Db2 instance. There is no need to be connected to the original database because all the control information is contained within the CLONES image.

Copy a Clone from or to a Pod

Syntax: `--mode=push_clone, --mode=pull_clone`

```
Copy Type          : ● Copy Clone to Target Pod ○ Copy Target Clone to Local
```

The Db2 Shift will copy a cloned database to a pod or retrieve the contents of a cloned database back to the local instance.

HADR Setup

Syntax: `--mode=hadr_setup`

The HADR option is used to initialize and start HADR between a source and target server. This step would be run after the database has been shifted to the new location. This option is not used for the initial HADR setup of the target system. The target database must be created with the `--hadr` option for it be placed into the correct mode for HADR communication. The `--hadr` is enabled for pods with the Move Database for HADR option in the menu.

```
Move Options       : ○ Database ○ Database/LDAP/DMC ● Move Database for HADR
```

When moving a database to another instance, the HADR option is on a separate line.

```
HADR Option        : ● Move Database for HADR
```


LDAP Security and DMC Setup

Syntax: `--mode=sec_and_monitor`

This setting only applies when the destination is a POD on K8s or OpenShift. If an associated LDAP repository exists, the `db2inst1` user is added. If the IBM Data management console is deployed with a Cloud Pak for Data system, it resets the console and applies the relevant DDL and privileges for monitors.

Copy Mode

Syntax: `--mode=push_clone, --mode=pull_clone`

```
Copy Type      : ● Copy Clone to Target Pod ○ Copy Target Clone to Local
```

The Db2 Shift will copy a cloned database to a pod or retrieve the contents of a cloned database back to the local instance.

Target Client Instance to Instance

Syntax: `--ssh, --local`

```
Target Location : ● Remote ○ Local
```

The client for a deploy (clone) operation must be supplied as part of the Db2 Shift command. Only one of the following targets must be used:

- `--ssh` Remote Db2 Instance
- `--local` Local Db2 Instance

If the target is a remote Db2 instance (`--ssh`), Db2 Shift expects that a passwordless ssh environment has been established between the source and target servers.

The `--local` option only applies to Clones (database copies) that are being deployed onto the same server that the Db2 Shift command is running on. The ability to shift a database locally (i.e., make a copy of a Db2 database on the same instance) is currently disabled due to some restrictions with `db2relocatdb`.

Target Client Instance to POD

Syntax: `--oc, --kubect1, --local`

```
Target Location : ● OpenShift ○ Kubernetes
```

```
Target Location : ● OpenShift ○ Kubernetes ○ Local
```

The pod client for a deploy (clone) operation must be supplied as part of the Db2 Shift command. Only one of the following clients must be used:

Appendix A: Options Reference

- `--oc` OpenShift Destination
- `--kubect1` Kubernetes Destination
- `--local` Copy of the cloned database is inside the pod

If the client is Kubernetes (`--kubect1`) or OpenShift (`--oc`), the program requires that the appropriate `kubect1` or `oc` client has been installed locally and that the namespace or project has already been specified.

The `--local` option is used when the cloned copy of the database has already been copied inside the POD that it is being shifted to.

Source Database

Syntax: `--source-database=""`

```
Database Name : lights_
```

The source database is the name of the database that you want to move to the new location. Note that you can have the same or different database name at the target. If you provide a different database name at the target, the program will copy the database from the source and place it on the target and use the existing name.

Source or Instance Owner

Syntax: `--source-owner="instance name"`

```
Instance Owner : db2inst1_-----
```

The Db2 Shift program assumes that the current userid you are logged into is the owner of the instance. This is necessary due to the requirement to access the underlying files that are used by Db2. If you supply the source owner value, Db2 Shift will double check that you are working in the correct instance and the settings files are consistent.

Destination Database

Syntax: `--dest-database=""`

```
Database Name : lights_
```

The destination database name can be the same as the source database, or a completely different name. Make sure that if you are creating a database with a different name that it doesn't currently exist on your target system. Otherwise, the target database will be deleted!

