AqBanking - Bug #187

Nach Migration von aq5 dauert das Finden von Aliasen sehr lange

01/30/2020 01:07 PM - rhabacker

Status:	Feedback	**Start date:**	01/30/2020
Priority:	Normal	**Due date:**	
Category:			
Betriebssystem:	Linux	**Anwendung:**	KMyMoney
AqBanking-Version:	6.0.2.0	**Version der Anwendung:**	4.8.5

Description

Nach einer Migration von aq5 erhalte ich bei einem älteren Laptop bei jedem Wechsel einer Ansicht in kmymoney die folgenden Meldungen.

```
3:2020/01/30 12-38-47:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-47:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-47:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-47:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-47:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-47:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-48:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-48:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-48:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-48:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-48:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-48:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-49:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-49:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-49:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-49:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-49:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-49:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-50:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-50:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-50:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-50:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-50:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-50:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
```

```
1"
3:2020/01/30 12-38-51:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-51:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-51:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-51:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-51:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-52:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-52:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-52:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
3:2020/01/30 12-38-52:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "82096d
96-867e-4c6c-a35e-9a5ded777668-A000001"
3:2020/01/30 12-38-52:aqbanking(1603):banking_compat.c:   62: Invalid account id for alias "A00000
1"
```

Wenn ich kmymoney mit gdb starte und während der Verarbeitung unterbreche, ist das immer innerhalb von GWEN_FSLock_Unlock(lck), was über mehrere frames von AB_Banking_GetAccountSpecByAlias() augefrufen wird.

History

#1 - 01/30/2020 01:20 PM - ipwizard

Schau dir mal https://cgit.kde.org/kmymoney.git/commit/?id=bb6cf996c8c2b63b4cb0e387dc4024413d95dfa3 an. Das sollte die Anzahl dieser Meldungen reduzieren.

#2 - 01/30/2020 11:13 PM - rhabacker

- File 0001-Cache-app-db-config-to-speed-up-checking-account-ali.patch added

> Schau dir mal https://cgit.kde.org/kmymoney.git/commit/?id=bb6cf996c8c2b63b4cb0e387dc4024413d95dfa3 an. Das sollte die Anzahl dieser Meldungen reduzieren.

Danke für den Tip, ich probier das mal aus.

Ich vermute aber, das dadurch nur die Anzahl der gemappten Konten erhöht wird, bei der das wieder ein Problem werden könnte, weil es ja mit aq5 kein Problem war und der Zugriff jetzt in gewissen Fällen langsamer geworden ist.

Bevor ich von den Anpassungen an kmymoney wusste, hatte ich bereits eine alternative Lösung für aqbanking, nämlich einen Lesecache (siehe angehängten Patch), der das Problem für mich löste. Der benötigt aber wahrscheinlich noch Überprüfungen, ob der Cache auf wirklich an allen relevanten Stellen invalidiert wird und eventuell Ergänzungen, das kein inkonsistenter Datenbestand entsteht.

PS: Ich verlinke den zugehörigen Bug mal als Referenz hier hin https://bugs.kde.org/show_bug.cgi?id=416963

#3 - 01/30/2020 11:55 PM - martin

- Status changed from New to Feedback

An sich hatte ich diese API in einer frueheren Version schon mal rausgeschmissen, weil ich der Meinung bin, das Mapping sollte von seiten der Anwendung kommen. In AqBanking wollte ich eigentlich gar keine Anwendungsdaten mehr speichern.

Ich hatte diese beiden Funktionen dann aber doch dringelassen, weil KMyMoney die noch verwendet(allerdings als einzige Anwendung) und ich da - mangels intimer Kenntnisse von KMYMoney - keine schnelle alternative Loesung fuer hatte, als ich die AQB6-Patches fuer KMyMoney gemacht habe.

Langfristig faende ich es aber weiterhin besser, KMyMoney wuerde die Zuordnung bei sich selbst speichern, vor allem auch, weil das Mapping vermutlich auch nur sicher funktioniert, solange man es nicht fuer weitere KMyMoney-Dateien verwendet (zumindest fuer die einfachen IDs, mit den GUIDs sollte es da ja keine Kollisionen geben).

