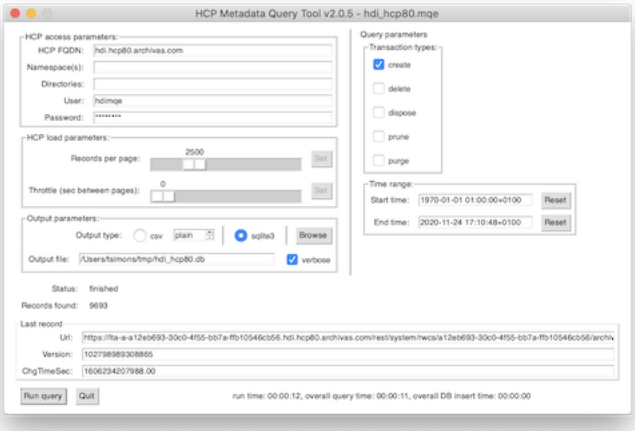


HCP MQE Tool

Release 2.0.11

Thorsten Simons

May 03, 2022

CONTENTS

CHAPTER 1

PROLOGUE / INTRO

There are situations where one needs to have information about all the objects stored in an Object Storage system, or even what has happened to an object during its lifetime. As well, sometimes one needs to find out if an object has been stored and later on deleted.

In general there is few abilities, beside of 'walking the tree', to get this kind of information from most of the Object Storage systems on the market.

For Hitachi Content Platform (HCP), things are different. HCP offers a built-in *metadata query engine* (MQE), which is able to provide the mentioned details.

The tool described in this document is using the *MQE API* to request information about object-related operations from HCP. Object-related operation means records describing what and when things happened to an object: it's creation, metadata changes as well as deletion (including disposition, prune and purge operations).

Is output is either a sqlite3 database file or comma-separated-value (csv) file, plain or compressed, holding a list of the requested operations.

Using the query options, it can answer questions like:

- which objects are in the system?
- which objects were deleted during a given time period?
- etc.

Some recipes for using the acquired data can be found in the *recipies* chapter.

CHAPTER 2

INSTALLATION

First of all, be aware that **hcpmqe** is a GUI-based tool. It will *not* run on a system without a GUI (headless Linux, for example).

Second, no binary installers are provided, due to the labor required to make it happen reliably for all platforms supported. A Python 3.7 (or newer) installation is required.

2.1 Installing the Python package

Given that Python 3 is installed, the process of installing **hcpmqe** is pretty straight forward. It's highly suggested to use a Python virtual environment, especially if the tools is used as a one-off.

Note: **Internet access is required** to be able to install the package, as it depend on other packages to be loaded from PyPi (the Python package index)[1].

This is how to do it:

- Check the Python version:

```
$ python3 --version
Python 3.7.4
```

  Note: Python >= 3.7 is required, any higher version should do as well.

- Create a folder to work in:

```
$ mkdir hcpmqe
$ cd hcpmqe
```

- Setup a Python virtual environment:

```
$ python3 -m venv .venv
```

- Activate the virtual environment:

[1] https://pypi.org

Linux, macOS:

```
$ source .venv/bin/activate
(.venv) $
```

Windows:

```
C:\Users\sm\hcpmqe> .venv\Scripts\activate
(.venv) C:\Users\sm\hcpmqe>
```

Noticed the changed prompt? This shows that you have activated the virtual environment.

- Update the Python setup tools:

```
(.venv) $ pip install -U pip setuptools
[.. a lot of messages shown here ..]
Successfully installed pip-19.2.2 setuptools-41.2.0
```

- Install the tools python package:

```
(.venv) $ pip install hcpmqe
Collecting hcpmqe
  Downloading hcpmqe-2.0.2.tar.gz (17 kB)
Collecting PySimpleGUI==4.30.0
  Using cached PySimpleGUI-4.30.0-py3-none-any.whl (233 kB)
Collecting httpx==0.16.1
  Using cached httpx-0.16.1-py3-none-any.whl (65 kB)
Collecting certifi
  Using cached certifi-2020.11.8-py2.py3-none-any.whl (155 kB)
Collecting httpcore==0.12.*
  Downloading httpcore-0.12.1-py3-none-any.whl (54 kB)
     |--------------------------------| 54 kB 968 kB/s
Collecting rfc3986[idna2008]<2,>=1.3
  Using cached rfc3986-1.4.0-py2.py3-none-any.whl (31 kB)
Collecting sniffio
  Using cached sniffio-1.2.0-py3-none-any.whl (10 kB)
Collecting h11==0.*
  Using cached h11-0.11.0-py2.py3-none-any.whl (54 kB)
Collecting idna; extra == "idna2008"
  Using cached idna-2.10-py2.py3-none-any.whl (58 kB)
Using legacy 'setup.py install' for hcpmqe, since package 'wheel' is␣
↪not installed.
Installing collected packages: PySimpleGUI, certifi, sniffio, h11,␣
↪httpcore, idna, rfc3986, httpx, hcpmqe
    Running setup.py install for hcpmqe ... done
Successfully installed PySimpleGUI-4.30.0 certifi-2020.11.8 h11-0.11.
↪0 hcpmqe-2.0.2 httpcore-0.12.1 httpx-0.16.1 idna-2.10 rfc3986-1.4.
↪0 sniffio-1.2.0
```

