Data Services

Database Maintenance Utility Guide

Version 11

Document Number: DSDMU100.003

Document Title: Data Services Database Maintenance Utility Guide
Document Release Date: December 2023
Document Number: DSDMU100.003

Published by:
Research & Development
meshIQ
88 Sunnyside Blvd, Suite 101
Plainview, NY 11803

Acknowledgements: The following terms are trademarks of meshIQ in the United States or other countries or both: AutoPilot, AutoPilot M6, M6 Web Server, M6 Web Console, M6 for WMQ, MQControl, Navigator, XRay.

Apache®, Apache Solr, Solr™, and its logo are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries. No endorsement by The Apache Software Foundation is implied by the use of these marks.

Other company, product, and service names may be trademarks or service marks of others.

Contents

Chapter 1: Introduction

The Data Services Database Maintenance Utility provides tasks for managing Data Services SolrCloud clusters. There are generally four categories of operations that it supports:

1. Backup/Restore*
2. Import/Export*
3. Migration
4. Purging

* For the operations with asterisks, unless a collection is specified, apply to all non-reference Solr collections. (Non-reference collections are those that do not start with "jkoolref.")

The general format for invoking the Data Services Database Maintenance Utility is:

```
jkool-db-maint.sh <operation> <operation_specific_args>
```

1.1 How this Guide is Organized

Chapter 1: Introduction to the guide

Chapter 2: Backup/Restore. Usage and Examples of the Backup and Restore features. This includes instructions for how to inquire about the status of the backup or restore processes while they are still running in Solr, and how to delete status records.

Chapter 3: Import/Export. Usage and Examples of the Import and Export features.

Chapter 4: Migration. Usage and examples of the Migration feature. This also covers the Compare feature, which shows document counts for the source and destination repositories.

Chapter 5: Purge. Usage and examples of the Purge feature.

1.2 History of this Document

Table 1-1. Document History

Release Date	Document Number	Version	Summary
November 2022	XRDMU100.001	1.4	The first version of this document covers six operations that can be performed by the XRay Database Maintenance Utility: Backup, Restore, Import, Export, Migration, and Purge.
December 2022	XRDMU100.002	1.5	Due to upgrade from Solr 6 to Solr 8, the BACKUP and RESTORE operations have changed from complete snapshots to incremental backups. You no longer need to delete collections prior to restoring them. The BACKUP operation now includes a maxbackups option to limit the number

Table 1-1. Document History			
Release Date	**Document Number**	**Version**	**Summary**
			of backups. The RESTORE operation now includes a backupid option to specify which backup to restore.
December 2023	DSDMU100.003	11	meshIQ Platform branding and label changes.

1.3 User Feedback

meshIQ encourages all users and administrators to submit comments, suggestions, corrections, and recommendations for improvement for all documentation. Please send your comments via e-mail to: support@meshiq.com. You will receive a response, along with the status of any proposed change, update, or correction.

1.4 Release Notes

For release notes for the meshIQ Platform, see the online *meshIQ Platform release notes* page of the Resource Center.

1.4 Intended Audience

This guide is intended for systems administrators and operating engineers responsible for the installation and administration of the meshIQ Platform Data Services environment.

1.5 Technical Support

Use one of the following methods for technical support:

- **Call:** 800-963-9822 ext. 1
  If you are calling from outside the United States: 001-516-801-2100
- **Email:** *mysupport@meshiq.com*
- **Resource center:** *http://customers.meshiq.com*
- **Automated support system:** *https://mysupport.meshiq.com/* (user ID and password required)

Chapter 2: Backup/Restore

The Data Services Backup and Restore processes are intended to support either backing up a cluster (or part of a cluster) to restore it into the same cluster, or making a copy of a cluster to restore it into a different cluster, possibly as part of a disaster recovery process.

Backup and Restore are based on Solr's BACKUP and RESTORE Collection API methods. The Solr methods operate without regard to the concepts of organizations, repositories, volumes, etc. They simply take a backup of a single collection in its entirety from a single cluster or restore a single collection to a single cluster. You cannot apply filters to control what is backed up or what is restored. If you are using volumes, each volume must be backed up and restored separately, since in this context, the volumes are just independent Solr clusters.