Daher wuerde ich - wenn ueberhaupt - den vorgeschlagenen Cache auch nur fuer diese beiden Alias-Funktionen verwenden wollen, und da dann vielleicht spezifisch einen Cache fuer die Aliases machen wollen. Das generelle Caching der App-Daten waere mir zuviel, weil die Idee mit dem Locking ja gerade war, dass sich die Daten auch ausserhalb der gerade aufrufenden Instanz aendern koennen sollen (z.B. bei CLI-Anwendungen).

Da die Alias-Funktionen aber nur noch von KMyMoney verwendet werden, und davon ja selten zwei Instanzen gleichzeitig laufen duerften, waere ein derart eingeschraenkter Cache also ein guter Kompromiss, sollte es ohne diesen wirklich zu spuerbaren Verzoegerungen kommen.

Allerdings: Diese Verzoegerungen kann ich mir nicht so recht erklaeren... Warum dauert hier das Locking so lange? Eigentlich sollte das ja doch recht fix gehen...

Gruss
Martin

#4 - 01/31/2020 07:17 AM - ipwizard

Ich halte diesen Cache für überflüssig und nur eine weitere Fehlerquelle. Die 'alte' Logik von KMyMoney hat für jeden einzelnen Account und jede Kategorie bei AqBanking angefragt. Da können schon mal leicht 100 Abfragen zusammenkommen. Das macht aber nur Sinn, wenn überhaupt ein Mapping existiert. Darauf achtet die neue Logik und reduziert die Aufrufe auf das Notwendige. Damit besteht das Problem der Laufzeit nicht mehr und es braucht keinen zusätzlichen Cache.

Ich werde mir das mit der API mal in KBanking anschauen und versuchen diese Aufrufe zu vermeiden/zu umgehen, so dass die entfernt werden könnten. Das ganze sollte allerdings transparent und rückwärtskompatibel funktionieren.

#5 - 01/31/2020 08:30 AM - rhabacker

> Das generelle Caching der App-Daten waere mir zuviel, weil die Idee mit dem Locking ja gerade war,
> dass sich die Daten auch ausserhalb der gerade aufrufenden Instanz aendern koennen sollen (z.B. bei CLI-Anwendungen).

Genau das meinte ich mit den Überprüfungen bzw. Ergänzungen.

> Allerdings: Diese Verzoegerungen kann ich mir nicht so recht erklaeren... Warum dauert hier das Locking so lange? Eigentlich sollte das ja doch recht fix gehen...

Es wäre natürlich besser und einfacher, wenn man das mit dem Cache nicht bräuchte und das eigentliche Problem der Zeitverzögerung finden und beheben könnte. Da es mit aq5 und der alten kmymoney Implementierung keine derartigen Probleme gab, stellt sich die Frage, ob da diesbezüglich was im aq/gw Unterbau verändert wurde.
Da ich ja den Fall nachstellen kann: Was würde helfen, das zu analysieren - ein strace-Ausschnitt und/oder Call-Stack ?

#6 - 01/31/2020 08:17 PM - rhabacker

- File 0001-abtest-add-test16-for-AB_Banking_GetAccountSpecByAli.patch added

- File abtest.conf added

Mit einer kompilierten Version on abtest, die den angehängten patch enthält und der Konfigurationsdatei in ~/.aqbanking/settings6/apps/abtest.conf erhalte ich folgende Werte:

aq6:
time ./abtest test16 100
real 0m2,901s
user 0m0,038s
sys 0m0,141s

time ./abtest test16 1000
real 0m26,662s
user 0m0,210s
sys 0m0,849s

aq5:
time ./abtest test16 100
real 0m0.116s
user 0m0.088s
sys 0m0.008s

time ./abtest test16 1000
real 0m0.185s
user 0m0.154s
sys 0m0.029s

#7 - 01/31/2020 11:19 PM - martin

Danke Dir, werde den Test uebernehmen und bei mir auch mal probieren.

Interessant waere hier wohl am ehesten ein strace, weil man da vielleicht sehen kann, wie lange der wo steckt...

