HCP Tool

User’s Manual

Release 1.0.8

License

Summary

While working with Hitachi Content Platform (HCP), some challenges may come up that are not easy to address without additional tools. Think of a customer who has created some directories and objects while playing around with a newly acquired HCP – how to get rid of them? Objects are somewhat easy deletable through the Namespace browser (if he played with an authenticated namespace), but what about directories? Or think of a customer who has several million objects ingested into HCP, now getting aware that he should have set a retention for these objects! How to change a million objects retention? No way with HCP alone.

Or think of a Proof of Concept situation – we need to showcase ingestion, reading and possibly deletion of objects into/within HCP. If it's more then CIFS or NFS access and we don't have an application to be tested with HCP, it's not that easy with HCP alone.

This is where '***HCP Tool***' steps in. As a command-line-tool, it offers functions to…

- Calculate the access token needed to access an authenticated namespace or the Management API
- Load HCP with test data
- List the content of a namespace (or parts of it)
- Change the retention of objects within HCP
- Delete objects from HCP, supporting Purge and Privileged Delete

Warning: some commands may have severe impact on a production HCP:

load - if you load data into a namespace running in ‚compliance' mode while setting a retention for the ingested objects, you will not be able to delete these objects before the retention has expired!

retention - you might lock data in longer retention then wanted if you use a nincorrect retention string - please refer to ‚Using a Namespace' on how to form this string!

unload - you might delete (purge) objects under retention in a namespace in ‚enterprise' mode.

Preparing for use

- Find the right server/workstation for '***HCP Tool***' to be run on -
  Don't run 'HCP tool' on a server running customer's production!
  ➔Under certain circumstances it may overload the server in terms of memory and cpu cycles, slowing down or killing other applications! Always use a server/workstation that has no impact to the customers business.
- Make sure the server/workstation has network access to HCP!
- Switch off the local DNS cache!
  ```
  c:> net stop dnscache
  ```
  This will allow '***HCP Tool***' to make use of HCPs load balancing mechanism!
- If possible, lower MS Windows' socket timeout parameter!
  See Appendix B for a description on how to do that.

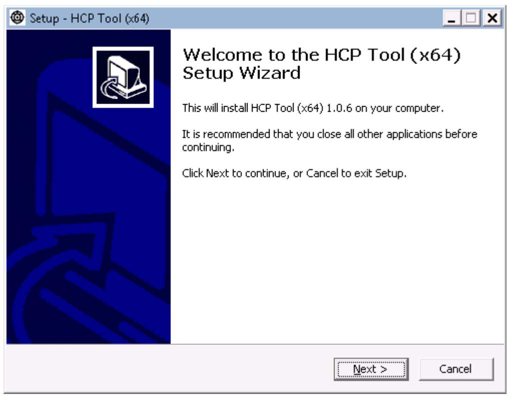
Installation on Linux

- ***HCP Tool*** needs an installation of Python, at least version 3.2.1.
- Download the latest pre-compiled release of ***HCP Tool*** for Linux (*hcpt_1.0.8.precomp.tar*) from:
  http://sourceforge.net/p/hcptool/wiki/Home

- You do not need to be ‚root' for the next steps, unless you want to enable all users to use ***HCP Tool*** on this server.
  - Unpack the tarball into a directory of your choice:
    ```
    sm@linux64:~> mkdir hcpt; cd hcpt
    sm@linux64:~/hcpt> tar -xvf hcpt_x.y.z.precomp.tar
    ```
  - Check for function:
    ```
    sm@linux64:~/hcpt> python hcpt.pyc --version
    ```

Installation on MS Windows

- Download the latest release of ‚***HCP Tool***' (*HCP Tool x.y.z (xxxx).exe*) from: http://sourceforge.net/p/hcptool/wiki/Home.
- Install it by running *HCP Tool x.y.z (xxxx).exe*:



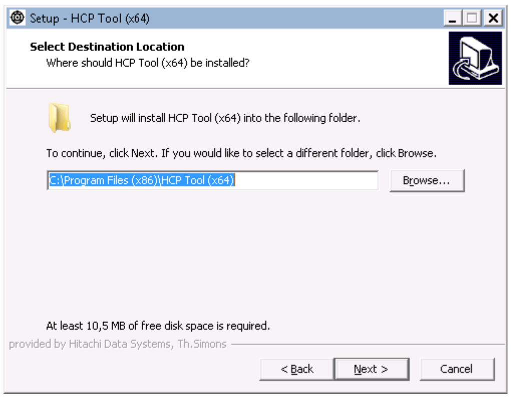
Setup - HCP Tool (x64)

Welcome to the HCP Tool (x64) Setup Wizard

This will install HCP Tool (x64) 1.0.6 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.