Since Backup and Restore are using Solr's Collection API, their functionality is limited to the functionality provided by the Solr API. See the SolrCloud Collections API documentation for more details. In particular, the following limitations apply:

- As mentioned, there is no way to filter what is backed up or restored. The entire collection is backed up or restored.
- Solr runs these operations on each node in the cluster and requires that each node write its data to the same folder. This folder *must* already exist. Solr will *not* create it. (Solr will, however, create subfolders.)
- The backup/restore folder must be on a shared network file system.
- The network folder *must be accessible from every Solr node*, since Solr, *not* the Data Services Database Maintenance Utility, is writing to it.
- Starting with XRay 1.5, the version of Solr used with XRay has been upgraded from Solr 6 (used in XRay versions 1.4 and earlier) to Solr 8, and there have been configuration and format changes to the BACKUP and RESTORE operations:
  - The format of the Solr backups has changed from complete snapshots to incremental backups. As a result, the format of the backups has changed. The Solr 8 RESTORE recognizes the older format, but since the structures of the XRay Solr collections were changed in XRay 1.5, the XRay 1.4 backups should *not* be loaded into XRay 1.5. The XRay 1.5 BACKUP operation only supports Solr incremental backups.
  - By default, Solr *requires* that the path to the shared file system be relative to SOLR_HOME. So, one of the following changes needs to be made. These examples use `/SolrBackups` as the root of the shared file system:
    - In solr.xml, set the <allowPaths> element to the root path to the shared file system. For example:

      ```
      <str name="allowPaths">${solr.allowPaths:/SolrBackups}</str>
      ```

    - Create a symbolic link within $SOLR_HOME to the shared file system. For example:

      ```
      ln -s /SolrBackups ${SOLR_HOME}/SolrBackups
      ```

- Solr's BACKUP not only saves the data, but the collection schema as well, and RESTORE will create the schema before loading the data into it if it does not exist. This behavior is different than that of Solr 6, where the collection *could not* exist. Now, if you are restoring a collection back into the

same cluster the backup was made from, there is no need to delete the collection prior to restoring it. However, if the collection *does* exist, you cannot change the replication or shards values (so you should not include those parameters in the RESTORE operation). In addition, the number of replicas and shards in the backup must match the collection's current configuration.

Solr's BACKUP and RESTORE Collection API methods support the ability to run asynchronously, which is desirable when collections are large, as it could be quite time-consuming to back up or restore. Although the script will be completed immediately, the backups within Solr will still be running. Solr will maintain the status of the asynchronous requests, allowing progress to be monitored. To monitor asynchronous backups and restores, there are two additional related Solr Collection API methods: REQUESTSTATUS and DELETESTATUS.

IMPORTANT!

If you have requested the BACKUP or RESTORE be done asynchronously, then when the BACKUP or RESTORE process has completed, you must use DELETESTATUS to manually delete the saved progress status information. Solr does not delete the status information automatically.

As the name implies, Solr's Collection API methods operate on individual collections. You could use Solr's API directly, but you would have to issue the method calls for each collection separately. If you were to request that the Solr operations be done asynchronously, you would have to maintain the resulting asynchronous request IDs to later query for the status and to remove those saved statuses. The Data Services Database Maintenance Utility puts a layer on top of this, issuing the individual Collection API calls for each specified collection (or all non-reference collections, if none are specified). For operations run asynchronously, the Data Services Database Maintenance Utility records all the request IDs into a single file, thus allowing the set of Solr API method calls to be treated as a single Data Services backup/restore/status operation.

To accomplish this, when doing a Backup, you give the backup a name, which allows the Data Services Database Maintenance Utility to collect all the related Solr backup files and identify them as being part of the same Database Maintenance operation. This backup name does not directly equate to the backup name as documented in the Solr BACKUP and RESTORE Collection API but is used to form it.

You can either use a single Backup operation to back up all collections, or you can use multiple operations, with each operation doing a subset of the Solr collections. If you choose to use multiple operations, you can still consider this group to be a single backup by using the same backup name. But for a given backup name, there should be only one backup process running at a time per collection. So, whether using single or multiple backups, make sure that there is only one backup actively taking place per Solr collection.

NOTE

Unlike the other Database Maintenance operations, the Backup/Restore uses only Solr API methods. As a result, meshIQ Track users (for example, Administrator) are not authenticated. If Solr authentication is enabled, it is done.