Now you can run the tool as described in the following chapters, by just calling `hcpmqe`.

Note: Please keep in mind that you **need to have the Python virtual environment activated** to be able to run the tool. If in need, simply activate it by running:

```
$ cd hcpmqe
$ source .venv/bin/activate
```

or

```
C:\Users\sm> cd hcpmqe
C:\Users\sm\hcpmqe> .venv\Scripts\activate
```

CHAPTER 3

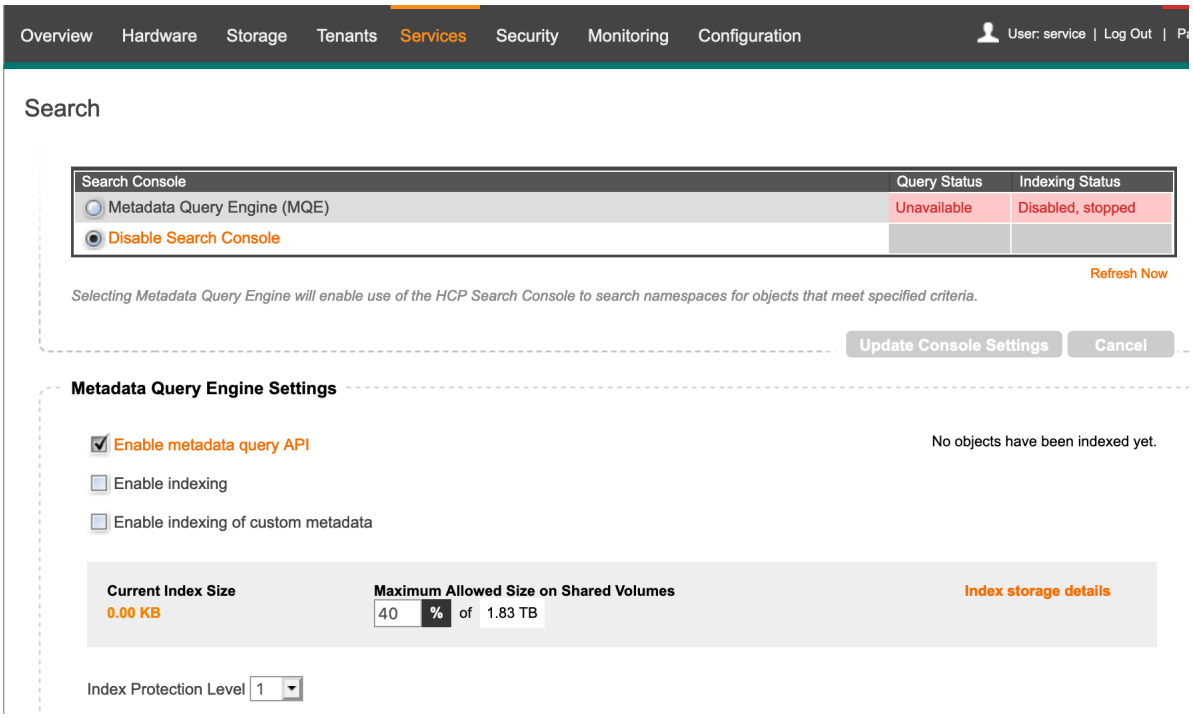
HCP PREREQUISITES

Warning: Not having the proper permissions and/or the MQE API being disabled will always lead to *error 403* when running a query:

```
Status:  fatal: 403 - Forbidden
```

