Bugs - Bug #1452

virtual session dealock

07/04/2012 06:46 AM - Constantin Asofiei

Status:	Closed	**Start date:**	07/03/2012
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 07/04/2012 06:49 AM - Constantin Asofiei

- File threaddump-1341287442432.tdump added

[ECF]
Constantin,

This is a server deadlock that has occurred on my system with some as yet unpublished QA vendor changes. It seems to be a race condition triggered by hand-written code which calls RouterSessionManager.connectVirtual on multiple, remote servers from the same session. So, it's definitely a new use case that's triggering it, but it's one we will have to support in the very near future. It doesn't occur that frequently; this is the second time I've seen it in a few weeks.

Although you don't have the code to recreate it yet, hopefully the stack trace and deadlock info can give you a good idea what's going on. Please let me know if you think this will be painful/fragile to fix. The fact that RSM.connectVirtual is itself synchronized and then internally synchronizes on an inherited lock looks suspicious at a glance, but I haven't looked deeply at it.

Thanks,
Eric

#2 - 07/04/2012 06:52 AM - Constantin Asofiei

The problem is not the RSM.connectVirtual; the problems I see are in the virtual session cleanup/disconnect code. The RSM.connectVirtual block in the thread dump is just a side-effect of Reader (for the server-to-server connection) and Conversation threads deadlocking while cleaning up and disconnecting the virtual session (see the last two threads in the thread dump).

The real problem is that both Reader and Conversation threads reach the session.terminate() call in the DirtyShareFactory.unregisterManagerForDatabase method call (for the same remote database), while cleaning up and disconnecting the remote database. I don't know how this can be possible, as the DSF.unregisterManagerForDatabase is sync'ed on the DSF.cache object, and before session.terminate() is called, the session object is removed from the DSF.sessions map (thus should be used by only one of the threads).

I think some other thread manages to call DSF.getManagerInstance (and inject a session object in the DSF.session map) for the remote database while the queue is shutting down.

Note that the Reader/Writer threads for the server-to-server connection are not explicitly named as the Reader/Writer threads for the client-to-server connection are.

#3 - 07/26/2012 10:22 AM - Constantin Asofiei

- Status changed from New to Feedback

- Assignee set to Constantin Asofiei

Needs to be closed, #1455 duplicates it and provides more details.

#4 - 07/26/2012 02:29 PM - Eric Faulhaber

- Status changed from Feedback to Closed

Replaced by #1455.

Files

threaddump-1341287442432.tdump	67.6 KB	07/04/2012	Constantin Asofiei