Delphix DR Playbook

Table of Contents

1. About the document

This document is meant to give instructions on Disaster Recovery readiness and rollout for a Delphix Engine or collection of Delphix Engines used by your organization. The initial layout of the document is meant to be general but can easily be customized to be specific to the operations at your site, and it is promoted to be a living document for your organization. This is not meant to be a replacement for the Delphix documentation available online at docs.delphix.com, and in fact, maybe slightly outdated at times. It is highly advised that the online documentation also be reviewed, and this document updated to reflect any noticed changes. The updating of this document is even more important in the case where proprietary information has been added specifically to your site, as no Delphix-provided updates would contain this proprietary information.

2. Definition of terms

Below is a list of definitions and/or abbreviations used throughout the document, as well as terms used by Delphix Support or Delphix Professional Services when discussing Delphix Engines.

Term	Abbreviation	Description
Delphix Engine	DE	A single virtual machine containing a Delphix installation.
dSource	DS	Also known as a source database or source time flow, this is the representation of the source database which is stored on the Delphix Engine
Virtual Database	VDB	This is a database provisioned from either a dSource or another VDB which is a full read/write copy of the source data
Command Line Interface	CLI	A method by which a Delphix Engine can be accessed using SSH
Application Programming Interface	API	A method by which a Delphix Engine can be accessed programmatically

3. Pre-Requisites for Metadata Backup and Restore

3.1. Pre-requisites for Metadata Backup

Before performing a metadata backup of the Delphix Engine, we need Delphix DxToolkit to be present on the server from where backup commands will execute.

Download latest version of DxToolkit from https://github.com/delphix/dxtoolkit

dxtools.conf must have two entries for the same engine, one with admin user and other with sysadmin user. Sysadmin entry hostname must be in pattern as *<adminHostName>_sys*

An example entry in dxtools.conf file-

```
{
        "data":[
        {
                "hostname" : "engine",
                "ip_address" : "1.1.1.1",
                "username" : "admin",
                "password" : "delphix",
                "port" : "80",
                "default" : "false",
                "encrypted" : "false",
                "timeout" : "60"
        },
        {
                "hostname" : "engine_sys",
                "ip_address" : "1.1.1.1",
                "username" : "sysadmin",
                "password" : "sysadmin",
                "port" : "80",
                "default" : "false",
                "encrypted" : "false",
                "timeout" : "60"
        }
        ]
}
```

3.2. Pre-requisites for Metadata Restore

Before performing a metadata restore of Delphix Engine, below pre-requisites are required-

1. Do not modify backup files, unless you are very much sure about the changes and the process.

2. Configured target Delphix Engine.

3. Delphix DxToolkit on the server where restore commands will execute.

- Download latest version of DxToolkit from https://github.com/delphix/dxtoolkit

4. dxtools.conf must have an entry for the target engine with admin user and the hostname must be the same as what's used during backup/restore.

An example entry in dxtools.conf file-

```
{
        "data":[
        {
                "hostname" : "engine",
                "ip_address" : "12.12.12.12",
                "username" : "admin",
                "password" : "delphix",
                "port" : "80",
                "default" : "false",
                "encrypted" : "false",
                "timeout" : "60"
        }
        ]
}
```

5. Create environment config file, **env_config.csv** (for environments) manually in backup directory location from which restore will be done with the below format.

```
<DelphixEnvName1>,<EnvOSUser>,<OSUserPassword>
<DelphixEnvName2>,<EnvOSUser>,<OSUserPassword>
….
<DelphixEnvName(n)>,<EnvOSUser>,<OSUserPassword>
```

Delphix APIs can't extract the actual Database passwords from the engine, due to security reasons. So, you need to supply one default OS password in backup and restore scripts (variable: *default_os_pwd*). There should be an entry for each environment in env_config.csv file which we extracted, otherwise default OS password will be used.

Before restoring the environment, actual passwords from this file will be updated in environment restore scripts based on Environment Name and OS Username. This is a case-sensitive validation. Make sure Environment Name and OS Username have the same case as in the backup file.

Make sure to verify environment names must be the same as what's extracted or backup up from Delphix Engine. Check **backup_env.txt** file for validating environment name and environment OS user under location, *<backupDirectory>/environments*.