Force Destination Database Creation

Syntax: `--dest-create-db`

```
Database Name      : flights_ ● Force Database Creation
```

During a shift or clone operation to a traditional Db2 INSTANCE, the Db2 Shift program will check whether the target database exists. If it does not, the Db2 Shift operation will fail. If the Force Database Creation setting is on, Db2 Shift will attempt to create the database first and then run the shift operation.

Note: Db2 Shift will create the database in a directory structure under `/db2_portable` on the target system.

Mirror Path

Syntax: `--mirror-path`

```
Database Name      : _____ ○ Create Database ○ Use Mirrored Paths
```

Mirror path is a valid switch for Applying a Clone or moving a database to a standard db2 instance.

It will take all the paths, except the default database path, and use them as the destination paths on the target instance. It assumes that those paths exist already, or it can create them. The database names and instance owners must also be the same.

If used in combination with `--dest-create-db`, the database that is created, rather than being created under the `/home/$DB2INST/db2_portable` directory, will be created on the same path as the source system. If using this option into a path (even if it is the default database path) you may receive an error if the database configuration directory number e.g., `SQL00001` does not match across source and target systems. In this case you should manually create the target database on a different path. This is a limitation of the Db2 relocation utility.

When using this option, you must be cautious that the target paths on the destination are on paths that the instance owner has permissions to create the underlying directory structures, and that you have free space to create the database.

Destination Owner

Syntax: `--dest-owner="instance name"`

```
Instance Owner : db2inst1_-----
```

The destination owner value is primarily used when settings up HADR between a primary and secondary instance. However, the value can be supplied so that Db2 Shift can check that the correct credentials are available at the target site.

Note that the `--dest-owner` parameter is only required when doing an HADR setup.

Destination Server POD

Syntax: `--dest-server="pod_name"`

```
POD Name : c-demo-db2u-0_-----+ POD Pro
```

For deployments to OpenShift, Kubernetes, or CP4D, you must supply the name of the POD that Db2U is running in. The OpenShift or Kubernetes client should be used to connect to the target namespace or project before issuing the Db2 Shift command.

Destination Server Instance

Syntax: `--dest-server=userid@ip.address`

```
Server Address : some.server.com_-----
```

For destinations that are traditional Db2 instances, you must provide the userid and the IP address or symbolic name of the destination server. You only use an IP address for shifting into a traditional Db2 instance.

The format of the parameter is `userid@address` when using the Db2 Shift command line. When using the UI, this field is generated automatically by combining the destination instance owner name with the destination server address.

When connecting to a remote instance, the Db2 Shift program expects that a passwordless ssh environment has been established between the source and target servers. The `--ssh` flag must be used in conjunction with this parameter.

HADR Setup

Syntax: `--hadr` or `--keep-rfw-pending`

```
HADR Option      : ● Move Database for HADR
```

When HADR mode is selected, the database will be copied over to the target location and initialized as an HADR secondary. The database can now be connected to the primary database as an HADR pair and participate in failover or read-only applications.

This option cannot be used if your database needs to be migrated or if the database needs to be renamed.

HADR Source or Destination Server

Syntax: `--source-hadr-server=""`, `--dest-hadr-server=""`

```
Source HADR IP   : some.server.com.....+
Target HADR IP   : other.server.com.....
```

For HADR setup, the Db2 Shift command requires the IP or symbolic name of the source server that will be used in an HADR setup. This server is referred to as the primary server.

If the destination server is a standard Db2 instance, use the IP address of the server. If the destination target is an OpenShift or Kubernetes cluster, use the address of the load balancer.

HADR ports

Syntax: `--source-hadr-port=#`, `--dest-hadr-port=#`

```
Source HADR Port : 3700_
Target HADR Port : 3700_
```

HADR communicates over a port which is different than the Db2 instance. You must supply the source and destination port numbers that Db2 will communicate between the HADR servers. The default port number is 3700 for HADR communications but verify the value. The target port number will also be required.