Setup - HCP Tool (x64)

Select Destination Location

Where should HCP Tool (x64) be installed?

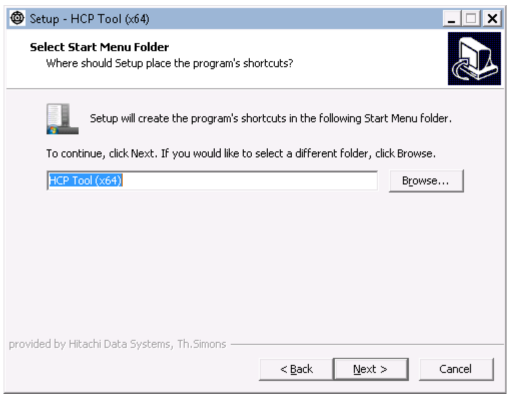
Setup will install HCP Tool (x64) into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files (x86)\HCP Tool (x64)

At least 10,5 MB of free disk space is required.

provided by Hitachi Data Systems, Th.Simons



Setup - HCP Tool (x64)

Select Start Menu Folder

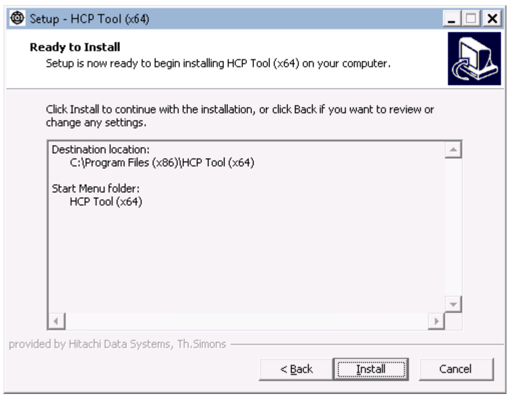
Where should Setup place the program's shortcuts?

Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

HCP Tool (x64)

provided by Hitachi Data Systems, Th.Simons



Setup - HCP Tool (x64)

Ready to Install

Setup is now ready to begin installing HCP Tool (x64) on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

```
Destination location:
      C:\Program Files (x86)\HCP Tool (x64)

Start Menu folder:
      HCP Tool (x64)
```

provided by Hitachi Data Systems, Th.Simons

(The installer will add the path to the executable file to the PATH environment variable.)

Verify the Installation

- Make sure that you have access to an HCP system and valid user credentials for an authenticated namespace at hand.
  The test may also be run against the default namespace, where's no need to have user credentials (simply provide random user / password).
- Open a Command-line Windows (cmd.exe)
- If ever possible, switch of the local machine's DNS cache:
  ```
  c:> net stop dnscache
  ```
- Run the '***HCP Tool***' in test mode:
  ```
  c:> hcpt -i5 --user <user> --password <password> test --cluster \
     ns.tenant.hcp.vm.loc --dir /rest/hcpt_test --file <filename> --structure 10 100
  ```
  (if the target namespace has 'versioning' eabled, add `--versionedNS` to the command above!)

  '***HCP Tool***' (hcpt.exe) will...
  1. calculate access tokens for both namespace and
  2. MAPI access
  3. feed file <filename> 100 times in each of 10 subdirectories of /rest/hcpt_test within namespace 'ns' in tenant 'tenant' within an HCP with a DNS name of 'hcp.vm.loc'
  4. discover a list of all objects and directories, beginning at /rest/hcpt_test. This list will be provided as 2 files: hcpt_test.csv (to be loaded into MS Excel) and hcpt_list.<timestamp>.db, a Sqlite3-database that will be used as base for the nexts steps.
  5. update the database file with a new retention setting to be processed in the next step.
  6. update HCP with the new retention settings for the objects selected in the database in the prior step.
  7. delete (purge) all the objects and the directories where they have been stored.

