Database - Bug #11810

READ-JSON transaction scope leak

08/31/2026 09:27 AM - Teodor Gorghe

Status:	Internal Test	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**reviewer:**	Ovidiu Maxiniuc
vendor_id:	GCD	**production:**	No
case_num:		**env_name:**	
version_reported:		**topics:**	
version_resolved:			

Description

History

#1 - 08/31/2026 09:33 AM - Teodor Gorghe

Testcase: readJsonTxLeak.preadJsonTxLeak.p

```
define temp-table tt1 no-undo
   field f1 as character
   field f2 as integer.

define temp-table tt2 no-undo
   field g1 as character.

define variable lc as longchar no-undo.
define variable lk as logical  no-undo.

output to "readjson_tx.out".

lc = '~{"tt1":[~{"f1":"bad","f2":~}]~}'.
lk = temp-table tt1:handle:read-json("LONGCHAR", lc, "EMPTY") no-error.
put unformatted "read-json=" string(lk) skip.

do transaction:
   create tt2.
   tt2.g1 = "x".
end.

put unformatted "done" skip.
output close.
```

Cause: when READ-JSON returns **false**, the JsonImporter doesn't commit/rollback the transaction when READ-JSON is not currently in full transaction. The session inTx remains **true**, which abends on DO TRANSACTION block when beginning a transaction.

logslogs

```
26/08/31 16:29:32.389+0300 |  SEVERE | com.goldencode.p2j.main.StandardServer [StandardServer.invoke()] | Thre
adName:Conversation [00000003:bogus-tg], Session:00000003, ThreadId:00000008, User:bogus | Abnormal end!
java.lang.RuntimeException: invoke() of class com.goldencode.testcases.tests.persistence.temp_table.read_json_
tx_leak.ReadJsonTxLeak and  method execute failed
    at com.goldencode.p2j.util.ControlFlowOps.invokeError(ControlFlowOps.java:8575)
    at com.goldencode.p2j.util.ControlFlowOps.invokeExternalProcedure(ControlFlowOps.java:6756)
    at com.goldencode.p2j.util.ControlFlowOps.invokeExternalProcedure(ControlFlowOps.java:6525)
    at com.goldencode.p2j.util.ControlFlowOps.invoke(ControlFlowOps.java:1379)
    at com.goldencode.p2j.util.ControlFlowOps.invoke(ControlFlowOps.java:966)
    at com.goldencode.p2j.main.StandardServer$LegacyInvoker.execute(StandardServer.java:2742)
    at com.goldencode.p2j.main.StandardServer.invoke(StandardServer.java:2126)
```

```
    at com.goldencode.p2j.main.StandardServer.standardEntry(StandardServer.java:725)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)
    at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43
)
    at java.base/java.lang.reflect.Method.invoke(Method.java:569)
    at com.goldencode.p2j.util.MethodInvoker.invoke(MethodInvoker.java:126)
    at com.goldencode.p2j.net.Dispatcher.processInbound(Dispatcher.java:812)
    at com.goldencode.p2j.net.Conversation.block(Conversation.java:461)
    at com.goldencode.p2j.net.Conversation.run(Conversation.java:254)
    at java.base/java.lang.Thread.run(Thread.java:840)
Caused by: java.lang.reflect.InvocationTargetException
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)
    at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43
)
    at java.base/java.lang.reflect.Method.invoke(Method.java:569)
    at com.goldencode.p2j.util.ControlFlowOps$InternalEntryCaller.invokeImpl(ControlFlowOps.java:9800)
    at com.goldencode.p2j.util.ControlFlowOps$InternalEntryCaller.invoke(ControlFlowOps.java:9757)
    at com.goldencode.p2j.util.ControlFlowOps.invokeExternalProcedure(ControlFlowOps.java:6667)
    ... 15 more
Caused by: java.lang.RuntimeException: Deferred error processing.
    at com.goldencode.p2j.util.TransactionManager$WorkArea.handleDeferredError(TransactionManager.java:11510)
    at com.goldencode.p2j.util.TransactionManager.popScope(TransactionManager.java:5157)
    at com.goldencode.p2j.util.TransactionManager$TransactionHelper.popScope(TransactionManager.java:9356)
    at com.goldencode.p2j.util.BlockManager.doBlockWorker(BlockManager.java:11198)
    at com.goldencode.p2j.util.BlockManager.doBlock(BlockManager.java:1571)
    at com.goldencode.testcases.tests.persistence.temp_table.read_json_tx_leak.ReadJsonTxLeak.lambda$execute$3
(ReadJsonTxLeak.java:51)
    at com.goldencode.p2j.util.Block.body(Block.java:636)
    at com.goldencode.p2j.util.BlockManager.processBody(BlockManager.java:9726)
    at com.goldencode.p2j.util.BlockManager.topLevelBlock(BlockManager.java:9336)
    at com.goldencode.p2j.util.BlockManager.externalProcedure(BlockManager.java:709)
    at com.goldencode.p2j.util.BlockManager.externalProcedure(BlockManager.java:682)
    at com.goldencode.testcases.tests.persistence.temp_table.read_json_tx_leak.ReadJsonTxLeak.execute(ReadJson
TxLeak.java:37)
    ... 22 more
Caused by: java.lang.IllegalStateException: [00000003:00000008:bogus-->local/_temp/primary] scope 2: database
transaction already active
    at com.goldencode.p2j.persist.TxWrapper.begin(TxWrapper.java:516)
    at com.goldencode.p2j.persist.TxWrapper.activate(TxWrapper.java:455)
    at com.goldencode.p2j.persist.TxWrapper$WorkArea.maybeActivateTxWrapper(TxWrapper.java:1188)
    at com.goldencode.p2j.persist.TxWrapper$WorkArea.lambda$beginTx$6(TxWrapper.java:1298)
    at java.base/java.util.LinkedHashMap.forEach(LinkedHashMap.java:721)
    at com.goldencode.p2j.persist.TxWrapper$WorkArea.beginTx(TxWrapper.java:1298)
    at com.goldencode.p2j.persist.TxWrapper$WorkArea.scopeStart(TxWrapper.java:1398)
    at com.goldencode.p2j.util.TransactionManager.processScopeNotifications(TransactionManager.java:8148)
    at com.goldencode.p2j.util.TransactionManager.pushScope(TransactionManager.java:4822)
    at com.goldencode.p2j.util.TransactionManager$TransactionHelper.pushScope(TransactionManager.java:9315)
    at com.goldencode.p2j.util.TransactionManager$TransactionHelper.pushScope(TransactionManager.java:9253)
    at com.goldencode.p2j.util.BlockManager.doBlockWorker(BlockManager.java:11083)
    ... 30 more
```