Seltsam ist ja, dass der FSLock-Code seit Jahren unveraendert ist, es sollte sich hier also eigentlich kein Unterschied ergeben. Also muss der Zeitbandit irgendwo anders stecken...

Eventuell waere hierzu parallel eine Konsolenausgabe mit AQBANKING_LOGLEVEL=info oder gar debug sinnvoll...

Gruss
Martin

#8 - 01/31/2020 11:31 PM - martin

Moin,

also bei mir sieht das Ergebnis so aus:

```
$ time ./abtest test16 100
Creating AB_Banking...
Initializing AB_Banking...
Fetching alias...
Deinitializing AB_Banking...
Freeing AB_Banking...
Finished

real    0m0,270s
user    0m0,064s
sys    0m0,029s
```

#9 - 01/31/2020 11:44 PM - martin

ipwizard schrieb:

> Ich halte diesen Cache für überflüssig und nur eine weitere Fehlerquelle. Die 'alte' Logik von KMyMoney hat für jeden einzelnen Account und jede Kategorie bei AqBanking angefragt. Da können schon mal leicht 100 Abfragen zusammenkommen. Das macht aber nur Sinn, wenn überhaupt ein Mapping existiert. Darauf achtet die neue Logik und reduziert die Aufrufe auf das Notwendige. Damit besteht das Problem der Laufzeit nicht mehr und es braucht keinen zusätzlichen Cache.
>
> Ich werde mir das mit der API mal in KBanking anschauen und versuchen diese Aufrufe zu vermeiden/zu umgehen, so dass die entfernt werden könnten. Das ganze sollte allerdings transparent und rückwärtskompatibel funktionieren.

Du muesstest Dir letztlich nur in den KMM-Konten die AqBanking-ID des assoziierten AqBanking-Kontos merken, also ein neues Feld einfuegen. Wie macht Ihr die Zuordnung denn mit Eurem OFX-Plugin?

Gruss
Martin

#10 - 01/31/2020 11:50 PM - rhabacker

Auffällig ist, das an zwei Stellen ein select() aufgerufen wird, was die Verzögerungen verursacht und in aq5 nicht vorhanden ist.

```
     0.000046 openat(AT_FDCWD, "/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717", O_RDWR|O_CREAT|
O_TRUNC, 0600) = 3
     0.038796 close(3)                  = 0
     0.000085 stat("/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717", {st_mode=S_IFREG|0600, st_s
ize=0, ...}) = 0
     0.000117 link("/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717", "/home/yyy/.aqbanking/setti
ngs6/apps/abtest.conf.lck") = -1 EEXIST (File exists)
     0.003188 stat("/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717", {st_mode=S_IFREG|0600, st_s
ize=0, ...}) = 0
     0.000109 unlink("/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717") = 0
     0.045817 select(1, NULL, NULL, NULL, {tv_sec=0, tv_usec=750000}) = 0 (Timeout)
     0.751014 openat(AT_FDCWD, "/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717", O_RDWR|O_CREAT|
O_TRUNC, 0600) = 3
     0.009039 close(3)                  = 0
     0.000219 stat("/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717", {st_mode=S_IFREG|0600, st_s
ize=0, ...}) = 0
     0.000182 link("/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717", "/home/yyy/.aqbanking/setti
ngs6/apps/abtest.conf.lck") = -1 EEXIST (File exists)
     0.003524 stat("/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717", {st_mode=S_IFREG|0600, st_s
ize=0, ...}) = 0
     0.000099 unlink("/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717") = 0
     0.003620 select(1, NULL, NULL, NULL, {tv_sec=0, tv_usec=750000}) = 0 (Timeout)
     0.751157 openat(AT_FDCWD, "/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717", O_RDWR|O_CREAT|
O_TRUNC, 0600) = 3
     0.021678 close(3)                  = 0
     0.000227 stat("/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717", {st_mode=S_IFREG|0600, st_s
ize=0, ...}) = 0
     0.000225 link("/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717", "/home/yyy/.aqbanking/setti
ngs6/apps/abtest.conf.lck") = -1 EEXIST (File exists)
     0.003103 stat("/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717", {st_mode=S_IFREG|0600, st_s
ize=0, ...}) = 0
     0.000081 unlink("/home/yyy/.aqbanking/settings6/apps/abtest.conf.lck.xxx-8717") = 0
     0.003070 select(1, NULL, NULL, NULL, {tv_sec=0, tv_usec=750000}) = 0 (Timeout)
     0.751345 stat("/home", {st_mode=S_IFDIR|0755, st_size=4096, ...}) = 0
     0.000447 stat("/home/yyy", {st_mode=S_IFDIR|0755, st_size=16384, ...}) = 0
```