Destination Pod Namespace or Project

Syntax: `--dest-namespace=""`, `--dest-project=""`

```
POD Project      : db2u_.....
```

In Kubernetes deployments, the location of a pod is associated with a namespace, while in OpenShift deployments, the pod is associated with a project.

When authenticating to a Kubernetes or OpenShift environment, it is recommended that the local client be connected to the project or namespace that the Db2U pod is running in.

If you do not supply a namespace or project value, the Db2 Shift program will assume that you are already connected to that project. If this is not the case, the program will stop with an error when it attempts to find the pod.

To have Db2 Shift connect to the appropriate project or namespace, supply the value of the namespace or project using this option.

Clone Directory

Syntax: `--clone-dir=""`

```
Directory       : /tmp/cache_.....
```

The cloned copy of the database will be placed into the directory that is specified by this field. When the database is cloned, the contents of the directory can be moved to a new location and the `--mode=apply_clone` option used to shift the contents of the clone into the Db2 database.

Source Clone Directory

Syntax: `--local-dir=""`

```
Directory       : /tmp/cache_.....
```

The cloned copy of the database will be retrieved or stored in this location depending on what type of copy operation is being performance.

Target Clone Directory

Syntax: `--dest-dir=""`

```
Directory      : /tmp/cache_-----
```

The cloned copy of the database will be placed into the target directory that is specified by this field. If you are pulling a copy (`--mode=pull_clone`), this field contains the cloned database that is going to move to the local directory.

Synchronization Options

Syntax: `--sync=[start_sync, rerun_sync, finish_sync]`

```
Sync Options   : ● None ○ Initialize ○ Refresh ○ Finalize
```

The Db2 Shift program has two methods of moving the database to the destination. The traditional method is to take one pass at the database and move everything at once. During the finishing process, the program will briefly suspend the database (depending on settings) and perform a final refresh. This last step will pick up any database objects that may have changed over the course of the move.

The total amount of time the database is suspended is directly related to how much information has changed during the shift process. If you have large number of changes, the final refresh may too long an outage. To minimize this outage, you can use the synchronization options provided.

The default setting is do a complete shift operation without intermediate sync operations.

To initialize the synchronization option, the first Db2 Shift command will use the sync option.

Syntax: `--sync=start_sync`

```
Sync Options   : ○ None ● Initialize ○ Refresh ○ Finalize
```

This initial step will instruct Db2 Shift to copy all the required database objects to the target system. Once the copies are complete, the program will end processing and leave the target system in an incomplete state. During this process the source database is operational, and it will not be suspended.

When there is an appropriate moment, the database movement can be finalized with the `finish_sync` option.

Appendix A: Options Reference

Syntax: `--sync=finish_sync`

```
Sync Options      :  None  Initialize  Refresh  Finalize
```

The `finish_sync` option will do one final pass against the source database and then it will finalize the database movement on the destination site. During the last pass the finish process will suspend the database to get a consistent database environment. Once this step completes, the destination database will be available.

If the source system has high update volumes, there may be a need to do multiple sync operations to minimize the finalize step. The Db2 Shift command will need to be told that it is syncing the database again, but not to start from scratch. The control files generated by the `--sync=start_sync` option will allow Db2 Shift to move database objects that are new or have changed since the initial synchronization request. The command to do the sync and only refresh the database objects requires the use of the `rerun_sync` option.

Syntax: `--sync=rerun_sync`

```
Sync Options      :  None  Initialize  Refresh  Finalize
```

The `rerun_sync` option indicates to the Db2 Shift program to start syncing process to look for delta changes only.

The `--sync=rerun_sync` process can be run repeatedly until the number of changes between runs is minimized. When an appropriate timeslot is available, the shift can be finalized by using the `finish_sync` option.

Syntax: `--sync=finish_sync`

```
Sync Options      :  None  Initialize  Refresh  Finalize
```

The `finish_sync` option will perform the final pass against the source database and complete the shift process as before.