2.1 Usage

This section covers the commands to back up and restore collections as well as commands to inquire about the status of these processes. Since the status records must be cleared, instructions for deleting those records are also included.

2.1.1 Backup

The full syntax for running a backup is:

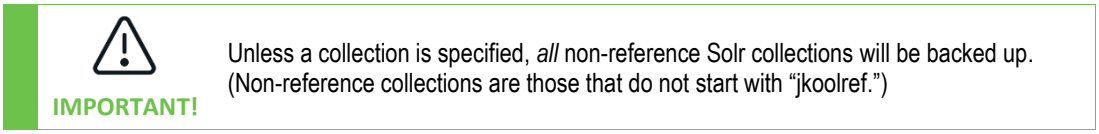
```
jkool-db-maint.sh -backup -src:<src_db_url> -f:<shared_folder> -name:<backup_name>
            [ -tables:<db_tables> ] [ -async ] [ -qt:<backup_threads> ]
            [ -maxbackups:<max_backup_points> ]
            [ -srcuser:<src_user> -srcpwd:<src_pwd> ]
```

`-src`: Defines the Solr URL from which to initiate the backup. This is just one of the Solr nodes in the cluster. The backup request(s) will be sent to all nodes in the cluster.

`-f`: Defines the folder where backup files are written. As mentioned above, this must be an *existing folder* on a shared network file system, and it must be accessible from *all* Solr nodes.

`-name`: Defines the name of the backup operation. If a backup with this same name already exists, it will be overwritten. (You will *not* be prompted to replace it, as Solr just writes the backups.)

`-tables`: This can be used to specify which specific database "tables" (that is, Solr collections) to back up.



`-async`: When you specify this option, the backups are done asynchronously. The request IDs returned by Solr are stored in a file, and the name of this file will be printed to the console and logged to the debug log file (each invocation of the maintenance script will generate a unique file name). You will need to specify this file when executing `-status` and `-delstatus` operations (see below). If the option is not specified, then backups are done synchronously.

`-qt`: This specifies how many threads to use to perform the backup operations. One backup "operation" is a backup request for a specific Solr collection. Using a value greater than one allows that number of backup operations to be run in parallel. When a collection backup is complete, the thread doing the backup will go on to the next collection in the queue. This is mainly useful when doing synchronous backups. Asynchronous backups simply send a request to Solr and are completed once Solr returns that it accepted the request, so there is no real need to multithread these.

`-maxbackups`: Defines the number of incremental backup points to maintain. If there are more than this number, the older ones are removed. If this option is omitted, the BACKUP operation keeps all prior backup points. The `-maxbackups` option maps directly to the **maxNumBackupPoints** Solr BACKUP API parameter.

`-srcuser`: Solr username to use to log into Solr.

`-srcpwd`: Solr password to use to log into Solr.

2.1.2 Restore

The full syntax for running a restore is:

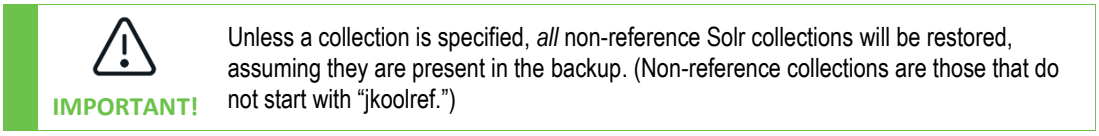
```
jkool-db-maint.sh -restore -dest:<dest_db_url> -f:<shared_folder> -name:<backup_name>
            [ -tables:<db_tables> ] [ -async ] [ -wt:<restore_threads> ]
            [ -backupid:<backup_id> ]
            [ -repl:<repl_factor> ] [-nodeshards:<shards_per_node> ]
            [ -destuser:<dest_user> -destpwd:<dest_pwd> ]
```

`-dest`: Defines the Solr URL to restore the collection(s) into. This is just one of the Solr nodes in the cluster. The restore request(s) will be sent to all nodes in the cluster.

`-f`: Defines the folder where restore files are read from. As mentioned above, this must be an *existing folder* on a shared network file system, and it must be accessible from all Solr nodes.

`-name`: Defines the name of the backup operation to restore.