#2 - 08/31/2026 09:50 AM - Ovidiu Maxiniuc

I think temp-table tt1:read-json() should be run inside a micro-transaction if none is already open. The pattern we use in such cases should close the transaction before the method returns. Secondary note: methods return No logical value instead of 'hard-stopping' the current block.

I scanned the json/xml readers and all seem to rely on transactions being already open and closed by an outer procedure.

#3 - 08/31/2026 09:51 AM - Constantin Asofiei

I think Paul did some changes related to micro-transactions and datasets with relations - maybe this bug comes from there.

Ovidiu: your note to use a full tx (if not already open) works only and only if 4GL does not leave behind already imported records, from before any error is encountered.

#4 - 08/31/2026 09:54 AM - Teodor Gorghe

Constantin Asofiei wrote:

> I think Paul did some changes related to micro-transactions and datasets with relations - maybe this bug comes from there.
>
> Ovidiu: your note to use a full tx (if not already open) works only and only if 4GL does not leave behind already imported records, from before any error is encountered.

I know, I have some changes which I will commit in **11810a**.

#5 - 08/31/2026 10:49 AM - Ovidiu Maxiniuc

Wait, wait, wait... Now it is coming back to me...

We have already this in trunk since r16566. One of the changes was to add/fix transactional reads from JSON files. The prepareTopLevel(), beginTopLevel, rollbackTopLevel, and commitTopLevel() from JsonImport do the transaction management driven by top-level records from the JSON input. When a top-level record is processed a new micro-transaction is opened (beginTopLevel() will check the transaction status and create a new savepoint if already in a transaction).

I think this is one of the testcases (a case of malformed JSON) which we fail to test in r16566. Reading the first record will cause the uTransaction to be open by beginTopLevel(), but the JSON syntax error did not triggered the rollbackTopLevel() so that the transaction remained open.

This is pure theoretical, it needs to be confirmed by debugging the testcase listed in 1st note.

#6 - 08/31/2026 11:00 AM - Teodor Gorghe

- Status changed from New to WIP

- % Done changed from 0 to 100

- reviewer Ovidiu Maxiniuc added

Yes, but this is the case when rollbackTopLevel or commitTopLevel doesn't get executed, which leads to session inTx = true flag set after READ-JSON execution.
Check 11810a/r16731.

#7 - 08/31/2026 11:01 AM - Teodor Gorghe

- Status changed from WIP to Review

#8 - 08/31/2026 03:49 PM - Ovidiu Maxiniuc

- Status changed from Review to Internal Test

I think 11810a/r16731 is good.

#9 - 09/01/2026 06:30 AM - Teodor Gorghe

Tested on all projects which uses READ-JSON and testing has passed.