  At the end, there will be a logfile (hcp_test.log) showing all the output that has been showed on the screen, an Excel-importable file (hcp_test.csv) conaining a list of all objects and directories found in 4. and two Sqlite3 database files, containing the results of 4. and 7.
  (See Appendix A for an example)

General usage

When installed, '***HCP Tool***' can be used by running the file 'hcpt.exe'. It takes some arguments, telling it what to do. You can always get some help by running `hcpt --help` (or: `hcpt 'subcommand' --help`)

The syntax is:

```
C:> hcpt [common arguments] subcommand [subcommand arguments]
```

- **`common arguments`** are valid for all subcommands

-h, --help	show help and exit
--version	show program's version number and exit
-u USER, --user USER	data access acount
-p PASSWORD, --password PASSWORD	password (will require manual input if not given)
-l LOGFILE, --logfile LOGFILE	logfile (defaults to 'hcpt_subcmd.log')
-i seconds, --loginterval seconds	logging interval (defaults to 10 sec.)
-t '# of threads', --threads '# of threads'	no. of parallel threads (defaults to 30)
--nossl	use http instead of https
-v	verbosity (-v = INFO, -vv = DEBUG, -vvv = garbage collection statistics)
--gc t1.t2.t3	garbage collection thresholds (defaults to '700.10.10' - see 'http://docs.python.org/py3k/library/gc.html#gc.set_threshold')

- **`subcommand`**s are

cookie	calculate HCP access token
load	load bulk testdata into HCP
list	list HCP content
retention	change retention setting for selected objects
test	test-run all the subcommands
unload	delete content from HCP

- **`subcommand arguments`**
  are described below

Subcommands

- **cookie** - calculate HCP access token
  Calculate the HCP access token to be used in http-requests.

```
usage: hcpt cookie [-h] [--version] {daac,mapi}

positional arguments:
  {daac,mapi}  account type (DataAccessACount or MAPI)

optional arguments:
  -h, --help   show this help message and exit
  --version    show subfunctions version and exit
```

Example:
```
C:> hcpt --user max --password meier cookie daac
hcp-ns-auth=bWF4:7a3bbfa99f014f41f2a4b368391c092c
```

- **load** - load bulk testdata into HCP
  Perform bulk data ingestion into HCP for testing (!) purposes.

```
usage: hcpt load [-h] [--version] -c CLUSTER -d directory -f ingestfile
                 [-r retention_string] --structure # [# ...]

optional arguments:
  -h, --help            show this help message and exit
  --version             show subfunctions version and exit
  -c CLUSTER, --cluster CLUSTER
                        target namespace (full qualified DNS-name)
  -d directory, --dir directory
                        target directory ('/rest/...' or '/fcfs_data/...')
  -f ingestfile, --file ingestfile
                        file to be ingested
  -r retention_string, --retention retention_string
                        retention (requires valid HCP retention string)
  --structure # [# ...]
                        directory structure to be build
```

Controlled by '--structure [#_of_dirs [#_of_dirs [...]]] #_of_files', a directory structure is build and '#_of_files' copies of 'ingestfile' will be ingested into each lowest level directory. Example: '3 3 3' causes three directories to be created below 'targetdir' (0000, 0001, 0002), with another three subdirectories (0000, 0001, 0002) in each of them and three copies of 'ingestfile' to be written into each of these subdirectories. Be cautious, you could use up a lot of capacity in HCP and generate a lot of network trafic while using it...

Example:
```
C:> hcpt --user ns1 --password ns101 -v -i3 load --cluster ns1.matrix.hcp1.vm.local \
          --dir /rest /hcpt_test1 --file c:\hitachi_logo.txt --structure 10 10 1
```

- **list** - list HCP content
  Discover all objects in a given subdirectory within an HCP namespace
  while discovering the directory tree top/down. List the found objects and directories in a MS Excel usable file (*.csv) and in a Sqlite3 database file.

```
usage: hcpt list [-h] [--version] -c CLUSTER -d directory [--all]
                 [-B DATABASE] [--fatDB] [--keepDB] [--QF queuesize]
                 [--Qdb queuesize] [--delay milliseconds] [--outfile OUTFILE]
                 [--nooutfile] [--showThreads]

optional arguments:
  -h, --help            show this help message and exit
  --version             show subfunctions version and exit
  -c CLUSTER, --cluster CLUSTER
                        target namespace (full qualified DNS-name)
  -d directory, --dir directory
                        target directory ('/rest/...' or '/fcfs_data/...')
  --all                 find deleted objects, too (if versioning is configured
                        for the namespace)
  -B DATABASE, --database DATABASE
                        database file (defaults to
                        'hcpt_list.<timestamp>.[fat|slim].sqlite3')
  --fatDB               include all available information in database
  --keepDB              do not delete the database file when finished
  --QF queuesize        defines the allowed no. of items in FindQueue
  --Qdb queuesize       defines the allowed no. of items in dbWriterQueue
  --delay milliseconds  add a delay (pause) in ms between two requests
                        executed against HCP by a single thread
  --outfile OUTFILE     filename for the resulting .csv file (defaults to
                        'hcpt list.csv')
  --nooutfile           don't write an outfile, but keep database
  --showThreads         show info about running threads
```