3.1 MQE API

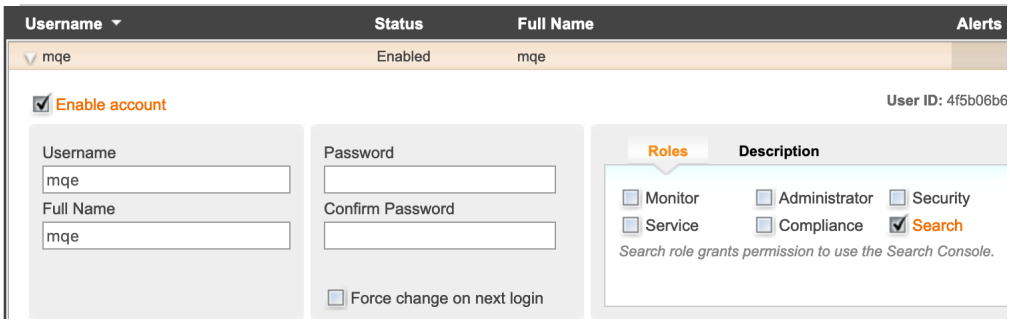
HCP needs to have the metadata query API enabled to allow the tool to function. The minimal setting needed can be set using the *HCP System Console* > *Services* > *Search* panel:



Enable metadata query API is the **only** setting required in **this** panel.

3.2 System administrator

A system-level administrator **must** at least have the **Search** role to access the MQE API:



Such an administrator is able to query Tenants that have granted system-level users to manage the tenant and search its namespaces in the respective *Tenant Console* > *Overview* panel:

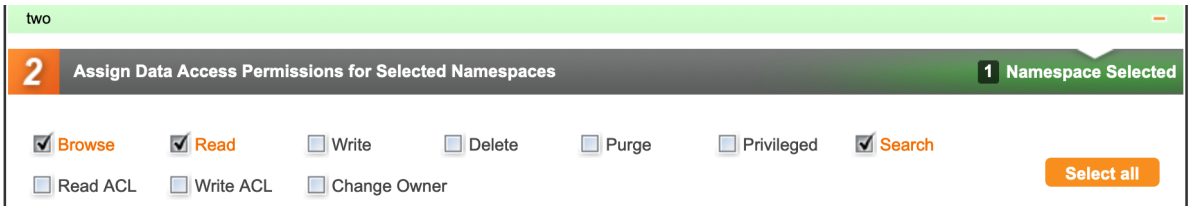
As a result of this, a full system-wide list of all operations can only be acquired if **all** Tenants have granted this privilege.

Using an HCP FQDN starting with *"tenantname."* will query just that Tenant. In this case, the *data network* configured for the Tenant must be reachable by the tool, and its FQDN must be resolvable via DNS.

Using an HCP FQDN starting with *"admin."* will query all Tenants that have granted the permission, even if the configured data network for some of the Tenants are not reachable by the tool.

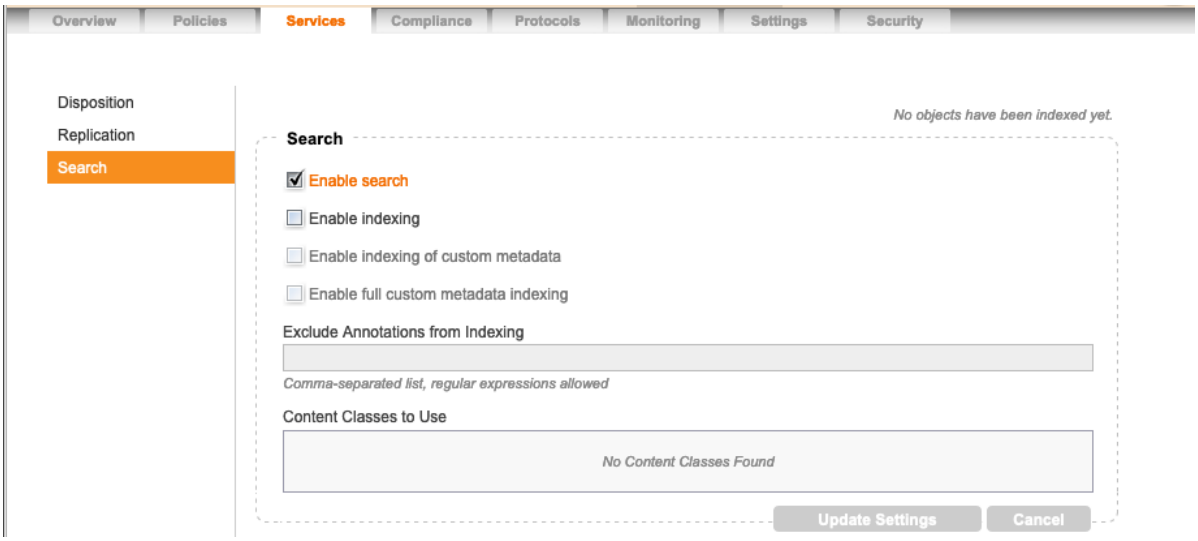
3.3 Tenant user

A Tenant user **must** have at least the **Search** permission for the Namespace(s) he shall query:



Of course, the tool must be able to reach the configured *data network* of the Tenant, and its FQDN must be resolvable via DNS.