Windows Domain\Username must be in below format.
<WindowsEnvName>,<Domain>\\\\<EnvOSUser>,<OSUserPassword>

Example env_config.csv entries:

```
WindowsTarget,delphix\\\\delphix_trgt,delphix
WindowsSource,delphix\\\\delphix_src,delphix
LinuxSource,delphix,delphix
LinuxTarget,delphix,delphix
```

Initially, we need to create this file manually which can be reused for any further restores.

6. Create database config file, **db_config.csv** (for dSources) manually in backup directory location from which restore will be done with the below format.

   ```
   <dSourceName1>,< DelphixEnvName1>,<DBUserName>,<DBUserPassword>
   <dSourceName2>,< DelphixEnvName2>,<DBUserName>,<DBUserPassword>
   ….
   <dSourceName(n)>,< DelphixEnvName(n)>,<DBUserName>,<DBUserPassword>
   ```

   Delphix APIs can't extract the actual Database passwords from the engine, due to security reasons. So, you need to supply one default DB password in backup and restore scripts (variable: *default_db_pwd*). There should be an entry for each database in the db_config.csv file which we extracted, otherwise the default DB password will be used.

   Before restoring dSources, actual passwords from this file will be updated in dSource restore scripts based on Database Name, Environment Name, and Database Username. This is a case-sensitive validation. Make sure Database Name, Environment Name, and Database Username have the same case as in the backup file.

   Make sure to verify database names, environment names and DB usernames must be the same as what's extracted or backup up from Delphix Engine. Check **backup_metadata_dsource.txt** file for validating all the fields user under location, *<backupDirectory>/db_objects*.

   If the source is Oracle Multi-tenant DB, have one entry for container user and one for the root user for each dSource.

   Example db_config.csv entries for Oracle MT:

   ```
   ORAPDB1,LinuxSource,c##delphixdb,delphixdb
   ORAPDB2,LinuxSource,c##delphixdb,delphixdb
   ORAPDB1,LinuxSource,delphixdb,delphixdb
   ORAPDB2,LinuxSource,delphixdb,delphixdb
   ```

   Initially, we need to create this file manually which can be reused for any further restores.

7. *(Optional)* If using a Oracle Multi-Tenant physical container to provision virtual pluggable databases on the target server, then this file is needed, otherwise skip this file.

Create vdb config file, **vdb_config.csv** (for VDBs) manually in backup directory location from which restore will be done with below format.

```
<PhysicalContainer1>,< DelphixEnvName1>,<CDBUserName>,<CDBUserPassword>
< PhysicalContainer1>,< DelphixEnvName2>,<CDBUserName>,<CDBUserPassword>
….
< PhysicalContainer1(n)>,< DelphixEnvName(n)>,<CDBUserName>,<CDBUserPassword>
```

Delphix APIs can't extract the Physical Container Database passwords from engine, due to security reasons. So, you need to supply one default DB password in backup and restore scripts (variable: *default_db_pwd*). There should be an entry for each physical container database in vdb_config.csv file which we extracted, otherwise default DB password will be used.

Before restoring VDBs, a physical container from this file will be updated in VDB restore scripts based on Container Database Name, Environment Name, and Container Database Username. This is a case-sensitive validation. Make sure Container Database Name, Environment Name and Container Database Username have the same case as in the backup file.

Make sure to verify container database names, environment names and container DB usernames must be the same as what's extracted or backup up from Delphix Engine. Check **backup_metadata_vdb.txt** file for validating all the fields user under location, *<backupDirectory>/db_objects*.

Example vdb_config.csv entries for Oracle MT:

```
CDB1,LinuxTarget1,c##delphixdb,delphixdb
CDB1,LinuxTarget2,c##delphixdb,delphixdb
```

Initially, we need to create this file manually which can be reused for any further restores.

8. Auto Discovery Requirements must be configured for all environments.

9. dxtools.conf in dxtoolkit directory must have same hostname for admin user which was used during backup.

4. Encrypt dxtools.conf file in dxtoolkit

To encrypt dxtools.conf file in dxtoolkit, first creates a file dxtools.conf.plain with plain passwords.