  Be aware: when discovering large directory trees, memory usage might become a problem, up to the point where this program might hang or even crash. You should monitor it by using '-v' or even '-vvv'. Best advice is to limit the number of threads (-t) to not more than 50 and limit the queues (--QF and --Qdb) to 10.000 and 20.000 respectively. You might encounter a deadlock situation, where "--QF" will be at max. and no object will be found. In this case, you'll need to unlimit '--QF' and maybe lower the threads. Speeding up the garbage collection by tuning '--gc' might help, too. But take care: this program might grab as many main memory as available, potentially affecting other applications - it's up to you to monitor that! Expect long (and I mean: really long) run times when discovering multi-million object directory trees! If you'd like to work with the database generated by this program, you could use tools provided at 'http://www.sqlite.org/download.html'. The Windows distribution provides an executable database shell (sqlite3.exe) in the installation folder.

  Example:

```
C:> hcpt --user ns1 --password ns101 -v -i3 list --cluster ns1.matrix.hcp1.vm.local \
          --dir /rest/hcpt_test1
```

- **retention** - change retention setting for selected objects
  Takes a database generated by 'hcpt list' as input to update the retention setting of the objects listed in the database. After generating the database with 'hcpt list', the field 'flist.new_ret' must be updated with the new retention setting for each object (see description below).

```
usage: hcpt retention [-h] [--version] -B DATABASE [--delay DELAY]

optional arguments:
  -h, --help            show this help message and exit
  --version             show subfunctions version and exit
  -B DATABASE, -database DATABASE
                        database file generated by 'hcp list' and altered as
                        described below
  --delay DELAY         add a delay (pause) in ms between two requests
                        executed against HCP by a single thread
```

  To alter the database, you could use 'sqlite3.exe', provided in the installation folder of this tool. For example, if your database file is called 'hcplist.sqlite3' and you want to add 1 year to every object's retention, you could follow these steps prior to running this tool:

```
...
c:\> sqlite3 hcplist.sqlite3
sqlite> UPDATE flist SET new_ret='R+1y' WHERE type='file' OR type='object';
sqlite> .quit
...
```

  It is **YOUR** responsibility to specify a valid retention string[1] - '***HCP Tool***' will not check it for validity!!!

  Example:

```
c:> hcpt --user ns1 --password ns101 -v -i3 retention --database hcplist.sqlite3
```

- **test** - test-run all the subcommands
  Runs all subcommands agains HCP, making sure that the program works.

```
usage: hcpt test [-h] [--version] -c CLUSTER -d directory -f ingestfile
                 [-r retention_string] --structure # [# ...]

optional arguments:
  -h, --help            show this help message and exit
  --version             show subfunctions version and exit
  -c CLUSTER, --cluster CLUSTER
                        target namespace (full qualified DNS-name)
  -d directory, --dir directory
                        target directory ('/rest/...' or '/fcfs_data/...')
  -f ingestfile, --file ingestfile
                        file to be ingested
  -r retention_string, --retention retention_string
                        retention (defaults to 'N+1s')
  --structure # [# ...]
                        directory structure to be build
```

  Example (see section 'Verify the Installation'):

```
c:> hcpt -i5 --user <user> --password <password> test --cluster \
       ns.tenant.hcp.vm.loc --dir /rest/hcpt_test --file <filename> --structure 10 100
```

[1] See HCP's manual‚Using a Namespace' on how to form a valid retention string.

- **unload** - delete content from HCP
  Perform deletion of data within HCP namespaces by discovering a directory tree top/down (alternatively, a list with objects to be deleted can be provided). Will find all directories and objects within that tree and will imediately begin with object deletion right after one has been found. Directory deletion will start down/up when the whole tree has been discovered. It will write a sqlite3 database file with a single record for each directory and object found, containing all the information available for it. This can grow quite large...

```
usage: hcpt unload [-h] [--version] -c CLUSTER -d directory
                   [--infile INFILE] [-B DATABASE] [--fatDB] [--keepDB]
                   [--QF queuesize] [--Qdb queuesize] [--objonly] [--purge]
                   [--privileged REASON] [--YES]

optional arguments:
  -h, --help            show this help message and exit
  --version             show subfunctions version and exit
  -c CLUSTER, --cluster CLUSTER
                        target namespace (full qualified dns-name)
  -d directory, --dir directory
                        target directory (/rest/... or /fcfs_data/...)
  --infile INFILE       file holding a list of objects to be deleted (full
                        path: '/rest/.../object' or '/fcfs_data/.../object'.
                        If set, '--dir' will be used to determine the type of
                        namespace, only.
  -B DATABASE, --database DATABASE
                        database file (defaults to
                        'hcpt_list.<timestamp>.[fat|slim].sqlite3')
  --fatDB               include all available information in database
  --keepDB              do not delete the database file when finished
  --QF queuesize        size of internal queue (defaults to unlimited)
  --Qdb queuesize       defines the allowed no. of items in dbWriterQueue
  --objonly             do not delete directories
  --versionedNS         set this if target namespace has versioning enabled
  --purge               purge versions (if not set, directory deletion will
                        fail if versioning is enabled)
  --privileged REASON   perform privileged delete (requires a 'reason')
  --YES                 ...if you really (!) want to delete the found
                        objects/directories (defaults to 'generate a list of
                        objects/directories only')
```