`-tables`: This can be used to specify which specific database "tables" (that is, Solr collections) to restore.



`-async`: When you specify this option, the restores are done asynchronously. When this option is used, the request IDs returned by Solr are stored in a file, and the name of this file will be printed to the console and logged to the debug log file (each invocation of maintenance script will generate a unique file name). You will need to specify this file when executing `-status` and `-delstatus` operations (see below). If the option is not specified, then restores are done synchronously.

`-wt`: This specifies how many threads to use to perform the restore operations. A restore "operation" is a restore request for a specific Solr collection. Using a value greater than one allows this many restore operations to be run in parallel. When a collection restore is complete, the thread doing the restore will go on to the next collection in the queue. This is mainly useful when doing synchronous restores. Asynchronous restores simply send a request to Solr and are completed once Solr returns that it accepted the request, so there is no real need to multithread these.

`-backupid`: Indicates the specific backup point to restore. When this option is omitted, the latest backup point is restored. The `-backupid` option maps directly to the **backupId** Solr RESTORE API parameter.

`-repl`: This overrides the default replication, which is the replication factor the collection had when the backup was generated.

`-nodeshards`: This overrides the default maximum shards per node value, which is the value factor the collection had when the backup was generated.

`-destuser`: Solr username to use to log into Solr.

`-destpwd`: Solr password to use to log into Solr.

2.1.3 Inquire Status (Asynchronous Backup or Restore)

The full syntax for inquiring about the status of an asynchronous backup or restore operation is:

```
jkool-db-maint.sh -status -sf:<request_id_file>
            { -src:<src_db_url> [ -srcuser:<src_user> -srcpwd:<src_pwd> ]
              | -dest:<dest_db_url> [ -destuser:<dest_user> -destpwd:<dest_pwd> ] }
```

You can specify either `-src` or `-dest`. When doing so, use the same Solr URL you used in the backup or restore operation.

`-sf`: This is the name of the Solr request ID file that is generated by the backup or restore operation. As mentioned above, this file name is printed by the backup or restore operation to the console and log file.

2.1.4 Delete Status (Asynchronous Backup or Restore)

The full syntax for deleting the status of an asynchronous backup or restore operation is:

```
jkool-db-maint.sh -delstatus [ -sf:<request_id_file> ]
            { -src:<src_db_url> [ -srcuser:<src_user> -srcpwd:<src_pwd> ]
              | -dest:<dest_db_url> [ -destuser:<dest_user> -destpwd:<dest_pwd> ] }
```

You can specify either `-src` or `-dest`. When doing so, use the same Solr URL you used in the backup or restore operation.

`-sf`: This is the name of the Solr request ID file that is generated by the backup or restore operation. As mentioned above, this file name is printed by the backup or restore operation to the console and log file. If you leave this out, *all* saved Solr asynchronous operation status records are deleted.

2.2 Examples

The `-backup` and `-restore` examples below assume that `/SolrBackups` was added to `allowPaths` in solr.xml, as described in the introduction to *Chapter 2: Backup/Restore*. Therefore, the `-f:` option shown below includes a leading "/". If instead the option for creating a symbolic link within $SOLR_HOME were chosen, then the leading "/" in the -f: option would be left off.

Here is an example of using the backup/restore to back up the set of tables that will be upgraded when you upgrade a Data Services installation. This example assumes access to a shared file system mounted at `/SolrBackups`, and that the subfolder `MyBackups` has already been created):

```
jkool-db-maint.sh -backup
                  -src:http://<solr-host>:8983
                  -f:/SolrBackups/MyBackups
                  -name:MyUpgradeBackup
                  -tables:jkooladmin.registeredusers,jkooladmin.organization,
                          jkooladmin.repositories,jkooladmin.accesstokens,
                          jkool.dictionaries
```

This backs up the necessary administrative collections (users, org, repos, tokens) along with the UI definitions (dashboards, viewlets, user settings) that will be upgraded.

