HASQ Servant Language

29 August 2016, ver 0.4.2

HASQ Servant Language is a script language which Servant executes if script specified with "script" command line option.

The source consists of a list of statements. Each statement is one line. Each statement is constructed recursively from functions. Each function has:

- a name
- arguments, and
- return value.

All multiple spaces suppressed to one space
IMPORTANT! All return values are strings. All tokens must be separated by a space.
All arguments must be separated by space from brackets.
An escape symbol is \
If there is an escape symbol \ then an ASCII code is expected.

Command	**Description**	**Examples**	**Return**
[	Verbatim		
{	Function sequence with separator		
<	Function sequence without separator (glue)		
arg	Returns the value of the n-th element of string or variable	<svt> x = [a b] <svt> y = arg 0 x <svt> print y a <svt> y = arg 1 x <svt> print y b <svt> print arg 10 x ARG_INDEX_OUT	
Variable, =	Creates a variable, assigns the value, and places it into the internal table of variables. Each variable then becomes a function returning its value.	A = [aaa] print A	aaa

Verbatim or [ ]	Returns verbatim string consisting of all arguments with a space between them.	[aaa bbb]	aaa bbb
conupdate	Updates connection	conupdate	
del	Removes a variable from the table.	A = [aaa] del A show	
define	Processes the input data (only variables) in accordance with a certain rule and returns the processing result. The result can be variables or performing of other functions such as "print".	<svt>define address A B C : C = < A [:] B > <svt>A = [127.0.0.1] <svt>B = [13131] <svt>C = address A B C <svt>print { C } 127.0.0.1:13131	
expect	Running next command N times while M field of answer is not X	<svt> print ex 0 OK 100 tcp self [ping] OK This mean "tcp self [ping] 100 times until 0 field of answer will be OK"	
file		<svt> print tcp self [file home/] OK index.html	
for	For is iterative loop. It creates a variable, assigns a value and runs its body.	<svt> print for i 1 10 { i } 12345678910	
hash	Returns hash from record	<svt> print tcp self { [last _md5] hash wrd [rdn1] } OK 1 73a0 eee1 b4b1 bf91	
net protocol	Sets and displays the current data transfer protocol (*hasq*, *http_post*, *http_get*)	<svt> print net protocol hasq <svt> print net protocol [http_post] hasq -> http_post	
nb	Assign locked neighbour.	<svt> nb 127.0.0.1:13131	

println	This function prints the value of its arguments, one value per line. Beware! This function is greedy: it consumes all the following arguments. This is important when used inside begin-end clause. To prevent its greediness use either **print** or **{ println x }**	println [aaa bbb] [ccc]	aaa bbb ccc
print	Print value of the argument. It differs from **println** by consuming only one argument. E.g. **println { print [a] [b] }** should print **a a b**	print a b	a b
quit	Terminates the program	quit	
recpwd	Generates records for database.		
replace	replace str [abc] [xyz]podmenit v strochke str vse podstrochki "abc" na "xyz".See more details below.	<svt> print tcp self replace g (FIX OM please explaing what is **g**)[$1] [/] HTTP/1.1 200 OK Server: Hasq server Access-Control-Allow-Origin: * Content-Type: text/html Content-Length: 116	
sendnote	Sends notification. This functions duplicates internal event to send notification when new records added. Maybe deprecated in future.		
show	This function prints all variables from the variable table and their values.	A = [aaa]	A = aaa
sleep	Set sleep time in ms		sleep 1000

tcp	Connects by tcp	<svt> print ex 0 OK 100 tcp node0 [ping] OK

Agent commands

agent filesys / ag fs		
ag fs mk	makes directory	<svt> filesys mk "a" <agt> # filesys mk a <agt> OK
ag fs cat	shows file content	<svt> fs cat "file.log" <agt> # fs cat file.log <agt> file.log 101035 # fs cat file.log
ag fs cp	copys files and dirs	<svt> fs cp "a/file.log" "b/file.log" <agt> # fs cp a/file.log b/file.log <agt> OK 302
ag fs mv	moves files	<svt> fs mv "a/file.log" "b/file.log" <agt> # fs mv a/file.log b/file.log <agt> OK
ag fs rm	remove files and dirs	<svt> fs rm "a/file.log" <agt> # fs rm a/file.log <agt> OK
agent config / ag cf		
ag cf database **ag cf db**	shows or sets database name	<svt> ag config database [md5.db] <agt> # config database md5.db <svt> ag cf db <agt> # cf db <agt> md5.db
ag cf logcomm **ag cf lc**	shows or sets current log level: n - no logs c - log connection d - log drags a - log all (c+d)	<svt> agent config logcomm <agt> # config logcomm <agt> no <svt> agent cf lc [a] <agt> # cf lc a
ag cf logfile **ag cf lf**	shows or sets logfile name	<svt> agent config logfile "agt0.log" <agt> # config logfile agt0.log <svt> agent config logfile <agt> # config logfile <agt> agt0.log

ag cf webpath **ag cf wp**		<svt> agent config webpath "/" <agt> # config webpath / <svt> ag cf wp <agt> # cf wp <agt> /
ag download **ag dl**	downloads slices from specified database	<svt> ag dl 127.0.0.1:13132 160102:160103 "slcs" <agt> # dl 127.0.0.1:13132 160102:160103 slcs <agt> [slice md5.db] -> {OK 20160102-1} <agt> [slice md5.db check 20160102-1] -> {OK} <agt> [slice md5.db get 20160102-1] -> {0 14fc21b590b4d...} <agt> [slice md5.db check 20160102-2] -> {REQ_FILE_BAD} <agt> [slice md5.db check 20160103-1] -> {REQ_FILE_BAD}
ag build **ag bd**	builds tokens list from downloaded slices	<svt> ag build "slcs" "idxs" "tokenlist.txt" [localhost:13131 127.0.0.1:13132] <agt> # build slcs idxs tokenlist.txt localhost:13131 127.0.0.1:13132
agent validation / ag vd		
ag vd check	checks builded tokens list and creates checklist	<svt> agent vd check "tokenlist.bd" "tokenlist.chk" <agt> # vd check tokenlist.bd tokenlist.chk <agt> Server 127.0.0.1:13132 drags on 1fe1f6d2b098d0117afc89e03a64dec3 with -1, needs 0 <agt> Server 127.0.0.1:13132 drags on 8e692b8ce018d1a58146bd2ca8908b3b with -1, needs 0 <agt> Server 127.0.0.1:13133 drags on dd20be44d1d8dcfcf63a69622390ef88 with -1, needs 0
ag vd notify	sends a notification to servers on the list	<svt> agent vd notify "tokenlist.chk" "tokenlist.ntf" <agt> # vd notify tokenlist.chk tokenlist.ntf <agt> [last wrd.db dd20] -> {IDX_NODN} <agt> Server 127.0.0.1:13133 drags on dd20 with -1, needs 0 <agt> [note wrd.db 0 dd20] -> {OK}
ag vd push	pushes a records to the servers on the list	<svt> agent vd push "tokenlist.chk" "tokenlist.psh" "idxs" <agt> # vd push tokenlist.chk tokenlist.psh idxs <agt> [last wrd.db 1fe1] -> {IDX_NODN} <agt> Server 127.0.0.1:13132 drags on 1fe1 with -1, needs 0 <agt> [zero * wrd.db 0 1fe1 0379 238d d734] -> {OK 1000}

ag report **ag re**	reports by the tokens list	<svt> report tokenlist.in <agt> # report tokenlist.in <agt> 3 DNs out-of-sync on: 127.0.0.1:13132
ag sort	sorts list	<svt> sort idxs <agt> # sort idxs