Be aware: if you have directories with a huge number (10.000++) of objects, main memory will become excessive used, even more the more threads you use. This could lead to runtime errors - in this case you will need to serialize the processing by limiting the number of threads down to 1 (one) depending on the available main memory. Of course, this will lead to a much longer runtime - monitor the processing by using the commandline switch '-v'.

Example:
```
c:> hcpt --user ns1 --password ns101 -v -i3 unload --cluster \
          ns1.matrix.hcp1.vm.local --dir /rest/hcpt_test1 --YES
```

Appendix A

Example logfile from 'hcpt test'

```
07/05 10:36:46 [INFO    ] hcpt test v.0.9.2 (2011-07-05/Sm)
07/05 10:36:46 [INFO    ]     Logfile: hcpt_test.log
07/05 10:36:46 [INFO    ]      Target: https://ns1.matrix.hcp1.vm.local/rest/hcpt_test
07/05 10:36:46 [INFO    ] DataAccAcnt: ns1
07/05 10:36:46 [INFO    ]  Ingestfile: c:\hitachi_logo.txt
07/05 10:36:46 [INFO    ]  Started at: Tue, 05.07., 10:36:46
07/05 10:36:46 [INFO    ] =====================================================
07/05 10:36:46 [INFO    ]
07/05 10:36:46 [INFO    ] -------------------------------------------------
07/05 10:36:46 [INFO    ] Test 1: generate a cookie for a DataAccessAccount
07/05 10:36:46 [INFO    ] -------------------------------------------------
07/05 10:36:46 [INFO    ] hcp-ns-auth=bnMx:780c3adf708201cbe71f9a9594bdf12c
07/05 10:36:46 [INFO    ]
07/05 10:36:46 [INFO    ] -----------------------------------------
07/05 10:36:46 [INFO    ] Test 2: generate a cookie for MAPI access
07/05 10:36:46 [INFO    ] -----------------------------------------
07/05 10:36:46 [INFO    ] hcp-api-auth=bnMx:780c3adf708201cbe71f9a9594bdf12c
07/05 10:36:46 [INFO    ]
07/05 10:36:46 [INFO    ] ----------------------------------
07/05 10:36:46 [INFO    ] Test 3: load some objects into HCP
07/05 10:36:46 [INFO    ] ----------------------------------
07/05 10:36:46 [INFO    ] hcpt load v.1.0.6 (2011-06-29/Sm)
07/05 10:36:46 [INFO    ]     Logfile: hcpt_test.log
07/05 10:36:46 [INFO    ] Loginterval: 5 sec.
07/05 10:36:46 [INFO    ]      Target: https://ns1.matrix.hcp1.vm.local/rest/hcpt_test/
07/05 10:36:46 [INFO    ] DataAccAcnt: ns1
07/05 10:36:46 [INFO    ]  Ingestfile: c:\hitachi_logo.txt
07/05 10:36:46 [INFO    ]    Filesize: 2051 Bytes (= 2.00 Kbyte)
07/05 10:36:46 [INFO    ]   Retention: 0
07/05 10:36:46 [INFO    ]   Structure: [10, 100] = 1,000 PUTs (= 1.96 Mbyte)
07/05 10:36:46 [INFO    ]     Threads: 30
07/05 10:36:46 [INFO    ]  Started at: Tue, 05.07., 10:36:46
07/05 10:36:46 [INFO    ] --------------------------------------------------
07/05 10:36:46 [INFO    ] MonitorThread started monitoring Worker.Qbox.qsize()
07/05 10:36:51 [INFO    ] files sent: 161, errors: 0, PUTs/sec.: 32.1
07/05 10:36:56 [INFO    ] files sent: 441, errors: 0, PUTs/sec.: 44.0
07/05 10:37:01 [INFO    ] files sent: 675, errors: 0, PUTs/sec.: 44.9
07/05 10:37:06 [INFO    ] files sent: 934, errors: 0, PUTs/sec.: 46.7
07/05 10:37:11 [INFO    ] files sent: 1000, errors: 0, PUTs/sec.: 40.0
07/05 10:37:11 [INFO    ] MonitorThread exits on Worker.Qbox.qsize() == 0
07/05 10:37:11 [INFO    ] --------------------------------------------------
07/05 10:37:11 [INFO    ]  Started at: Tue, 05.07., 10:36:46
07/05 10:37:11 [INFO    ]    Ended at: Tue, 05.07., 10:37:11
07/05 10:37:11 [INFO    ]     Runtime: 25.03 Sekunden
07/05 10:37:11 [INFO    ]     Success: 1000
07/05 10:37:11 [INFO    ]      Errors: 0