Behind the scenes, individual Solr BACKUP requests are issued, one for each of the specified collections. These will create folders within `/SolrBackups/MyBackups` for each collection with the name `MyUpgradeBackup-<collection-name>` (for example, `MyUpgradeBackup-jkooladmin.registeredusers`). If you need to equate these to the Solr BACKUP parameters:

`-f`: Argument value is the Solr BACKUP **location** parameter (for example, `/SolrBackups/MyBackups`)

`-name`: Argument value combined with "`-<collection-name>`" is the Solr BACKUP **name** parameter (for example, `MyUpgradeBackup-jkooladmin.registeredusers`)

Now, if you need to restore these collections, run the following restore:

```
jkool-db-maint.sh -restore
                  -dest:http://<solr-host>:8983
                  -f:/SolrBackups/MyBackups
                  -name:MyUpgradeBackup
                  -tables:jkooladmin.registeredusers,jkooladmin.organization,
                          jkooladmin.repositories,jkooladmin.accesstokens,
                          jkool.dictionaries
```

Chapter 3: Import/Export

Import/Export allows records from one or more collections to be written to and loaded from a CSV file. Unlike Backup/Restore, you have some basic filtering ability to choose which records are exported. Also, Import/Export will require you to authenticate using the meshIQ Track Administrator user.

You may have noticed that Import/Export operations are also available via the jKQL Command Line Utility (`jkool-cmd`), and the functionality of the Data Services Database Maintenance Utility is very similar. However, the Data Services Database Maintenance Utility and the jKQL Command Line Utility view the information from different points of view. In particular, the jKQL Command Line Utility is repository-centric. This means that when operating on items whose records can only belong to a specific repository, like streamed data or non-organization, non-admin table entries, the operations are performed on that specific repository only. The Data Services Database Maintenance Utility is Solr collection-centric, meaning that it operates on Solr collections as a whole, regardless of the repositories to which the data in those collections belongs.

3.1 Usage

This section covers export and import of collections.

3.1.1 Export

The full syntax for running an export is:

```
jkool-db-maint.sh -export -src:<src_db_url>
            -f:<folder> -name:<backup_name> -P:<jkool_pwd>
            [ -tables:<db_tables> ] [ -dates:<date_filter> ]
            [ -qb:<query_block_size> ] [ -qt:<export_threads> ]
            [ -srcuser:<src_user> -srcpwd:<src_pwd> ]
            [ -C:<db_url> [ -UD:<server_user> -PD:<server_pwd> ] ]
            [ -mr:<max_retries> ] [ -t:<solr_timeout> ]
```

`-src`: Defines the Solr URL from which to initiate the export. This is just one of the Solr nodes in the cluster. The export will export data from all nodes in the cluster.

`-f`: Defines the folder where export files are written. Unlike Backup, this is a local file system folder where export files will be written. (One file is written per collection.)

`-name`: Defines the name of the export operation. If an export with this same name already exists, it will be overwritten. (You will *not* be prompted to replace it.)

`-tables`: This can be used to specify which specific database "tables" (that is, Solr collections) to export. If this option is not specified, then all non-reference Solr collections will be exported.

`-dates`: This specifies a date filter, as a jKQL Date Filter expression, which will limit the streaming-related data to be exported.

`-qb`: This specifies how many records to retrieve in a single query operation. The data for each collection is queried for in blocks of this many records, until all data is retrieved.

`-qt`: This specifies how many threads to use to perform the export operations. An export "operation" is an export for a specific Solr collection. Using a value greater than one allows this many export operations

to be run in parallel. When a collection export is complete, the thread doing the export will go on to the next collection in the queue.

`-P`: This is the meshIQ Track Administrator user password.

`-srcuser`: Solr username to use to log into Solr.

`-srcpwd`: Solr password to use to log into Solr.

`-C`: This is the Solr URL for the master Solr cluster. If not specified, the cluster from `-src` is assumed to also be the master cluster.

`-UD`: Solr username to use to log into master Solr cluster.

`-PD`: Solr password to use to log into master Solr cluster.

`-mr`: Maximum number of times to retry a failed Solr request.

`-t`: Timeout, in milliseconds, to use for Solr requests. If the Solr cluster is large and has large collection sizes, you may need to increase this value.