Execute the command below-

```
dx_encrypt -plainconfig dxtools.conf.plain -encryptedconfig dxtools.conf
```

5. Backup Metadata

Script Name: backup_dlpx_metadata.sh

Execution Command: ./backup_dlpx_metadata.sh <engineName>

Each Execution of script will create backup directory in below format –
<engineName>-metadata-backup-<date>-<time>

Under backup directory, multiple directories get created for a specific task.

Below is an explanation on directories structure present under backup directory-

sys_config

This directory contains information about all system configuration with the below files.

- **sys_config.csv**: This file contains system configuration including LDAP info, SMTP info, etc, present under the sysadmin portal.
- **appliance_info.csv**: This file contains information about engine version, CPU, memory, and object number.
- **network_latency.csv**: This file contains information about network latency tests present on the engine.
- **network_throughput.csv**: This file contains information about network throughput tests present on the engine.

users

This directory contains information about all Delphix users on Delphix Engine, with the below files.

- **users.csv**: CSV file contains information about all users present on Delphix Engine.
- **profile.csv: CSV** file contains information about users and profile mappings present on Delphix Engine.

environments

This directory contains information about all environments on Delphix Engine, with the below files.

- **backup_env.txt**: A text file having dxtoolkit commands to create environments.
- **backup_env.sh**: executable script which will create environments during restore.

db_objects

This directory contains information about all database objects including dSources and VDBs, with the below files.

- **backup_metadata_dsource.txt**: A text file having dxtoolkit commands to create dSources.

- **backup_metadata_dsource.sh**: an executable script which will create dSources during restore.
- **backup_metadata_vdb.txt**: A text file having dxtoolkit commands to create VDBs.
- **backup_metadata_vdb.sh**: an executable script that will create VDBs during restore.
- **<dbname>.dbhooks**: Multiple files are created for each database hooks attached to the database object.

hook_templates

This directory contains information about all Hook Operation templates, with below files.

- **<hookTemplate>.opertemp**: Multiple files are created for each hook operation template.

config_templates

This directory contains information about all VDB Config templates, with the below files.

- **<configTemplate>.template**: Multiple files are created for each VDB Configuration template.

ss_objects

This directory contains information about all Self-Service objects, with the below files.

- **backup_selfservice_templates.txt**: A text file having dxtoolkit commands to create self-service templates.
- **backup_selfservice_containers.txt**: A text file having dxtoolkit commands to create self-service containers.
- **backup_selfservice_templates.sh**: An executable script which will create self-service templates during restore.
- **backup_selfservice_ containers.sh**: an executable script which will create self-service containers during restore.

policies

This directory contains information about all policies, with below files.

- **<policyName>.policy**: Multiple files are created for each policy.
- **policy.mapping**: mapping file which will map policies to DB objects.

5.1. Backup Process Sequence

The process of backup process is ordered in the below sequence.

1. Create backup directory in format, <engineName>-metadata-backup-<date>-<time>
2. Create sys_config directory in backup directory, and export system configuration in required files.
3. Create users directory in the backup directory, and export the user info in the required files.
4. Create policies directory in the backup directory and export the policies info in required files.
5. Create config_templates directory in the backup directory and export the vdb configuration templates info in the required files.
6. Create hook_templates directory in the backup directory and export the hook operation templates info in the required files.
7. Create environments directory in the backup directory and export the environments templates info in required files.
8. Create db_objects directory in the backup directory and export the database objects (dSources/VDBs) info in the required files.
9. Create ss_objects directory in the backup directory and export the self-service objects (templates/containers) info in required files.

5.2. Required Updates in Backup Script as per Environment

Before executing the backup script, do the changes in the below section of the script as per the environment. You will find this section at the starting of the script.

```
###### change below variable values as per your environment #######

dxtoolkit_dir="<path of DxToolkit Directory>"
backup_path="<path where backup directory gets created>"
default_os_pwd="<default OS password to use, if entry not present in env_config.csv file>"
default_db_pwd="<default DB password to use, if entry not present in db_config.csv and
                        vdb_config.csv file>"

####### no changes required below this point ##########
```

6. Restore Metadata

Script Name: restore_dlpx_metadata.sh

Execution Command: ./restore_dlpx_metadata.sh <engineName> <backupDirectoryName>

Execution of script will read the data present under the backup directory, perform manipulation on backed up data to make it ready for restore, and then restore the metadata.