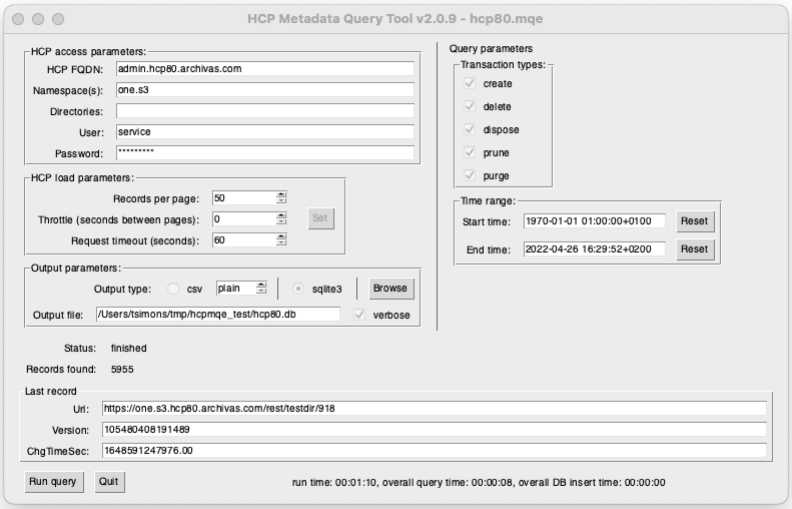
In addition to that, **Search** needs to be enabled for any Namespaces that shall be queried:


Overview
Policies
Services
Compliance
Protocols
Monitoring
Settings
Security
Disposition
Replication
Search
No objects have been indexed yet.
Search
Enable search
Enable indexing
Enable indexing of custom metadata
Enable full custom metadata indexing
Exclude Annotations from Indexing
Comma-separated list, regular expressions allowed
Content Classes to Use
No Content Classes Found
Update Settings
Cancel

CHAPTER 4

USER INTERFACE

4.1 Main panel



4.1.1 HCP access parameters

Here, the HCP system to query is addressed. Either the system can be addressed entirely (FQDN starting with *admin.*) or a specific Tenant (FQDN starting with the Tenants name) can be addressed.

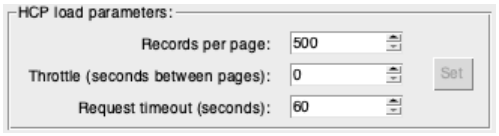
The query can optionally be re-fined by specifying one (or more) Namespaces, separated by comma.

Note: Please note that Namespaces **must always** be specified as *Namespace.Tenant*, even in case a specific Tenant is queried!

Further refining is available by specifying one (or more) directories (starting with /, and separated by comma). Please note that directories specified will be used for each and every Namespace addressed by the query.

The user specified must be a local HCP user (no AD account) with the proper permissions granted, as described in the prior chapter.

4.1.2 HCP load parameters



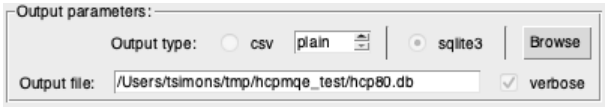
MQE queries can produce a huge amount of records to be fetched from HCP, depending on the number of objects addressed by the query. Therefore, paged queries of up to 10,000 records are used to keep the peak load in an acceptable range.

A throttle of up to 60 seconds can be tuned in to relax the load on HCP even more, at the cost of a longer query run time.

In case timeout errors are reported, try a longer request timeout than the default 60 seconds.

Tip: The values (except for timeout) can be changed while a query is running. Use the slider to change the value, then click the **[Set]** button. The new value will be picked up with the next page request.

4.1.3 Output parameters



Supported output types are comma-separated-value (csv-) files, plain as well as compressed (bz2, gzip, lzma), and Sqlite3 database files.

Selecting *verbose* will request *all* system metadata values per object from HCP, while not selecting it will request just the bare minimum (4 fields) that clearly identifies each object and the operation that triggered the record)

4.1.4 Query parameters

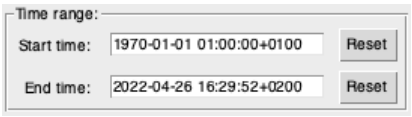
The operations (transactions) to be queried for:

- *create* - list all actually existing objects and their versions ingested

- *delete* - list all objects and object versions deleted
- *dispose* - list all objects deleted by disposition (automatic delete when retention period ends)
- *prune* - list all object versions automatic deleted when the configured version life span ended
- *purge* - list object versions deleted along with the objects actual version

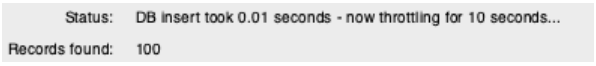
Note that only objects / versions are returned where the respective operation happened during the selected time frame. Also, note that -depending on HCP configuration- records of deleted objects / versions are held for a limited number of days, only.