3.1.2 Import

The full syntax for running an import is:

```
jkool-db-maint.sh -import -dest:<dst_db_url>
            -f:<folder> -name:<backup_name> -P:<jkool_pwd>
            [ -tables:<db_tables> ] [ -wb:<write_page_size> ] [ -wt:<import_threads> ]
            [ -destuser:<dest_user> -destpwd:<dest_pwd> ]
            [ -C:<db_url> [ -UD:<server_user> -PD:<server_pwd> ] ]
            [ -mr:<max_retries> ] [ -t:<solr_timeout> ]
```

`-dest`: Defines the Solr URL into which to import the data. This is just one of the Solr nodes in the cluster. The import will import data across all nodes in the cluster.

`-f`: Defines the folder where export files to import are read from. Unlike Restore, this is a local file system folder where export files will be read from.

`-name`: Defines the name of the export operation. If an export with this same name already exists, it will be overwritten. You will *not* be prompted to replace it.

`-tables`: This can be used to specify which specific database “tables” (that is, Solr collections) to export. If this option is not specified, then *all* Data Services Solr collections will be exported.

`-wb`: This specifies how many records to include in a single Solr document index (write) operation.

`-wt`: This specifies how many threads to use to perform the import operations. An import “operation” is an import for a specific Solr collection. Using a value greater than one allows this many import operations to be run in parallel. When a collection import is complete, the thread doing the import will go on to the next collection in the queue.

`-P`: This is the meshIQ Track Administrator user password.

`-destuser`: Solr username to use to log into destination Solr.

`-destpwd`: Solr password to use to log into destination Solr.

`-C`: This is the Solr URL for the master Solr cluster. If not specified, the cluster from `-dest` is assumed to also be the master cluster.

`-UD`: Solr username to use to log into master Solr cluster.

`-PD`: Solr password to use to log into master Solr cluster.

`-mr`: Maximum number of times to retry a failed Solr request.

`-t`: Timeout, in milliseconds, to use for Solr requests. If the Solr cluster is large and has large collection sizes, you may need to increase this value.

Chapter 4: Migration

Migration was designed to move a repository from one volume to another (one Solr cluster to another). Unlike the features above, this one is completely repository-centric. Note that all meshIQ Track admin, reference, and dictionary (dashboard/viewlet) records are not tied to a single repository; this data must reside in master cluster. As a result, it is not migrated.

Migration can be thought of as a parallel export and import, querying for records from one Solr cluster and writing them to another Solr cluster. As queries are completed, the result records are queued up for writing, and the writes are happening in parallel with the queries, unlike exports and imports, which occur separately: all the data is first exported, and then imported.

The migration is done by taking the range of dates covered by the data in the repository being migrated and partitioning them into date blocks. (The length of time for each block is configurable; by default, it is 7 days.) The output from the migration process will indicate the date range being processed. If there is a problem, and migration terminates prematurely, you can use the output to find the last date block that was completed successfully and restart the migration. In this case, start with a date on or after the last successful date range, to avoid migrating data that was already migrated. Although running multiple migrations with overlapping dates/times will not cause problems (data that was already moved will simply just overwrite the existing records), by starting the new migration where the first migration left off, you can avoid wasting time and resources on data that has already been migrated.

There is a related command, Compare (`-compare`), that can be used to verify the results of the migration by showing the document counts for the repository for each Solr collection, highlighting any differences.

The process of migrating a repository is as follows:

- Initiate a migration, moving some or all of the data from its current cluster to the new cluster for the repository.
- When migration is complete (you may have to wait several minutes for newly written data to be indexed by Solr), you can use COMPARE to compare the document counts to verify that number of documents in original cluster matches that in the new cluster. If data was actively being streamed when the migration was initiated, you will have to limit the days that the count is being applied to. For example, comparing the two volumes before the date on which migration was initiated, using a date filter of "before <migration-start-date>", for example: `-dates:before xxx`). This step is not necessary, but it may help you to verify that all data was moved.
- When migration is complete, update the repository definition to specify the name of the volume using the new cluster.
- Check the CEP log file for an indication that the repository has been switched over to the new cluster. (If you are using multiple CEP engines, check the log file for the CEP that's running the Gateway.) Log message will be one of the following:
  - `Moving registration of repository rrr from volume abc to volume xyz`
  - `Registering repository rrr with volume xyz`
- Once the migration is complete and the repository has been actively associated with the new volume, rerun the migration using a date filter of "since <migration-start-date>" (`-dates:since xxx`), to pick up any new data that was streamed in but was missed by the original migration.

