qq and jq Basics:

Guide to enabling qq autocomplete and aliases:

https://docs.qumulo.com/administrator-guide/qq-cli/enabling-autocomplete.html

List all `qq` commands:

```
qq -h | less
```

Human Readable output vs --json:

Example:

```
qq smb_list_shares
```

vs

```
qq smb_list_shares --json
```

Quick overview of using jq and finding keys:

Find the command name (“quotas” here):

```
qq -h | grep quotas
```

Find the keys in a JSON file: (`jq` only accepts JSON as input)

```
qq quota_list_quotas | jq keys
```

```
qq quota_list_quotas | jq '.[]|keys'
```

Sample qq Tasks:

Find the serial numbers and state of all disks in the cluster:

```
qq cluster_slots | jq
'sort_by(.node_id)|.[]|{id,disk_serial_number,state}'
```

Get a full report in CSV format:

```
qq cluster_slots | jq -r '([.[0]|keys[]]),(.[]|[.[]])|@csv' >
~/Desktop/drive_report.csv
```

Learn about the cluster's current Protection State and Restriper Status:

```
qq protection_status_get

qq restriper_status
```

List all connected clients:

```
qq network_list_connections -c (And without -c for JSON)
```

List all SMB sessions:

```
qq smb_list_sessions | jq '.session_infos[]'
```

List Open SMB File Handles that match a pattern:

Search for the pattern "home" in path names

```
qq smb_list_file_handles -p | jq
'.file_handles[].handle_info|select(.path|test("home"))|{owner,path,l
ocation}'
```

Find the User owner of the open File Handle (User auth ID):

```
qq smb_list_sessions --identity "auth_id:25769805128"
```

Close open file handle using the location from the above command:

```
smb_close_file_handle --location "1.5007655.84717"
```

Find largest snapshot:

```
qq snapshot_get_capacity_used_per_snapshot | jq
'[.entries[]]|max_by(.capacity_used_bytes|tonumber)'
```

Find the 3 largest snapshots:

```
qq snapshot_get_capacity_used_per_snapshot | jq
'[.entries[]]|sort_by(.capacity_used_bytes|tonumber)|.[-3:]'
```

Convert to CSV:

```
qq snapshot_get_capacity_used_per_snapshot | jq -r
'[.entries[]]|sort_by(.capacity_used_bytes|tonumber)|reverse|["Used
Bytes","Snap ID"],(.[]|[.capacity_used_bytes,.id])|@csv' >
~/Desktop/snaps.csv
```

Find exceeded quotas:

```
qq quota_list_quotas | jq
'.quotas[]|select((.capacity_usage|tonumber) > (.limit|tonumber))'
```

Find a snapshot by name pattern match:

```
qq snapshot_list_snapshots | jq
'.entries[]|select(.name|test("joe"))'
```

Print report on Replication statuses as a CSV:

Source Replication relationships:

```
qq replication_list_source_relationship_statuses | jq -r
'[.[]]|["Status","Enabled?","Target Cluster","Target Address","Source
Path","Target Path","Last
Error"],(.[]|[.state,.replication_enabled,.target_cluster_name,.targe
t_address,.source_root_path,.target_root_path,.error_from_last_job])|
@csv' > ~/Desktop/replication.csv
```

Target Replication relationships:

```
qq replication_list_target_relationship_statuses | jq -r
'[.[]]|["Status","Enabled?","Target Cluster","Target Address","Source
Path","Target Path","Last
Error"],(.[]|[.state,.replication_enabled,.target_cluster_name,.targe
t_address,.source_root_path,.target_root_path,.error_from_last_job])|
@csv' > ~/Desktop/target_replication.csv
```

Piping `qq` output to other shell utilities:

This example finds over-limit quotas and converts units to human-readable format - This makes use of the GNU "`numfmt`" utility to convert integers to MB/GB/TB, etc and will create a new JSON object.

Note: Mac users should install the GNU Core Utils (`brew install coreutils`)

```
quotas=$(qq quota_list_quotas | jq
'.quotas[]|select((.capacity_usage|tonumber) > (.limit|tonumber))') |
echo "{ \"human_readable_limit\":" "\"$(echo $quotas | jq -r .limit |
numfmt --to=si)\",  \"human_readable_capacity\":" "\"$(echo $quotas |
jq -r .capacity_usage | numfmt --to=si)\" }" | jq
```

Sample input:

```
{
  "capacity_usage": "413310976",
  "id": "6302866",
  "limit": "50000000",
  "path": "/home/joe/100k/"
}
```

Sample output:

```
{
  "human_readable_limit": "50M",
  "human_readable_capacity": "414M"
}
```

Running qq inside of shell scripts:

This example runs the exceeded quotas example above and converts the numbers to a human-readable format.

Run the script with the address of the cluster and the local path to the credentials file:

```
/path/to/script.sh my.qumulo.com ~/.qfsd_creds
```

Bash Script:

```
#!/bin/bash
host=$1
credentials=$2

quotas=$(qq --host $host --credentials-store $credentials quota_list_quotas
--page-size 100 | jq '[.quotas[]|select((.capacity_usage|tonumber) >
(.limit|tonumber))]')
array_length=$(echo $quotas | jq .[].limit | wc -l)

convert_format() {
new_num=$(numfmt --to=si $1)
echo $new_num
}

for i in $(seq 0 $(($array_length - 1 )))
do
        limit_human=$(convert_format $(echo $quotas | jq -r .[$i].limit))
        capacity_human=$(convert_format $(echo $quotas | jq -r .[$i].capacity_usage))
        path=$(echo $quotas | jq -r .[$i].path)
        printf '{"path": "%s", "capacity_used": "%s","quota_limit": "%s"}\n' $path
$capacity_human $limit_human
done
```