4.1.5 Time range



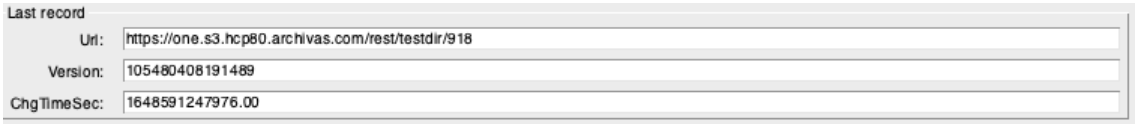
Defines the time range for which operations are requested.

4.1.6 Status



The *Status* line tells what's going on, *Records found* informs about how many records (object operations) have been returned so far. The *Last record* block tells about the identity of the last found record. These values can be used to restart an interrupted quuery, for example. See the following recipies chapter.

4.1.7 Last Record

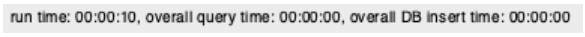


This area displays the last record received. It is either the last record within a received page (as long as a query is running), or the final record received during the query.

Note: The configuration file is auto-updated with these values after every page received successfully, to allow to continue with a query later on from exactly that position.

That means that a query can always be repeated or extended from that position - if a query finished successfully, if a query was canceled before finished, or if even the tool crashed.

4.1.8 Time bar



During and after a query, the time bar shows the overall run time, the time spent on page queries as well as the time spent on writing the database (or csv file).

CHAPTER 5

RUN QUERIES

5.1 Preparation

You **need to have the Python virtual environment (created during install) activated** to be able to run the tool. If in need, simply activate it by running:

Linux, macOS:

```
$ cd hcpmqe
$ source .venv/bin/activate
```

or

Windows:

```
C:\Users\sm> cd hcpmqe
C:\Users\sm\hcpmqe> .venv\Scripts\activate
```

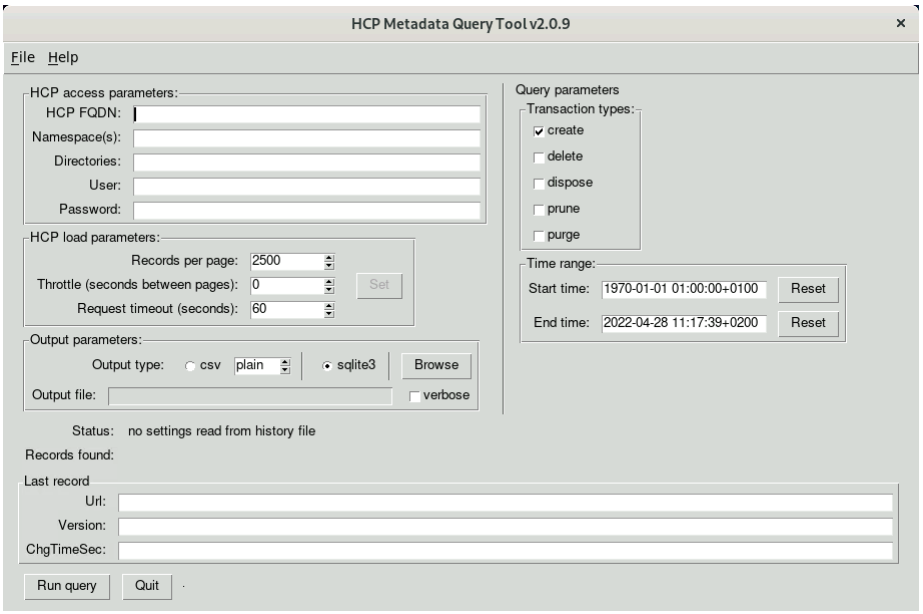
5.2 Start the tool

```
$ hcpmqe --help
usage: hcpmqe [-h] [--version] [-C]

optional arguments:
  -h, --help         show this help message and exit
  --version          show program's version number and exit
  -C, --log2console  instead of logging to hcpmqe.log, log to console
```

