Hasqd Network Commands

29 August 2016, ver 0.4.2

Introduction

The main way to interact with hasqd server is to use *network commands*. They are normally issued remotely and used for manipulating database records, for retrieving information or for changing hasqd server behaviour.

All commands fall into one of the two categories: unprivileged and privileged. Unprivileged commands form main interface that users and developers utilize to communicate with hasqd. Privileged commands are restricted to hasqd server administrator only. They are used for performing administrative tasks.

Hasqd server recognises three protocols in which network commands can be issued: hasqd protocol, HTTP GET request based and HTTP POST request based. HTTP-based protocols are provided for the ease of communication with hasqd servers using web browsers. Hasqd protocol is more direct and designed to be used mostly by third party developers.

Descriptions of protocols in the following document sections share some commonality outlined below.

1) Quotes do not form part of a request. They are shown to highlight fields that include a mandatory whitespace character.

2) **CRLF** and **CRLFCRLF** are sequences of characters with ASCII codes 13(**CR**) and 10(**LF**).

3) *hasqd_command* is a list of words separated by whitespaces. Whitespace is defined as in the Standard C++ Library. The following example demonstrates how hasqd extracts separate words from *hasqd_command:*

```
istringstream is(hasqd_command);
for(string w; is >> w; ) {...}
```

Hasqd protocol

The format of a network request in this protocol is as follows.

hasqd_command **CRLFCRLF**

The format of a hasqd reply is below:

hasqd_reply **CRLFCRLF**

HTTP-based protocols

The format of HTTP GET and POST based requests in simplified form is as follows.

"**GET** /" hasqd_command " **HTTP"** *http_version http_headers* **CRLFCRLF**

"**POST** /" *http_version CRLF http_headers* "**Content-Length: "** http_packet_length *http_headers* **CRLFCRLF** http_packet

where http_packet is: *[...&]***command=**hasqd_command*[&...]*

Notes

1) Bold font indicates parts of a request that must be present in order for hasqd server to accept it. Italic font indicates fields that are ignored by hasqd if present.

2) http_packet_length is a number that represents the length of http_packet in bytes. If missing or doesn't match the actual length of http_packet, the request will be discarded.

Examples

GET /info db HTTP/1.0CRLFCRLF

GET /data _md5db 098f6bcd4621d373cade4e832627b4f6 HTTP/1.0CRLFCRLF

POST / HTTP/1.1CRLF**Host: 10.233.214.182**CRLF**Content-Length: 8**CRLFCRLF**job 1332**

POST / HTTP/1.0CRLF**Content-Length: 4**CRLFCRLF**ping**

Upon extracting hasqd_command from the incoming request, hasqd executes it. The result of the execution is sent back to a user in the following format.

"HTTP/1.0 200 OK" CRLF "Server: " server_name **CRLF "Access-Control-Allow-Origin: *" CRLF "Content-Type: "** mime_type **CRLF "Content-Length: "** hasqd_reply_length **CRLFCRLF** hasqd_reply **CRLFCRLF**

Notes

1) Bold font indicates fixed parts of a hasqd response. Normal font indicates variable parts.

2) hasqd_reply_length is a number that represents the length of hasqd_reply in bytes.

3) hasqd_reply contains the result of execution of the received network command.

Example of user-server communication

User (issues network command **ping**):

GET /ping HTTP/1.0CRLFCRLF

Hasqd reply (**OK**):

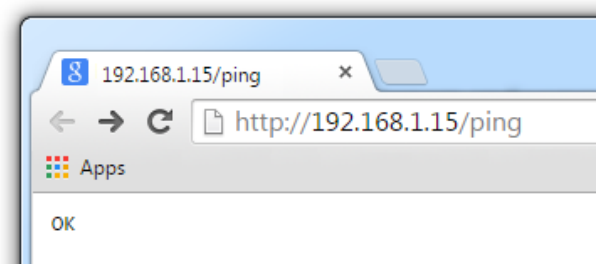
> **HTTP/1.0 200 OK**CRLF**Server: Hasq server 0.3.0 (Win_x86) Hasq Technology Pty Ltd (C) 2013-2015**CRLF**Access-Control-Allow-Origin: ***CRLF**Content-Type: text/plain**CRLF**Content-Length: 2**CRLFCRLF**OK**CRLFCRLF

Web browsers and communication with hasqd servers

GET HTTP request is a simple way to issue a hasqd command using web browser. Hasqd network commands can simply be typed into the address bar of a web browser which will then take care of constructing a proper GET request. Upon receiving hasqd reply, browsers extract and display hasqd_reply part of a response.

Example

Checking availability of a hasqd server running at 192.168.1.15 using **ping** command typed in the address bar of a web browser with the reply displayed in the web browser window.