4.1 Usage

This section covers the migration of a repository and includes instructions for checking to ensure that all data has been migrated.

4.1.1 Migration

The full syntax for running a migration is:

```
jkool-db-maint.sh -migrate -repo:<repoid> -P:<jkool_pwd>
            -src:<src_db_url> [ -srcuser:<src_user> -srcpwd:<src_pwd> ]
            -dest:<dst_db_url> [ -destuser:<dest_user> -destpwd:<dest_pwd> ]
            [ -items:<jkql_items> ] [ -dates:<date_filter> ]
            [ -qd:<days_per_partition> ] [ -qb:<query_block_size> ]
            [ -wb:<write_page_size> ] [ -pw:<pending_writes> ]
            [ -qt:<query_threads> ] [ -wt:<write_threads> ]
            [ -C:<db_url> [ -UD:<server_user> -PD:<server_pwd> ] ]
            [ -mr:<max_retries> ] [ -t:<solr_timeout> ]
```

`-repo`: The complete repository ID to migrate.

`-src`: Defines the Solr URL to migrate the data from. This is just one of the Solr nodes in the cluster. The migrate feature will move data from all nodes in the cluster.

`-srcuser`: Solr username to use to log into source Solr.

`-srcpwd`: Solr password to use to log into source Solr.

`-dest`: Defines the Solr URL to migrate the data to. This is just one of the Solr nodes in the cluster. The migrate feature will write the data across all nodes in the cluster.

`-destuser`: Solr username to use to log into destination Solr.

`-destpwd`: Solr password to use to log into destination Solr.

`-items`: Defines the jKQL items (and therefore the Solr collections) that data should be taken from. If omitted, move repository data for *all* jKQL items that can only reference a single repository (streaming data-related items like events, relatives, etc., and some definition items like Sets, Triggers, etc.). As mentioned earlier, all admin, reference, and dictionary (dashboard/viewlet) records are *not* tied to a single repository so must instead reside in master cluster. Therefore this data is *not* migrated.

`-dates`: This specifies a date filter, as a jKQL Date Filter expression, which will limit what streaming-related data is migrated.

`-qd`: The number of days in each date partition.

`-qb`: This specifies how many records to retrieve in a single query operation. The data for each item is queried for in blocks of this many records, until all data (for the current partition) is retrieved.

`-wb`: This specifies how many records to include in a single Solr document index (write) operation.

`-pw`: This specifies maximum number of Solr document index (write) operations to queue up. When this limit is reached, query operations will wait for write operations to be completed before continuing (so that queries don't get too far ahead of writes).

`-qt`: This specifies how many threads to use to perform the migration query operations. A migration "query operation" is the set of queries for a specific Solr collection. Using a value greater than one allows this many migration query operations to be run in parallel. When an item migration query operation is complete, the thread doing the item queries will go on to the next item in the queue.

`-wt`: This specifies how many threads to use to perform the migration writes. A migration "write operation" writes a block of records for a single item. Using a value greater than one allows this many Solr write operations to be done in parallel. Unlike query threads and write threads for other Data Services Database Maintenance Utility operations, these threads are not necessarily tied to a specific item. They are just a pool of threads performing Solr index operations (each operation is for a single item only, but any one thread will do writes for different items).

`-P`: this is the meshIQ Track Administrator user password.

`-C`: this is the Solr URL for the master Solr cluster. If not specified, the cluster from `-src` is also assumed to be the master cluster.

`-UD`: Solr username to use to log into master Solr cluster.

`-PD`: Solr password to use to log into master Solr cluster.

`-mr`: Maximum number of times to retry a failed Solr request.

`-t`: Timeout, in milliseconds, to use for Solr requests. If the Solr cluster is large with large collection sizes, you may need to increase this value.

