Runtime Infrastructure - Bug #7291

LOG-MANAGER to log on client-side

04/21/2023 01:57 AM - Galya B

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Galya B	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to Runtime Infrastructure - Bug #5703: rationalize, standardize and s...	**Closed**
Related to Base Language - Feature #3853: implement LOG-MANAGER runtime	**New**

History

#1 - 04/21/2023 01:57 AM - Galya B

- Related to Bug #5703: rationalize, standardize and simplify the client-side log file name configuration added

#3 - 04/21/2023 01:59 AM - Galya B

- Related to Feature #3853: implement LOG-MANAGER runtime added

#4 - 04/21/2023 02:39 AM - Galya B

Initial state and facts on LOG-MANAGER (LegacyLogManagerImpl):

- lives server-side;
- receives some logs from the server Conversation thread with WRITE-MESSAGE and some logs from the client-side ErrorManager;
- with #3853 it will implement the entry types 4GLMessages, 4GLTrace, QryInfo, ASDefault, ASPlumbing, DB.Connects (appserver and database logging), so more logs will come from the server-side;
- its lifecycle is handled on the server - instantiated as ContextLocal by LegacyLogOps and reset by ContextLocal with appserver type State-reset;
- its state is handled on the server - it can be configured by setting OE attributes (LOGFILE-NAME, LOGGING-LEVEL, LOG-ENTRY-TYPES, SESSION:DEBUG-ALERT) programmatically in OE procedures, which is executed on the server Conversation thread. Every change to a configuration changes substantially the behavior of the log manager;
- appserver processes keep each only one instance of LOG-MANAGER for all their clients;
- multiple clients in OE (processes in FWD) can share the same log file, if no rotation is enabled, without the files being broken;
- multiple clients in OE (processes in FWD) cannot share the same log file, if rotation is enabled. The second client doesn't log.

Task at hand:

It's requested in #5703#note-271 LOG-MANAGER to move to logging on the clients instead of on the server.

Approach:
???

Questions to answer:

- LOG-MANAGER can't live client-side (check facts section), then how does it make the file system checks, which is the core of its rotation functionality (and not only)?
- Do we implement slow OS level shared locks on each file write to not break logs? Current syncing method between processes LegacyLogWriteExecutor should be abandoned.

#5 - 04/21/2023 07:43 AM - Greg Shah

- Start date deleted (04/21/2023)

Create the minimum API which will be a new remote service exported from the client side. This will provide the remote file system checks and the ability to write already formatted output into the log file on the client. All other processing stays on the server.

> Do we implement slow OS level shared locks on each file write to not break logs? Current syncing method between processes LegacyLogWriteExecutor should be abandoned.

Client side logging is much less likely to have more than one process writing to the same file. It is still possible but it is a low priority case. Can we use the existing code to implement it? If so, let's retain the feature so that we have full compatibility.

#6 - 04/21/2023 07:57 AM - Galya B

Greg Shah wrote:

> Client side logging is much less likely to have more than one process writing to the same file. It is still possible but it is a low priority case. Can we use the existing code to implement it?

Current sync between clients relies on data structures and mechanisms in one JVM. If each client writes itself its logs, then only OS level shared lock will work (or something we don't have as an option, a separate shared JVM/service deployed on the same machine that takes care of all writes).

So at this point I guess the best option is to hope the customer sets proper unique numbers in the freely modifiable name.

#7 - 04/21/2023 08:15 AM - Greg Shah

Just document the file locking restriction for now.

#8 - 05/04/2023 07:19 AM - Galya B

If remote clients' file systems are accessible through the OS Resources as expected from #4065, then LOG-MANAGER can keep its centralized nature on the server and only write to the remote hosts.

Then we can think of a way to synchronize file locking on the remote hosts as well, but there are several prerequisites and some limitations:

- Hosts should be uniquely identifiable - do we have this at the moment?
- If another non-FWD process (e.g. OE) writes to the same files, logs will still get corrupted.

#9 - 05/05/2023 03:08 AM - Galya B

- Assignee set to Galya B

- Status changed from New to WIP

#10 - 05/05/2023 07:57 AM - Galya B

I need a clarification on OSResourceManager.isServerSideFileSystem. filesystem is enabled by being included in the directory.xml value of server-side-resources. ClientSideResourceManager is calling isServerSideFileSystem on initializeFileSystem. For standalone clients this should always be false, because directory.xml are on the server. Does it mean standalone clients are always creating a local instance of FileSystemDaemon?

#11 - 05/05/2023 08:05 AM - Greg Shah

> If remote clients' file systems are accessible through the OS Resources as expected from #4065, then LOG-MANAGER can keep its centralized nature on the server and only write to the remote hosts.