Unprivileged hasqd network commands

Command (Alias)	Description
/	**Format**: */path* **Description**: get content of file or directory **Arguments**: *path* - path to file or directory to read **Return**: content of file or directory **Example**: /dir1/ <html><head><title>Hasq server 0.3.0 (Win_x64) Hasq Technology Pty Ltd (C) 2013-2015</title></head><body> Hasq server 0.3.0 (Win_x64) Hasq Technology Pty Ltd (C) 2013-2015 db.traits </body></html>
add	**Format**: add *S Db N Dn K *(G) O [D]* **Description**: Add a new record to a chain **Arguments**: *S* - signature of the core of a record being added. The core includes *N S K *(G) O* fields. Signature is the first several (from 1 to all) hex digits taken from the hash value of the record's core. For calculating purposes core fields must be separated by exactly one space character (ASCII code 32). That is *S=first-few-hex-digits-of-Hash(N+" "+S+" "+K+" "+*(G)+" "+O)*. *S* can be specified as " * " in which case its value will be ignored, otherwise it will be re-calculated. If newly calculated value differs from *S*, hasqd returns error. *Db* - database *Dn* belongs to *N* - new record number except 0 (must be equal to *last-record-number-in-a-chain* + *1*) *Dn* - digital token whose chain will receive a new record *K* - record's Key **(G)* - record's Generators (number of *G*'s depends on database configuration) *O* - record's Owner *[D]* - optional data (must contain printable characters only) **Return**: OK *job-id* *job-id* - job id for further enquires **Example**: add * _wrd 3 fd80 f602 e1a1 57b1 Hello OK 1001

data	**Format**: data *Db Dn* **Description**: get last record data **Arguments**: *Db* - database *Dn* belongs to *Dn* - digital token whose last record data is requested **Return:** OK *last-record-data* **Example**: data _wrd fd80 OK Hello, World!
drop	**Format**: drop *X Data* | drop *get X* **Description**: set variable value | get variable value **Arguments**: *X* – variable name (a-z, A-Z, 0-9, '.', '-', '_'. Max length 128) *Data* – variables value ('!'-'~' ASCII 33-126 plus space 32, max 10k) *get* – service command **Return:** OK | OK *value* **Example**: drop X data OK drop get X OK data
file	**Format**: file *path* | f *path* **Description**: get content of file or directory **Arguments**: *path* - path to file or directory **Return:** *content-of-file* | OK *content-of-directory-in-text-format* **Note:** if *path* specifies directory name, it must end with '/' **Example**: file dir1/ OK dir11 dir12

first	**Format**: first *Db Dn* **Description**: get first record in a chain **Arguments**: *Db* - database *Dn* belongs to *Dn* - digital token whose first record is requested **Return:** OK *first-record-in-a-chain* **Note:** first record number may differ from 0 depending on database configuration **Example**: first _wrd fd80 OK 3 fd80 f602 e1a1 57b1 Hello, World!
html	**Format**: html *path* | h *path* **Description**: get content of file or directory in html format **Arguments**: *path* - path to file or directory **Return:** OK *file-or-directory-content* **Note:** if *path* specifies directory name, it must end with '/' **Example**: html dir1/ OK <html><head><title>Hasq server 0.3.0 (Win_x64) Hasq Technology Pty Ltd (C) 2013-2015</title></head><body> Hasq server 0.3.0 (Win_x64) Hasq Technology Pty Ltd (C) 2013-2015 db.traits </body></html>

info db	**Format**: info db
	Description: get properties of all server databases
	Arguments: none
	Return: OK { *db1-info* } { *db2-info* } ...
	db1-info, db2-info ... - database properties in the following format:
	{ name=*db-name* hash=*db-hash-type* description=*db-description* nG=*number-of-G's* magic=[*magic-string*] size=*slice-size-kB* thin=*0|1* datalimit=*data-limit-bytes-or-hashes* altname=*hash* }
	db-hash-type - md5, sha1, sha2-256 ...
	data-limit-bytes-or-hashes - record data limit specified in bytes or hashes; -1 means no limit
	Example:
	info db
	OK
	{
	name=_md5base
	hash=md5
	description=MD5
	nG=1
	magic=[magic]
	size=10
	thin=0
	datalimit=-1
	altname=b11a0f22557e9e0fae4ef66cd5df56d2
	}

info id	**Format**: info id **Description**: get server info **Arguments**: none **Return:** OK *server-info* *server-info = server-name server-family server-version shared-keys public-key* **Example**: info id OK i580 Family: [] Version: 0.2.5 SkcKeys: 0 PblicKey: 0,0