In summary, the standard shift operation will complete in one step (no `--sync` option is used). The `--sync=start_sync` option allows you to gradually move a database over time. For a gradual database shift, use the `start_sync` option on the first run. This will move an initial set of database files to the target. Then use the `rerun_sync` option on subsequent runs to copy any files that may have been added or changed to the database. When you are ready to finalize the shift, use the `finish_sync` option to gather any remaining files and complete the shift operation on the target system.

Metadata Generation

Syntax: `--blank-slate=[true|false], --gen-settings, --verify-only`

Metadata : ● Refresh ○ None ○ Settings Only ○ Verify

The Db2 Shift command generates metadata that is used during the shift process. This metadata is key to determining which objects need to be moved from a source to destination as well as validating that the source can be successfully moved.

Generating metadata requires access to the source and destination systems. If for some reason the connection to the source or destination is unavailable, the existing metadata files can still be used. In most cases you will not need to adjust these settings unless you have encountered a shift error.

Blank Slate

Syntax: `--blank-slate=true`

Metadata : ● Refresh ○ None ○ Settings Only ○ Verify

The `--blank-slate` option determines whether the existing metadata is refreshed. The default is value for `--blank-slate` is `true` which will display in the GUI as Refresh. The `--blank-slate` option will delete any existing metadata files and recreate them on your system during the shift operation. The default option is `true` which will result in new source and destination settings being generated.

If you are importing settings files from other systems, or if you need to rerun the shift process without regenerating the files, use the `--blank-slate=false` option. When Db2 Shift executes it will use the existing metadata in the working directory and attempt to use those settings.

Syntax: `--blank-slate=false`

Metadata : ○ Refresh ● None ○ Settings Only ○ Verify

One scenario that involves the use of `--blank-slate=false` occurs when a shift operation fails at the target OC/Kubernetes pod because of a communication error. The database at the destination will be left in an inconsistent state and must be rebuilt by the shift process. The settings for the destination database can no longer be retrieved because the database cannot be started. Because of this reason, you must use the existing destination settings that were generated when you first ran the Db2 Shift command.

Generate Settings

Syntax: `--blank-slate=true, --gen-settings`

Metadata : Refresh None Settings Only Verify

The `--gen-settings` (Generate Settings) option is used in conjunction with `--blank-slate`. The use of `--gen-settings` will prevent Db2 Shift from continuing execution after the metadata files have been created.

Verification

Syntax: `--verify-only`

Metadata : Refresh None Settings Only Verify

The `--verify-only` option will generate the metadata files and check the connectivity and all settings and then stop execution. If `--verify-only` completes successfully, the Db2 Shift command will be able to execute the shift process.

Online or Offline Move

Syntax: `--online, --offline`

Database Mode : Online Move Offline Move

By default, the Db2 Shift command will suspend the database while it completes the last scan of the files. During this time, the database will not complete any insert, update, or delete transactions. This will result in a consistent database at the destination, but some transactions will not have been committed to the database. If the database at the destination must be identical to the source database, then you must choose the `--offline` option which will shut down the database so there will not be any transactions outstanding when the shift is completed.

Note: The period that the database is suspended or stopped is dependent on the changes that have occurred in the database from the time the shift operation started to the time when copying of the data is complete. The changes that have occurred during this period needs to be captured during the final step. It is during this step that the database must be suspended or stopped. During the initial scan the database will remain completely online and will not be impacted by the shift utility from a transaction perspective. However, since there is a large amount of disk reads taking place, it may impact buffer pool read performance.

Appendix A: Options Reference

The `--offline` mode must be used when shifting a database that requires a migration from an older release of Db2. This option must be specified when shifting Db2 versions 10.5 or 11.1.

Threading

Syntax: `--threads=[0..8]`

```
Thread count      : 4
```

The copy phase of the Db2 Shift program can use multiple threads to transmit data to a destination. This setting allows you to increase the parallelism up to 8 threads. As you increase the number of threads, the amount of data being transmitted increases, at the expense of greater CPU usage and network congestion. The default value is 4 which strikes a balance between overhead and network performance.