6.1. Restore Process Sequence

The process of the restore process is ordered in the below sequence.

1. Read backup directory passed to restore script.
2. Read users directory in the backup directory and create the users on the Delphix Engine.
3. Read environments directory in the backup directory and create the environments.
4. Read db_objects directory in the backup directory and create the dsources.
5. Read config_templates directory in the backup directory and create the vdb configuration templates.
6. Read hook_templates directory in the backup directory and create the hook operation templates.
7. Read db_objects directory in the backup directory and create the virtual database.
8. Read policies directory in backup directory and create, assign and update policies.
9. Read ss_objects directory in the backup directory and create the self-service objects, templates, and containers.

6.2. Required Updates in Restore Script as per Environment

Before executing the restore script, do the changes in the below section of script as per the environment. You will find this section in the starting of script.

```
###### change below variable values as per your environment #######

dxtoolkit_dir="<path of DxToolkit Directory>”
backup_path="<path where backup directory gets created>"
env_config="<path>/env_config.csv"
db_config="<path>/db_config.csv"
# (Optional) Only required for Oracle Multitenant VDBs using physical containers on target server
vdb_config="<path>/vdb_config.csv"
default_os_pwd="<default OS password to use, if entry not present in env_config.csv file>"
default_db_pwd="<default DB password to use, if entry not present in db_config.csv and
                        vdb_config.csv file>"

####### no changes required below this point ##########
```

7. Logging

Redirect the output of scripts in logfile to keep the logging.

Example of logging backup output

```
./backup_dlpx_metadata.sh engine1 > backup.log &
```

Tail the log file to monitor the progress

```
tail -f backup.log
```

Example of logging restore output

```
./restore_dlpx_metadata.sh engine1 engine1-metadata-backup-03192020-163607 > restore.log &
```

Tail the log file to monitor the progress

```
tail -f restore.log
```

8. Encrypt & Decrypt csv files

As csv files (env_config.csv, db_config.csv, vdb_config.csv) created in Step 3.2 are pre-requisites for Metadata Restore contain passwords for environments and databases, it is recommended to encrypt the files for security, when not using them.

GnuPG(gpg) utility can be used to encrypt and decrypt files. This is present on most Linux distributions. Check the gpg availability and version by executing *gpg --version*

Encrypt CSV files

1. Encrypt env_config.csv

```
gpg -c env_config.csv

# this will ask for password and create encrypted file, env_config.csv.gpg

# remove uncrypted file

rm env_config.csv
```

2. Encrypt db_config.csv

```
gpg -c db_config.csv

# this will ask for password and create encrypted file, db_config.csv.gpg

# remove uncrypted file

rm db_config.csv
```

3. Encrypt vdb_config.csv

```
gpg -c vdb_config.csv

# this will ask for password and create encrypted file, vdb_config.csv.gpg

# remove uncrypted file

rm vdb_config.csv
```

Decrypt CSV files

1. Decrypt env_config.csv

```
gpg -d env_config.csv.gpg > env_config.csv

# this will ask for password and create decrypted file, env_config.csv
```

2. Decrypt db_config.csv

```
gpg -d db_config.csv.gpg > db_config.csv

# this will ask for password and create decrypted file, db_config.csv
```

3. Decrypt vdb_config.csv

```
gpg -d vdb_config.csv.gpg > vdb_config.csv

# this will ask for password and create decrypted file, vdb_config.csv
```

9. Limitations

Consider few limitations listed below with this DR approach

1. Rebuild metadata and objects from scratch. No old data or snapshots will be restored.
2. Any conflicts while restoring the object will get skipped, and will not replace the object.
3. Masked VDBs are not supported by the restoration.
4. The Parent-Child sequence is not maintained while backup objects.
   - Manual Workarounds
     - Manually arrange the object creation scripts in <backupDir>/environment/backup_env.sh
     - Manually arrange the object creation scripts in <backupDir>/db_objects/backup_metadata_vdb.sh
   - Re-run restore script few times, until all objects are created.
5. Backup and Restore scripts are currently tested and certified with Oracle and MSSQL database.
6. Backup and Restore scripts are currently tested and certified on Linux platform.
7. Restore is done serially.