info nbs	**Format**: info nbs **Description**: get server's neighbours **Arguments**: none **Return:** OK *neighbour-1 neighbour-2 ...* *neighbour-1, neighbour-2 ...* - neighbour's ip-addresses and ports in the following format: *ip-address:port* **Example**: info nbs OK 298.10.115.125:13131 116.110.28.14:13131
info fam	**Format**: info fam **Description**: get known family servers **Arguments**: none **Return:** OK *server-1 server-2 ...* *server-1, server-2 ...* - family server info in the following format: *name ip-address:port status* where *status* = N|F A|D U|L N|F - neighbour(N) or family(F) server A|D - alive(A) or dead(D) server U|L - unlocked(U) or locked(L) server **Example**: info fam OK v1 127.0.0.1:13141 N A L v2 127.0.0.1:13142 N A U v6 127.0.0.1:13146 N A U v8 127.0.0.1:13148 N D L v3 127.0.0.1:13143 F D U v4 127.0.0.1:13144 F A U v5 127.0.0.1:13145 F A U

info log agent	**Format**: info agent **Description**: get information about operations performed by agent **Arguments**: none **Return:** OK time > agent command time : agent command response **Example**: info log agent OK 101035 > fs mk aaa 101035 : OK 101035 > fs rm aaa 101036 : OK 101040 > cf db md5.db 101041 > cf db
info log conflict	Get conflict log info
info log connect	Get connect log info
info log critical	Get critical errors log info
info sys	**Format**: info sys **Description**: get server system info (disk and memory usage, CPU load) **Arguments**: none **Return:** OK *system-info* **Example**: info sys OK Dsk usg: 61466 M Dsk tot: 79999 M Mem usg: 2848 M Mem tot: 7851 M Cpu load: 2 %

job	**Format**: job *job-id* **Description**: get job's queue status **Arguments**: *job-id* - job id whose status is requested **Return:** OK | *error-code* **Example**: job 1001 WRONG_SEQ_NUMBER job 1000 OK job 10 JOB_NO_INFO
last	**Format**: last *Db Dn* **Description**: get last record in a chain **Arguments**: *Db* - database *Dn* belongs to *Dn* - digital token whose last record is requested **Return:** OK *last-record-in-a-chain* **Example**: last _md5 fd80 OK 7 fd80 f602 e1a1 57b1
lastdata	**Format**: lastdata *Db Dn N* **Description**: get last available data **Arguments**: *Db* - database *Dn* belongs to *Dn* - digital token whose last available data is requested *N* - record number to search from; search starts from *N* and goes backwards up to X records where X=100 by default or set by a server administrator using *lastdata_max* command line option during hasqd start **Return**: OK *N' [data]* *N'* - record number where data is found; if no data found *N'* specifies the last checked record number *data* - data found (if any) **Example**: lastdata _wrd f5b4 14 OK 6 Test data

net protocol	**Format**: net protocol [protocol] **Description**: notification about newly added record **Arguments**: *Db* - database *Dn* belongs to *N* - new record number *Dn* - digital token whose chain received a new record *server* – name of the server who added record (optional)**.** It is checked for presence in family. **Return**: none **Note**: this notification triggers the receiving server to request the new record from *server* in order to add it to its database **Example**: note _md5 13 f084e1b2bfd3eda3689bed4d069b6df4 server1
note	**Format**: note *Db N Dn [server]* **Description**: notification about newly added record **Arguments**: *Db* - database *Dn* belongs to *N* - new record number *Dn* - digital token whose chain received a new record *server* – name of the server who added record (optional)**.** It is checked for presence in family. **Return**: none **Note**: this notification triggers the receiving server to request the new record from *server* in order to add it to its database **Example**: note _md5 13 f084e1b2bfd3eda3689bed4d069b6df4 server1
ping	**Format**: ping **Description**: check server availability **Arguments**: none **Return**: OK **Example**: ping OK

range	**Format**: range *Db fromN toN Dn* **Description**: get a range of records **Arguments**: *Db* - database *Dn* belongs to *fromN* - first record number in the range *toN* - last record number in the range *Dn* - digital token whose records are requested **Return**: 1) OK *N list-of-records* Normal return. *N* - number of records which would be returned if no restriction on number of records is applied *list-of-records* - records which fall into specified range minus restricted. 2) IDX_HIGH Returned when *fromN* is higher than the highest record number in the *Dn*'s chain 3) IDX_NEG Returned when *toN* is negative 4) NO_RECS No records in specified range (for thin databases only) 5) BAD_RANGE Returned when *fromN* is higher than *toN* **Note**: The returned range may contain less elements than requested due to the restriction on the maximum number of records which can be extracted by this command or due to the number of records in a chain being less than requested. The default maximum number of records which can be extracted by *range* command is 100. This limit can be changed by *range_max* command line option during hasqd start. The limit is always applied from *toN* (or the last record number in a chain if *toN* exceeds it) backwards. **Example**: range _wrd 6 10 f5b4 OK 3 7 f5b4 1af5 26f4 88bc Data-7 8 f5b4 49a3 ab77 1367 Data-8 Note: this return could be possible if the following conditions are met: a) the last record in the chain has number 8 (hence 'OK 3' since there are only 3 records available starting from 6: 6, 7, 8), b) the maximum number of records which can be extracted by *range* is set to 2 (hence only 2 records in the returned list)