Compression

Syntax: `--compression=[0..9]`

```
Compression      : 4
```

RSYNC compresses the data during the transfer process to allow for faster movement of data. The amount of compression can be adjusted from 0 to 9 with 0 turning off compression and values between 1 and 9 increasing the amount of compression applied to the data. Higher compression values will result in more CPU usage and may not significantly reduce the size of the data stream.

A value of 4 has been found to be a good compromise between compression overhead and data size on slow networks (< 1Gb/s). For high-speed networks, a value of 0 is recommended unless there is a requirement to reduce network traffic.

Stored Procedures and Functions

Syntax: `--exclude-functions`

```
Exclude Routines :  Do not move external routines
```

By default, the Db2 Shift command will automatically move all external stored procedures and functions that are found in the `$HOME/sqllib/function` path to the new destination. All SQLPL and PL/SQL routines are moved as part of the database move, so there is no migration required for those routines.

Use the `exclude` flag to prevent any external functions being moved to the destination.

Overrides

Syntax: `--overrides=""`

```
Overrides      : instance_memory 3468896_-----
```

The overrides field is used to update INSTANCE settings at the target location. All database settings are moved during the shift process, but no INSTANCE settings are changed.

If you need to change any INSTANCE settings, place the name of the parameter, following by the value on this line. Multiple parameters can be placed on the line by separating each parameter/value pair with a semi-colon ;.

B

Appendix

ADDITIONAL RESOURCES FOR DB2

Appendix B: Additional Resources for Db2

International Db2 User Group (IDUG)

IDUG is all about community. Through education events, technical resources, unique access to fellow users, product developers and solution providers, they offer an expansive, dynamic technical support community. IDUG delivers quality education, timely information and peer-driven product training and utilization that enable Db2 users to achieve organizational business objectives, and to drive personal career advancement. Visit idug.org.

Support

For any questions regarding the Db2 Shift tool, including any suggestions, general comments, or bug reports, please contact:

- George Baklarz bak1arz@ca.ibm.com
- Phil Downey phil.downey1@ibm.com

We would also appreciate any feedback on the successful use of the tool.

Thanks for using the Db2 Shift Program!

Phil & George

Db2 Shift

For Linux X64 Installations

Version 2 Usage Guide

George Baklarz and Phil Downey

The Db2 Click to Containerize family encompasses several tools that provide customers with the ability to quickly modernize their Db2 landscape. The Db2 Shift utility can be used to clone a copy of Db2 into an OpenShift, Kubernetes, Cloud Pak for Data (CP4D), or a standard Db2 instance.

The utility is intended to help customers move their existing Db2 databases on Linux into a containerized environment with the minimum amount of effort.

Some benefits of Db2 Shift are:

- Automated, fast, and secure movement of Linux x64 databases to Hybrid Cloud
- Massively reduces time to containerize database workloads
- Enables alignment with Agile Delivery and Project Lifecycle with Cloning capabilities once in the cloud

Features of Db2 Shift include:

- The ability to move a database without the need to unload, export, de-crypt, or backup the database
- Automatic upgrades from Db2 Version 10.5 to the latest version (11.5.8) of Db2
- Shifting of all database settings and objects, including external functions located in the Db2 library path
- Row, Columnar, and Encrypted databases can be moved
- SMP, and MPP databases can be moved (excluding pureScale installations)
- Easy setup of HADR Servers

George Baklarz, B. Math, M. Sc., Ph.D. Eng., has spent 31 years at IBM working on various aspects of database technology. George has written 14 books on Db2 and other database products. You can reach George at baklarz@ca.ibm.com.

Phil Downey, B. CompSci and Psychology, has over 30 years' experience in the IT industry and 25 years' experience working with enterprise database systems across many different industries in several different technical and product management roles. He is experienced in designing and deploying enterprise data architectures and migrating existing applications to them. He is one of the inventors of the Click to Containerise technology. You can reach Phil at phil.downey1@ibm.com.