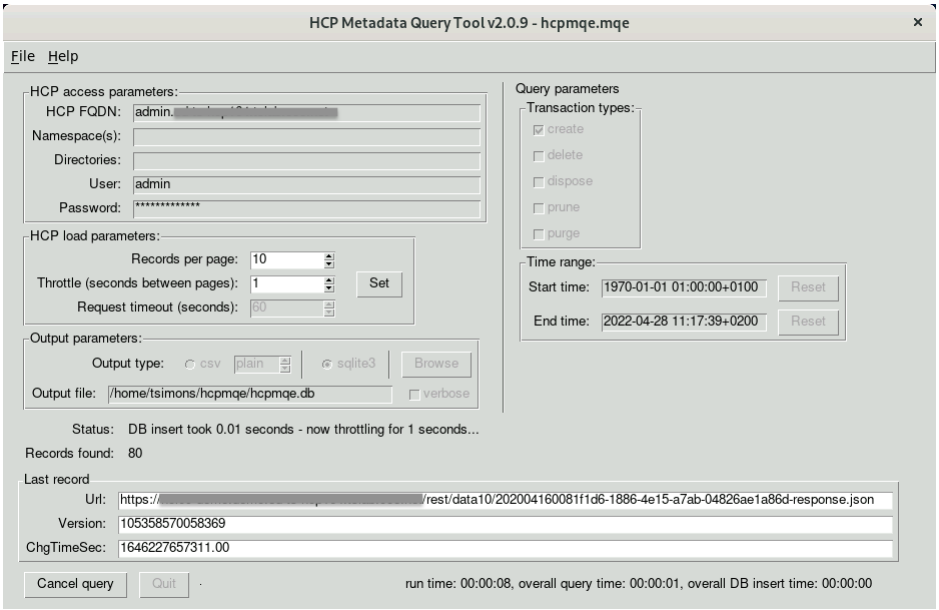
The tool always logs its doings - either into a file in the current directory (*hcpmqe.<pid>.log*), or, if the `-C` argument is used, to the console.

Running the command will open the GUI:



5.3 Run a query from scratch

Once the form is filled with parameters matching the wanted query, save the configuration, then click the [**Run query**] button to start the process.

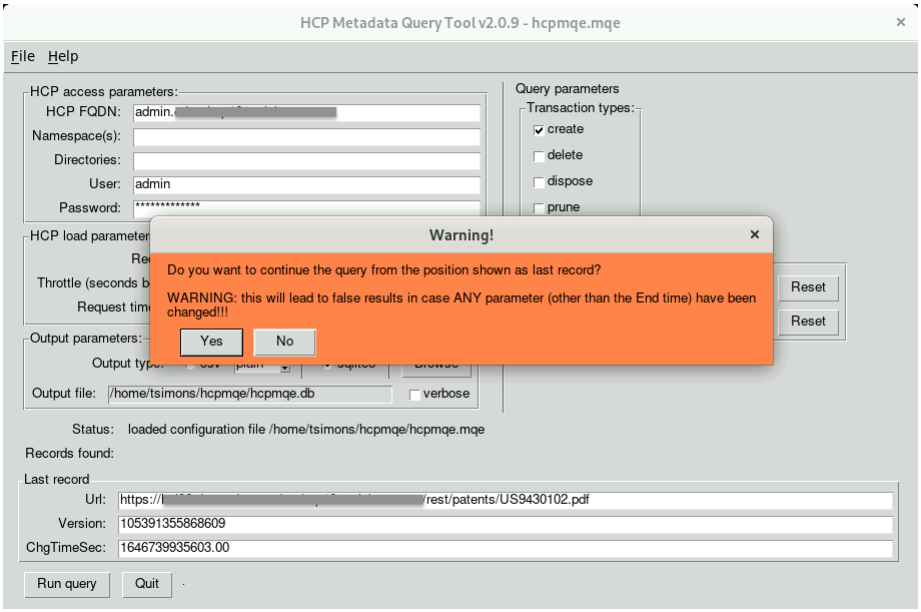


All the entry fields will be disabled, except the ones that allows to change page size and throttle. The *Status* line will show progress information, *Records found* reports the no. of records received so far, the *Last record* section shows the identity of the last pages final record, and in the very bottom, some timing information is displayed.

5.4 Re-start a query

If a query was canceled or interrupted for whatever reason, it can be restartet. If the tool crashed or was killed somehow, just start it again and load the configuration file. It will show information about the last record that was written to the output file.

Do **not change any (!!!) parameter** and press [**Run query**] again (changing values will likely cause the query to end up incomplete). You'll be asked if you want to continue or start from scratch.