07/05 10:37:11 [INFO    ]   Obj./Sec.: 39.9
07/05 10:37:11 [INFO    ]    Transfer: 80.01 Kbyte/sec.
07/05 10:37:11 [INFO    ]
07/05 10:37:11 [INFO    ] -------------------------------------
07/05 10:37:11 [INFO    ] Test 4: list the newly fed-in objects
07/05 10:37:11 [INFO    ] -------------------------------------
07/05 10:37:11 [STARTUP ] hcpt list v.1.2.7 (2011-06-29/Sm)
07/05 10:37:11 [STARTUP ]     Logfile: hcpt_test.log
07/05 10:37:11 [STARTUP ] Loginterval: 5 sec.
07/05 10:37:11 [STARTUP ]    Database: hcpt_list.20110507103646.db
07/05 10:37:11 [STARTUP ]              (will be kept when finished)
07/05 10:37:11 [STARTUP ]              (will include deleted objects)
07/05 10:37:11 [STARTUP ]      Target: https://ns1.matrix.hcp1.vm.local/rest/hcpt_test
07/05 10:37:11 [STARTUP ]     OutFile: hcpt_test.csv
07/05 10:37:11 [STARTUP ] DataAccAcnt: ns1
07/05 10:37:11 [STARTUP ]     Threads: 30
07/05 10:37:11 [STARTUP ]     FindQue: -1
07/05 10:37:11 [STARTUP ] dbWriterQue: -1
07/05 10:37:11 [STARTUP ]   Verbosity: 1
07/05 10:37:11 [STARTUP ]  Started at: Tue, 05.07., 10:37:11
07/05 10:37:11 [STARTUP ] ------------------------------------------------------------------------------
```

```
07/05 10:37:11 [INFO    ] *** monitor started ***
07/05 10:37:11 [INFO    ] *** dbWriter started ***
07/05 10:37:11 [INFO    ] *** Discovery finished after 00:00:00.48 ***
07/05 10:37:11 [INFO    ] *** dbWriter finished after 00:00:00.55***
07/05 10:37:16 [INFO    ] Dirs:11; Objs:1,000; Err/Warn:0/0; dbWrites:1,011 [QF:0; Qdb:0]
07/05 10:37:16 [INFO    ] *** monitor exits ***
07/05 10:37:16 [INFO    ] *** writing outfile 'hcpt_test.csv' ***
07/05 10:37:16 [INFO    ] *** finished writing outfile 'hcpt_test.csv' after 00:00:00.2 ***
07/05 10:37:16 [STOPDOWN] -------------------------------------------------------------------------------
07/05 10:37:16 [STOPDOWN]  Started at: Tue, 05.07., 10:37:11
07/05 10:37:16 [STOPDOWN]    Ended at: Tue, 05.07., 10:37:16
07/05 10:37:16 [STOPDOWN]     Runtime: 00:00:05.5 h:m:s.ms
07/05 10:37:16 [STOPDOWN]       Found: 11 Directories, 1,000 Objects
07/05 10:37:16 [STOPDOWN]    Warnings: 0
07/05 10:37:16 [STOPDOWN]      Errors: 0
07/05 10:37:16 [INFO    ]
07/05 10:37:16 [INFO    ] -------------------------------------------------------
07/05 10:37:16 [INFO    ] Test 5: prepare the database file for retention changes
07/05 10:37:16 [INFO    ] -------------------------------------------------------
07/05 10:37:16 [INFO    ] *** database update begins ***
07/05 10:37:16 [INFO    ] *** changing all object's retention to 'N+1s' ***
07/05 10:37:16 [INFO    ] *** database update finished ***
07/05 10:37:16 [INFO    ]
07/05 10:37:16 [INFO    ] --------------------------------------------------------
07/05 10:37:16 [INFO    ] Test 6: change the retention of the newly fed-in objects
07/05 10:37:16 [INFO    ] --------------------------------------------------------
07/05 10:37:16 [STARTUP ] hcpt retention v.0.9.5 (2011-07-02/Sm)
07/05 10:37:16 [STARTUP ]     Logfile: hcpt_test.log
07/05 10:37:16 [STARTUP ] Loginterval: 5 sec.
07/05 10:37:16 [STARTUP ]    Database: hcpt_list.20110507103646.db (1.2.7/2011-06-29/Sm)
07/05 10:37:16 [STARTUP ]      Target: https://ns1.matrix.hcp1.vm.local
07/05 10:37:16 [STARTUP ] DataAccAcnt: ns1
07/05 10:37:16 [STARTUP ]     Threads: 30
07/05 10:37:16 [STARTUP ]   Verbosity: 1
07/05 10:37:16 [STARTUP ]  Started at: Tue, 05.07., 10:37:16
07/05 10:37:16 [STARTUP ] -------------------------------------------------------------------------------