4.1.2 Compare

The full syntax for running a compare is:

```
jkool-db-maint.sh -compare -repo:<repoid> -P:<jkool_pwd>
            -src:<src_db_url> [ -srcuser:<src_user> -srcpwd:<src_pwd> ]
            -dest:<dst_db_url> [ -destuser:<dest_user> -destpwd:<dest_pwd> ]
            [ -items:<jkql_items> ] [ -dates:<date_filter> ]
            [ -C:<db_url> [ -UD:<server_user> -PD:<server_pwd> ] ]
            [ -mr:<max_retries> ] [ -t:<solr_timeout> ]
```

4.2 Examples

Here is an example of the series of steps for migrating the contents of a repository:

1. At command line (for *yyyy-mm-dd*, use the current date):

   ```
   jkool-db-maint.sh -migrate
                     -repo:R_Repo1\$O_Org1
                     -src:http://<solr1-host>:8983
                     -dest:http://<solr2-host>:8983
                     "-dates: before yyyy-mm-dd"
                     -P:admin
   ```

2. When step one is complete (or if you are required to run it multiple times, after the last attempt has successfully completed), use QB (or the UI, if supported), to add/replace the current VolumeName field on the repository with the volume that the repository was migrated to. (in statement below, replace *vol-name* with the volume name that the cluster repository was migrated to):

```
Alter repository 'R_Repo1$O_Org1' volumeName='vol-name'
```

3. After verifying the repository, switch to the new volume (as described above) and rerun the migrate to move any additional records that were written since the original migrate in step 1. At the command line, for *yyyy-mm-dd*, use the same date that you used in step 1:

```
jkool-db-maint.sh -migrate
                  -repo:R_Repo1\$O_Org1
                  -src:http://<solr1-host>:8983
                  -dest:http://<solr2-host>:8983
                  "-dates: since yyyy-mm-dd"
                  -P:admin
```

4. (Optional.) Compare the record counts between the two Solr clusters. For *yyyy-mm-dd*, use the current date:

```
jkool-db-maint.sh -compare
                  -repo:R_Repo1\$O_Org1
                  -src:http://<solr1-host>:8983
                  -dest:http://<solr2-host>:8983
                  "-dates: before yyyy-mm-dd"
                  -P:admin
```

When you are satisfied that migration was successful, you can purge the records for the repository from the old cluster (see *Purging*).

Chapter 5: Purging

The Purge operation removes all data and definitions for the specified repository. This operation was added mainly to remove the data for a migrated repository from the old cluster but can be used to purge data for a repository for any reason. By default, executing purge from the Data Services Database Maintenance Utility will purge data from items whose records can only reference a single repository (streaming data-related items like events, relatives, etc., and some definition items like Sets, Triggers, etc.). The Purge operation will not remove dashboards. Dashboards would have to be removed using another utility, such as the jKQL Command Line Utility (`jkool-cmd`).

5.1 Usage

The full syntax for running a purge is:

```
jkool-db-maint.sh -purge -repo:<repoid> -P:<jkool_pwd>
            -src:<src_db_url> [ -srcuser:<src_user> -srcpwd:<src_pwd> ]
            [ -items:<jkql_items> ] [ -dates:<date_filter> ]
            [ -C:<db_url> [ -UD:<server_user> -PD:<server_pwd> ] ]
            [ -mr:<max_retries> ] [ -t:<solr_timeout> ]
```

`-repo`: The complete repository ID to purge.

`-src`: Defines the Solr URL to purge the data from. This is just one of the Solr nodes in the cluster. The migration will move data from all nodes in the cluster.

`-srcuser`: Solr username to use to log into Solr.

`-srcpwd`: Solr password to use to log into Solr.

`-items`: Defines the jKQL items (and therefore Solr collections) that data should be purged from. If omitted, data is purged from repository for *all* jKQL items that can only reference a single repository (streaming data-related items like events, relatives, etc., and some definition items like Sets, Triggers, etc.).

`-dates`: This specifies a date filter, as a jKQL Date Filter expression, which will limit what streaming-related data is purged.

`-P`: This is the meshIQ Track Administrator user password.

`-C`: This is the Solr URL for the master Solr cluster. If not specified, the cluster from `-src` is also assumed to be the master cluster.

`-UD`: Solr username to use to log into master Solr cluster.

`-PD`: Solr password to use to log into master Solr cluster.

`-mr`: Maximum number of times to retry a failed Solr request.

`-t`: Timeout, in milliseconds, to use for Solr requests.

5.2 Examples

To continue the migration example above, when migration is complete and the repository is active on the new volume, here is the command to purge the data for this repo from the old cluster:

```
jkool-db-maint.sh -purge
                  -repo:R_Repo1\$O_Org1
                  -src:http://<solr1-host>:8983
                  -P:admin
```