CHAPTER 6

RECIPIES

6.1 Migration cross-check

6.1.1 Situation

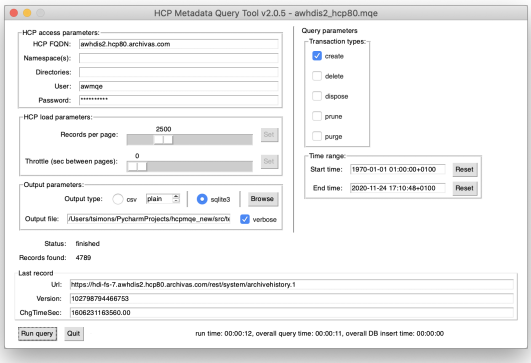
You've migrated a Namespace, a Tenant or an entire system to another HCP, using replication, and you need to have a verification that the data in source and target is exactly the same.

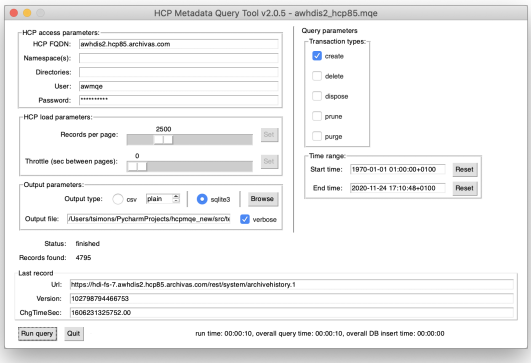
6.1.2 Recipe

This example will use a single Tenant as an example.

Acquire a list of existing objects from both HCP systems

- Use the **hcpmqe** tool to query both HCP systems for a list of existing objects:
  - Use a Tenant user with **Search** permission for all Namespaces within the Tenant
  - Select **create** as the only transaction type
  - Leave **Start time** at the default, set **End Time** to when you finished the migration
  - Select **sqlite3** as output format
  - Check **verbose**
  - Run the query for both involved HCP systems





– You should have two database files, once finished:

```
$ ls -lh *.db
-rw-r--r--  1 tsimons  staff   2.4M Nov 24 17:03 awhdis2_hcp80.
↪db
-rw-r--r--  1 tsimons  staff   2.4M Nov 24 09:04 awhdis2_hcp85.
↪db
```

Note: For a comparison like this, just a few of the columns in the databases are relevant to clearly identify an object:

- hash
- ingesttime
- namespace
- objectPath
- version

Some more are interesting, as well:

- replicated
- size

- Use the sqlite3 commandline tool to run SQL queries to compare the two databases:

  Note: For a valid result, make sure to limit the set of objects investigated to exactly the same time frame - we'll use the epoch time stamp (seconds since 1970/1/1

0:00:00) for that - you can use this to convert[2].

For this example, migration ended 2020/11/23 08:00:00 –> **1606114800** epoch time.

- Open the origin HCP database (awhdis2_hcp80.db):

```
$ sqlite3 awhdis2_hcp80.db
```

- Attach the migration target HCP database (awhdis2_hcp85.db):

```
sqlite> ATTACH 'awhdis2_hcp85.db' AS replica;
```

- Check if the no. of records are equal:

```
sqlite> SELECT count(*) FROM main.ops
            WHERE ingestTime <= 1606114800;
4673
sqlite> SELECT count(*) FROM replica.ops
            WHERE ingestTime <= 1606114800;
4673
```

- Check if there are any non-replicated objects:

```
sqlite> SELECT count(*) FROM main.ops
            WHERE NOT replicated
                  AND ingestTime <= 1606114800;
0
sqlite> SELECT count(*) FROM replica.ops
            WHERE NOT replicated
                  AND ingestTime <= 1606114800;
0
```

- Now, lets check which records don't exist in one of the databases:
  - List all records **not** in the migration target database:

```
sqlite> SELECT hash, ingesttime, namespace,
                objectPath, version FROM main.ops
            WHERE ingestTime <= 1606114800
            EXCEPT SELECT hash, ingesttime, namespace,
                          objectPath, version
                      FROM replica.ops
                          WHERE ingestTime <= 1606114800;
[..]
```

  - List all records **not** in the origin database:

```
sqlite> SELECT hash, ingesttime, namespace,
                objectPath, version FROM replica.ops
            WHERE ingestTime <= 1606114800
            EXCEPT SELECT hash, ingesttime, namespace,
```

(continues on next page)

[2] https://www.epochconverter.com

(continued from previous page)

```
                          objectPath, version
                        FROM main.ops
                            WHERE ingestTime <= 1606114800;
[..]
```

Alternative way to achieve the same result:

* List all records **not** in the migration target database:

```
sqlite> SELECT DISTINCT hash, ingesttime, namespace,
                        objectPath, version FROM main.ops
            WHERE ingestTime <= 1606114800
                AND (hash, ingesttime, namespace, objectPath,
                     version) NOT IN
                (SELECT DISTINCT hash, ingesttime, namespace,
                                 objectPath, version
                    FROM replica.ops
                        WHERE ingestTime <= 1606114800);
```

* List all records **not** in the origin database:

```
sqlite> SELECT DISTINCT hash, ingesttime, namespace,
                        objectPath, version FROM replica.ops
            WHERE ingestTime <= 1606114800
                AND (hash, ingesttime, namespace, objectPath,
                    version) NOT IN
                (SELECT DISTINCT hash, ingesttime, namespace,
                                 objectPath, version
                    FROM main.ops
                        WHERE ingestTime <= 1606114800);
```

CHAPTER 7

DATABASE SCHEMA

The schema of the `ops` database table, containing the collected operation records, differs between verbose or non-verbose query mode.