07/05 10:37:16 [INFO    ] *** monitor started ***
07/05 10:37:16 [INFO    ] *** Changing retentions started ***
07/05 10:37:21 [INFO    ] Changed: 427, Errors=0, Warnings=0
07/05 10:37:26 [INFO    ] Changed: 825, Errors=0, Warnings=0
07/05 10:37:28 [INFO    ] *** Changing retentions finished after 00:00:12.71 ***
07/05 10:37:31 [INFO    ] Changed: 1,000, Errors=0, Warnings=0
07/05 10:37:31 [INFO    ] *** monitor exits ***
07/05 10:37:31 [STOPDOWN] -------------------------------------------------------------------------------
07/05 10:37:31 [STOPDOWN]  Started at: Tue, 05.07., 10:37:16
07/05 10:37:31 [STOPDOWN]    Ended at: Tue, 05.07., 10:37:31
07/05 10:37:31 [STOPDOWN]     Runtime: 00:00:15.3 h:m:s.ms
07/05 10:37:31 [STOPDOWN]     Changed: 1000
07/05 10:37:31 [STOPDOWN]    Warnings: 0
07/05 10:37:31 [STOPDOWN]      Errors: 0
07/05 10:37:31 [INFO    ]
07/05 10:37:31 [INFO    ] ---------------------------------------
07/05 10:37:31 [INFO    ] Test 7: delete the newly fed-in objects
07/05 10:37:31 [INFO    ] ---------------------------------------
07/05 10:37:31 [STARTUP ] hcpt unload v.1.1.8 (2011-07-04/Sm)
07/05 10:37:31 [STARTUP ]     Logfile: hcpt_test.log
07/05 10:37:31 [STARTUP ] Loginterval: 5 sec.
07/05 10:37:31 [STARTUP ]    Database: hcpt_unload.20110507103646.db
07/05 10:37:31 [STARTUP ]              (will be kept when finished)
07/05 10:37:31 [STARTUP ]      Target: https://ns1.matrix.hcp1.vm.local/rest/hcpt_test
07/05 10:37:31 [STARTUP ] DataAccAcnt: ns1
07/05 10:37:31 [STARTUP ]     Threads: 30
07/05 10:37:31 [STARTUP ]   QueueSize: -1
07/05 10:37:31 [STARTUP ]       Purge: yes
07/05 10:37:31 [STARTUP ]   Verbosity: 1
07/05 10:37:31 [STARTUP ]  DeleteMode: REQUESTED - Objects and Directories
07/05 10:37:31 [STARTUP ]  Started at: Tue, 05.07., 10:37:31
07/05 10:37:31 [STARTUP ] -------------------------------------------------------------------------------
07/05 10:37:31 [WARNING ] MonitorThread started - monitoring Worker Threads
07/05 10:37:31 [INFO    ] *** dbWriter started ***
07/05 10:37:31 [WARNING ] ***Discovery finished***
07/05 10:37:36 [INFO    ] Dirs: 11/0, Objects: 1000/265 (found/deleted), Errors: 0 [Q:0/705/0 (find/del/dbW)]
07/05 10:37:41 [INFO    ] Dirs: 11/0, Objects: 1000/723 (found/deleted), Errors: 0 [Q:0/247/0 (find/del/dbW)]
07/05 10:37:44 [WARNING ] ***Object deletion finished***
07/05 10:37:45 [INFO    ] Thread-132: http (DELETE) error: 403:/rest/hcpt_test (re-scheduled for later deletion)
```

```
07/05 10:37:45 [WARNING ] ***Directory deletion finished***
07/05 10:37:46 [INFO    ] Dirs: 11/11, Objects: 1000/1000 (found/deleted), Errors: 0 [Q:0/0/0 (find/del/dbW)]
07/05 10:37:46 [WARNING ] MonitorThread exits - all Worker Threads ended
07/05 10:37:46 [STOPDOWN] ------------------------------------------------------------------------------
07/05 10:37:46 [STOPDOWN]  Started at: Tue, 05.07., 10:37:31
07/05 10:37:46 [STOPDOWN]    Ended at: Tue, 05.07., 10:37:46
07/05 10:37:46 [STOPDOWN]     Runtime: 15.03 Sekunden
07/05 10:37:46 [STOPDOWN]       Found: 11 Directories, 1000 Objects
07/05 10:37:46 [STOPDOWN]     Deleted: 11 Directories, 1000 Objects
07/05 10:37:46 [STOPDOWN]      Errors: 0
07/05 10:37:46 [INFO    ]
07/05 10:37:46 [INFO    ] =====================================================
07/05 10:37:46 [INFO    ]  Started at: Tue, 05.07., 10:36:46
07/05 10:37:46 [INFO    ]    Ended at: Tue, 05.07., 10:37:46
07/05 10:37:46 [INFO    ]     Runtime: 60.26 Sekunden
```