record	**Format**: record *Db N Dn* **Description**: get record **Arguments**: *Db* - database *Dn* belongs to *N* - record number *Dn* - digital token whose record is requested **Return**: 1) OK *record* Normal return 2) IDX_NEG Returned if *N* is negative 3) IDX_HIGH Returned if *N* exceeds the last *Dn*'s record number 4) NO_RECS Returned if no record with number *N* is found (thin databases only) **Example**: record _wrd 10 96f0 10 96f0 3421 0000 67a4 Data
slice	**Format**: slice *Db* | *Db check Slice* | *Db get Slice* **Description**: get record **Arguments**: *Db* – database name *Slice* – slice name **Return**: 1) for "*slice Db"*: OK Slice (last slice name) 2) for "*slice Db check Slice"*: OK | REQ_FILE_BAD 3) for "slice *Db get Slice"*: Returns content of slice or empty **Example**: 1) slice md5.db OK 20160101-1 2) slice md5.db check 20160101-1 OK slice md5.db check 20160101-2 REQ_FILE_BAD 3) slice md5.db get 20160101-1 0 0cc1 0000 2e6c adb2 1 0cc1 3217 345d 64e1

zero	**Format**: zero *S Db N Dn K *(G) O [D]* **Description**: create zero record (start new chain) **Arguments**: *S* - signature (see *add* command for more information) *Db* - database *Dn* belongs to *N* - must be 0 for this command *Dn* - digital token whose chain will be created *K* - record's Key **(G)* - record's Generators (number of *G*'s depends on database configuration) *O* - record's Owner *[D]* - optional data (must contain printable characters only) **Return**: OK *job-id* *job-id* - job id for further enquires **Example**: zero * _wrd 0 3421 0000 96f0 0c01 Data OK 1000

Privileged hasqd network commands

Command (Alias)	Description
admin disable net	**Format**: admin disable net **Description**: disables a server **Arguments**: none **Return**: OK **Example**: admin disable net OK ping DISABLED
admin enable net	**Format**: admin enable net **Description**: enables a server **Arguments**: none **Return**: OK **Example**: admin enable net OK ping OK
admin reorg	**Format**: admin reorg **Description**: forces a server to reorganise its connections **Arguments**: none **Return:** OK **Example**: admin reorg OK
admin skc add	**Format**: admin skc add **Description**: add a new key to the list of keys **Arguments**: key **Return:** OK **Example**: admin skc add SKCKEY OK

admin skc pop	**Format**: admin skc pop **Description**: remove the oldest key from the list **Arguments**: none **Return:** OK **Example**: admin skc pop OK
admin skc show	**Format**: admin skc show **Description**: show encrypted keys **Arguments**: none **Return:** OK **Example**: admin skc show OK b4ecba2798cf9270802bc56932dd526e60c549c4f1595c3545e6749eadbeb235
connect	**Format**: connect *family-server-ip-address:port* **Description**: inform a server about another family server; the family server will be ping'ed and if it's alive, it will be added to a list of known family members. In case of the server's list of neighbours getting short, this family member may be added to the list of neigbours **Arguments**: IP-address:port **Return:** OK **Example**: connect srv1.hasq.org:13135 OK
unlink	**Format**: unlink *family-server-ip-address:port* **Description**: inform a server about unlink another family or neighbour server; it will be removed from a list of known family members or neighbours list. **Arguments**: IP-address:port **Return:** OK **Example**: unlink srv1.hasq.org:13135 OK

pleb	**Format**: pleb *sub-command* [*arguments*] **Description**: interact with the server file system **Arguments**: *sub-command* - app *filename base64-data* | clean | del *file-or-directory-name* | exec *directory filename* | get *filename* | list *path* | mkdir *directory-name*| put *filename base64-data* **Return:** OK *data* **Example**: pleb mkdir dir1 OK pleb list dir1 OK dir1/
quit	**Format**: quit **Description**: shut down the server **Arguments**: none **Return:** OK **Example:** quit OK

The latest version of this document can be downloaded from http://hasq.org