#11 - 01/31/2020 11:54 PM - rhabacker

```
(gdb) bt
#0  0x00007ffff534afb0 in __GI___select (nfds=1, readfds=0x0, writefds=0x0, exceptfds=0x0, timeout=timeout@ent
ry=0x7fffffffc9d0) at ../sysdeps/unix/sysv/linux/select.c:41
#1  0x00007ffff58d0f4a in GWEN_Socket_Select (rs=rs@entry=0x0, ws=ws@entry=0x0, xs=xs@entry=0x0, timeout=timeo
ut@entry=750) at inetsocket.c:500
#2  0x00007ffff58ce244 in GWEN_FSLock_Lock (fl=fl@entry=0x681eb0, timeout=timeout@entry=60000, gid=gid@entry=0
) at fslock.c:405
#3  0x00007fffe9aef876 in GWEN_ConfigMgrDir_LockGroup (cfg=<optimized out>, groupName=0x7ffff7b3f313 "apps", s
ubGroupName=0x660d70 "abtest") at cfgdir.c:599
#4  0x00007ffff79e22e0 in  () at /usr/lib64/libaqbanking.so.44
#5  0x00007ffff79ef58a in AB_Banking_GetAccountSpecByAlias () at /usr/lib64/libaqbanking.so.44
#6  0x0000000000404a7d in test16 (argc=<optimized out>, argv=<optimized out>) at abtest.c:1125
#7  0x00007ffff527bf8a in __libc_start_main (main=0x401f90 <main>, argc=3, argv=0x7fffffffda68, init=<optimize
d out>, fini=<optimized out>, rtld_fini=<optimized out>, stack_end=0x7fffffffda58)
    at ../csu/libc-start.c:308
#8  0x00000000004021da in _start () at ../sysdeps/x86_64/start.S:120
```

#12 - 01/31/2020 11:58 PM - rhabacker

```
GWEN_FSLOCK_RESULT GWEN_FSLock_Lock(GWEN_FSLOCK *fl, int timeout, uint32_t gid)
{
...

      /* sleep for at most GWEN_GUI_CHECK_PERIOD */
      if (doWait)
        GWEN_Socket_Select(0, 0, 0, distance);
```

#13 - 02/01/2020 12:02 AM - martin

Der select() ist okay, weil ja bei link() -EEXIST kommt, und dann wartet gwen mittels select einige ms. Aber warum existiert der Link schon? Werden bei Dir die symlinks bei unlink() erst verzoegert entfernt??

#14 - 02/05/2020 11:57 PM - rhabacker

> Aber warum existiert der Link schon?

Das ist wohl ein Überbleibsel von abgebrochenen Testen z.B. starten von kmymoney mit gdb und dann abbrechen oder durch einen

Programmabsturz.

Gibt es dafür auch einen Dialog zum Entfernen der Locks, wie ich das bereits an anderen Stellen gesehen habe ?

Nach dem Entfernen eventueller *.lck- Dateien, habe ich das nochmal getestet und leider das gleiche Problem erhalten, was entsteht, wenn das Home-Dir oder ~/.aqbanking auf einem Netzlaufwerk liegt (in meinem Fall ein NFS Mount) und ist damit auch reproduzierbar.

> Werden bei Dir die symlinks bei unlink() erst verzoegert entfernt??

Kann mit dem nfs mount zusammenhängen.

Files

0001-Cache-app-db-config-to-speed-up-checking-account-ali.patch	3.67 KB	01/30/2020	rhabacker
0001-abtest-add-test16-for-AB_Banking_GetAccountSpecByAli.patch	2.6 KB	01/31/2020	rhabacker
abtest.conf	1.36 KB	01/31/2020	rhabacker