Changes in 3.5

<https://code.kx.com/q/releases/ChangesIn3.5/>  
Production release date

2017.03.15

Update 2019.11.13

[.z.ph](https://code.kx.com/q/ref/dotz/#zph-http-get) now calls [.h.val](https://code.kx.com/q/ref/doth/#hval-value) instead of [value](https://code.kx.com/q/ref/value/), to allow users to interpose their own evaluation code. .h.val defaults to value.

Enhanced debugger

In V3.5, the debugger has been extended to include the backtrace of the q call stack, including the current line being executed, the filename, line and character offset of code, with a visual indicator (caret) pointing to the operator which failed. The operator and arguments may be captured programmatically for further propagation in error reporting. Backtraces may also be printed at any point by inserting the .Q.bt[] command in your code. Please see [here](https://code.kx.com/q/basics/debug/) for further details.

Concurrent memory allocator

V3.5 has an improved memory allocator which allows memory to be used across threads without the overhead of serialization, hence the use-cases for *peach* now expand to include large result sets.

**Note**

Kdb+ manages its own heap, using thread-local heaps to avoid contention. One complication of thread-local heaps is how to share allocations between threads, and avoid ballooning of allocated space due to the producer allocating in one arena and the consumer freeing that area in another arena. This was the primary reason for serialization/deserialization of the results from slave threads to the main thread when *peach* completes. This serialization and associated overhead has now been removed.

Socket sharding

V3.5 introduces a new feature that enables the use of the SO\_REUSEPORT socket option, which is available in newer versions of many operating systems, including Linux (kernel version 3.9 and later). This socket option allows multiple sockets (kdb+ processes) to listen on the same IP address and port combination. The kernel then load-balances incoming connections across the processes.

When the SO\_REUSEPORT option is not enabled, a single kdb+ process receives incoming connections on the socket.

With the SO\_REUSEPORT option enabled, there can be multiple processes listening on an IP address and port combination. The kernel determines which available socket listener (and by implication, which process) gets the connection. This can reduce lock contention between processes accepting new connections, and improve performance on multicore systems. However, it can also mean that when a process is stalled by a blocking operation, the block affects not only connections that the process has already accepted, but also connection requests that the kernel has assigned to the process since it became blocked.

To enable the SO\_REUSEPORT socket option, include the new *reuseport* parameter (rp) to the listen directive for the \p command, or -p command-line arg. e.g.

q)\p rp,5000

Use cases include coarse load-balancing and HA/failover.

**Note**

When using socket sharding (e.g. -p rp,5000) the Unix domain socket (uds) is not active; this is deliberate and not expected to change.

Number of slaves

Slave threads can now be adjusted dynamically up to the maximum [specified on the command line](https://code.kx.com/q/basics/syscmds/#s-number-of-slaves). A negative number indicates that processes should be used, instead of threads.

q)0N!("current slave threads";system"s");system"s 4";0N!("current,max slave threads";system"s";system"s 0N"); / q -s 8

("current slave threads";0i)

("current,max slave threads";4i;8i)

q)system"s 0" / disable slave threads

q)system"s 0N" / show max slave threads

8i

Improved sort performance

Kdb+ uses a hybrid sort, selecting the algorithm it deems best for the data type, size and domain of the input. With V3.5, this has been tweaked to improve significantly the sort performance of certain distributions, typically those including a null. e.g.

q)a:@[10000001?100000;0;:;0N];system"t iasc a" / 5x faster than V3.4

Improved search performance

V3.5 significantly improves the performance of bin, find, distinct and various joins for large inputs, particularly for multi-column input. The larger the data set, the better the performance improvement compared to previous versions. e.g.

q)nn:166\*n:60000;V1.50?V2.neg[100]?`2;t1:`c1`c2`c3#n?t2:([]c1:`g#nn?V1.c2:nn?V1.c3:nn?V2.val:nn?100);system"ts t1 lj 3!t2" / 100x faster than V3.4

q)a:-1234567890 123456789,100000?10;b:1000?a;system each("ts:100 distinct a";"ts:1000 a?b") / 30% faster than V3.4

SSL/TLS

Added hopen timeout for TLS, e.g.

q)hopen(`:tcps://myhost:5000:username:password;30000)

NUCs – not upwardly compatible

We have tried to make the process of upgrading seamless, however please pay attention to the following NUCs to consider whether they impact your particular installation

* added ujf (new keyword) which mimics the behaviour of uj from V2.x, i.e. that it fills from lhs, e.g.
* q)([a:1 2 3]b:2 3 7;c:10 20 30;d:"WEC")~([a:1 2]b:2 3;c:5 7;d:"WE")ujf([a:1 2 3]b:2 3 7;c:10 20 30;d:" C")
* constants limit in lambdas reduced from 96 to 95; could cause existing user code to signal a 'constants error, e.g.
* q)value raze"{",(string[10+til 96],\:";"),"}"
* now uses abstract namespace for Unix domain sockets on Linux to avoid file permission issues in /tmp. N.B. hence V3.5 cannot connect to V3.4 using UDS. e.g.
* q)hopen`:unix://5000
* comments are no longer stripped from the function text by the tokenizer (-4!x); comments within functions can be stripped explicitly from the -4! result with
* q){x where not(1<count each x)&x[;0]in" \t\n"} -4!"{2+ 3; /comment\n3\n\t/ another comment\n \n/yet another\n /and one more\n}"
* the structure of the result of value on a lambda, e.g. value {x+y}, is:
* (bytecode;parameters;locals;(namespace,globals);constants[0];…;constants[n];m;n;f;l;s)

where

| this | is |
| --- | --- |
| m | bytecode to source position map; -1 if position unknown |
| n | fully qualified (with namespace) function name as a string, set on first global assignment, with @ appended for inner lambdas; () if not applicable |
| f | full path to the file where the function originated from; "" if not applicable |
| l | line number in said file; -1 if n/a |
| s | source code |

This structure is subject to change.

Suggested upgrade process

Even though we have run a wide range of tests on V3.5, and various customers have been kind enough to repeatedly run their own tests during the last few months of development, customers who wish to upgrade to V3.5 should run their own tests on their own data and code/queries before promoting to production usage. In the event that you do discover a suspected bug, please email [tech@kx.com](mailto:tech@kx.com)