Appendix B

If , ***HCP Tool***‘ is used effectively, it may send many requests per second to HCP. This may lead to ‚10048 - Address already in use‘ errors. This happens because Windows doesn’t release used (and closed) TCP/IP-scockets until a timeout of 240 seconds (default) has elapsed. This adds to the fact, that the nummer of outgoing TCP/IP-ports is limited to less than 4.000. Taken as fact that every socket needs it’s own outgoing port, it get’s clear that we will run out of free sockets when doing multi-ten requests per second. ‘***HCP Tool***’ catches the described error, waits a second and then tries again - but that’s just a workaround slowing down the process substantially.

This can be fixed by:

- lowering the „**TCP/IP socket connection timeouts**“ from 240 seconds (default) to e.g. 30 seconds. A Registry-Key needs to be added as DWORD:

  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\TcpTimedWaitDelay = 30

  → In the test environment, there was a gain of 20% in performance; the error disappeared.
- Increasing the **number of dynamic available TCP/IP Ports** (indirect by increasing the highst available dynamic Port). A Registry-Key needs to be added as DWORD:

  HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\MaxUserPort = 32768

  → This seems to put more load on the systems – CPU-load is substancially higher and the system feels tenacious.

-

Please consider:

- Changing these Registry-Keys needs an system restart to activate them!
- Changing theses keys has effect for the whole system - other applications could be affected! These Changes must be done with care - the author is not willing to take responsibility for whatever impact they might have!

See also: http://msdn.microsoft.com/en-us/library/aa560610%28v=bts.20%29.aspx

Appendix C - HowTo…

…compare the content of two namespaces (or two directory trees) ?

Generate a list of all objects for each of the namespaces (or directory trees):

```
C:> hcpt --user ns1 --password ns101 -v -l hcp1.log -i3 list --cluster \
    ns1.matrix.hcp1.vm.local --dir /rest --database hcp1_db --keepDB --fatDB --nooutfile

C:> hcpt --user ns1 --password ns101 -v -l hcp2.log -i3 list --cluster \
    ns1.matrix.hcp2.vm.local --dir /rest --database hcp2_db --keepDB --fatDB --nooutfile
```

You'll end up with two files, hcp1_db and hcp2_db, each containing a SQLite3 database. You'll also have two logfiles, hcp1.log and hcp2.log, at whose end you'll find the number of objects found.

Now, prepare a command file 'cmd.sqlite':

```
attach database hcp2_db as hcp2;
.header on
.output compare.txt
select min(urlname) from
(
  Select urlname, size, retentionstring
  from main.flist
  Union all
  Select urlname, size, retentionstring
  from hcp2.flist
) tmp
group by urlname
having count(*) = 1
order by urlname;
```

Run SQLITE3.exe to find the differences between the namespaces (or directory trees):

```
C:> sqlite3 -init cmd.sqlite hcp1_db
```

You'll end up with a file 'compare.txt', holding the paths/names of those objects available in only one namespace (or directory tree).