In addition, the database schema is build dynamically from the metadata keys HCP returns; that said, there might be slight differences between HCP versions. As of now (April 2022), this has been just added keys. Nevertheless, if such a change happens during an HCP version upgrade, the database in use might not be usable with the newer version of HCP, and thus needs to be created from scratch (*just delete the existing database and run a new query*).

Here are samples of the `ops` tables schema:

7.1 Non-verbose mode

```
                  Column | example value
-------------------------+-------------------------------------------------
  changeTimeMilliseconds | 1648129832613.00
               operation | CREATED
                 urlName | https://one.s3.hcp80.archivas.com/rest/hallo.txt
                 version | 105480309271425
```

7.2 Verbose mode

```
                  Column | example value
-------------------------+-------------------------------------------------
              accessTime | 1648129832
        accessTimeString | 2022-03-24T14:50:32+0100
                     acl | 0
  changeTimeMilliseconds | 1648129832613.00
        changeTimeString | 2022-03-24T14:50:32+0100
          customMetadata | 1
customMetadataAnnotation | .metapairs
```

(continues on next page)

(continued from previous page)

```
                  dpl | 1
                  gid | 0
                 hash | SHA-256 78FC...<cut>...232F
           hashScheme | SHA-256
                 hold | 0
               _index | 1
           ingestTime | 1648129832
     ingestTimeString | 2022-03-24T14:50:32+0100
            namespace | one.s3
           objectPath | /hallo.txt
            operation | CREATED
                owner | USER,s3,s3
          permissions | 555
           replicated | 0
 replicationCollision | 0
            retention | 0
       retentionClass |
      retentionString | Deletion Allowed
                shred | 0
                 size | 6
                 type | object
                  uid | 0
           updateTime | 1648129832
     updateTimeString | 2022-03-24T14:50:32+0100
              urlName | https://one.s3.hcp80.archivas.com/rest/hallo.txt
             utf8Name | hallo.txt
              version | 105480309271425
```

CHAPTER 8

RELEASE HISTORY

2.0.11 - 2022-05-03

- Added tool tips to most of the form fields

2.0.10 - 2022-04-28

- Warning box title corrected
- some documentation corrections

2.0.9 - 2022-04-28

- replaced the sliders in the UI with pre-seeded spin boxes
- simplified the HCP load parameters to a single [Set] button
- added the database schema to the documentation
- fixed a bug where the last record values were removed from the configuration file in case a repeated query did not return new records

2.0.8 - 2022-04-26

- made the request timeout configurable from the UI

2.0.7 - 2021-12-22

- copyright note fixed
- added _recipies folder, w/ a script to count objects per folder from a hcpmqe database

2.0.6 - 2021-10-20

- fixed a bug that causes SSL handshake errors when used with Python 3.10

2.0.5 - 2020-11-23

- start and end time are now in ISO 8601 format, added field verification

2.0.4 - 2020-11-17

- db/csv columns are now sorted by name, to make sure they are uniform across multiple runs
- fixed a bug where columns in sqlite3 databases were incorrectly named, occasionally
- fixed the start- / end-times (needs to be converted to milliseconds to be accurate)

2.0.3 - 2020-11-11

- preparation for publishing
- corrected the URL for help/documentation

2.0.2 - 2020-11-10

- configuration file now saved/loaded via menu entries
- configuration file is auto-updated when changes happen
- logging to file now into hcpmqe.<pid>.log

2.0.1 - 2020-11-08

- automatically adopts to whatever metadata fields the HCP MQE API delivers
- allows to restart a canceled or interrupted query

2.0.0 - 2020-11-03

- complete re-write using tkinter through pySimpleGUI
- runs on all major platforms (Linux, Windows, macOS)

1.0.x releases

- 1.0.11 - 2014-08-21

  [..]

- 1.0.1 - 2012-09-06

  initial release for Windows only

CHAPTER 9

LICENSE / TRADEMARKS

9.1 The MIT License (MIT)

9.2 Trademarks and Copyrights of used material