When you say "remote clients", are you referring to those FWD client JVMs which run on a different machine (physical or virtual) than the machine on which the FWD server JVM runs?

Whether a resource is server-side or client-side really has nothing to do with whether a client is remote or local to the server system. If for a given session, server-side filesystem support is enabled, then all filesystem usage will be on the server, even if this is a "remote client".

#12 - 05/05/2023 08:08 AM - Greg Shah

> I need a clarification on OSResourceManager.isServerSideFileSystem. filesystem is enabled by being included in the directory.xml value of server-side-resources. ClientSideResourceManager is calling isServerSideFileSystem on initializeFileSystem. For standalone clients this should always be false, because directory.xml are on the server. Does it mean standalone clients are always creating a local instance of FileSystemDaemon?

What do you mean by "standalone clients"? We have a "single" mode which is probably broken and which we never use. It probably should be removed since it complicates things and it is not clear if we will ever pursue the idea. Is that what you mean by standalone?

#13 - 05/05/2023 08:10 AM - Galya B

Greg Shah wrote:

> What do you mean by "standalone clients"?

Not web/appserver/scheduled batch/single running on the server, but launched by scripts manually.

#14 - 05/05/2023 08:11 AM - Galya B

Greg Shah wrote:

> If for a given session, server-side filesystem support is enabled, then all filesystem usage will be on the server, even if this is a "remote client".

So server-side-resources is not context-local and can't be different for each client?

#15 - 05/05/2023 08:12 AM - Galya B

Also in the context of logging it matters if it's a remote client (on a different host) or not. I spoke about synchronization between processes. Sorry if the term bothers you.

#16 - 05/05/2023 08:13 AM - Galya B

Galya Bogdanova wrote:

> I need a clarification on OSResourceManager.isServerSideFileSystem. filesystem is enabled by being included in the directory.xml value of server-side-resources. ClientSideResourceManager is calling isServerSideFileSystem on initializeFileSystem. For standalone clients this should always be false, because directory.xml are on the server. Does it mean standalone clients are always creating a local instance of FileSystemDaemon?

My question is how the logic in ClientSideResourceManager.initializeFileSystem works, when the client can't read the directory.xml configs.

#17 - 05/05/2023 08:15 AM - Greg Shah

> > If for a given session, server-side filesystem support is enabled, then all filesystem usage will be on the server, even if this is a "remote client".
>
> So server-side-resources is not context-local and can't be different for each client?

It is context-local. That is what I mean when I say "for a given session". Using context-local is how we make something session-specific.

#18 - 05/05/2023 08:16 AM - Galya B

Greg Shah wrote:

> It is context-local. That is what I mean when I say "for a given session". Using context-local is how we make something session-specific.

How do you make it different for different gui clients, if it's defined in one place in directory.xml?

#19 - 05/05/2023 08:19 AM - Greg Shah

> Also in the context of logging it matters if it's a remote client (on a different host) or not. I spoke about synchronization between processes.

The objective stands. If server-side is being used, then we write on the server, otherwise we write on the client. In regard to locking, if we have to use OS level locks to get the right behavior with multiple processes, then so be it.

> Sorry if the term bothers you.

No bother. I just want us to be talking about the same thing.

#20 - 05/05/2023 08:20 AM - Greg Shah

> > I need a clarification on OSResourceManager.isServerSideFileSystem. filesystem is enabled by being included in the directory.xml value of server-side-resources. ClientSideResourceManager is calling isServerSideFileSystem on initializeFileSystem. For standalone clients this should always be false, because directory.xml are on the server. Does it mean standalone clients are always creating a local instance of FileSystemDaemon?
>
> My question is how the logic in ClientSideResourceManager.initializeFileSystem works, when the client can't read the directory.xml configs.

The client has a session with the server and uses an up-call to read values from the directory.

#21 - 05/05/2023 08:21 AM - Greg Shah

> It is context-local. That is what I mean when I say "for a given session". Using context-local is how we make something session-specific.

> How do you make it different for different gui clients, if it's defined in one place in directory.xml?

It can be defined at different levels (global default, per-server, group, account).

#22 - 05/05/2023 08:22 AM - Greg Shah

Galya Bogdanova wrote:

> Greg Shah wrote:
>
> > What do you mean by "standalone clients"?
>
> Not web/appserver/scheduled batch/single running on the server, but launched by scripts manually.

I wouldn't say "standalone". These are just clients not launched by the spawner. Logging behavior should not be different based on using the spawner or not using the spawner.

#23 - 05/05/2023 08:23 AM - Galya B

Greg Shah wrote:

> I wouldn't say "standalone". These are just clients not launched by the spawner. Logging behavior should not be different based on using the spawner or not using the spawner.

Yes, you're right. I brought up the question of client permissions before and still forget about it. Even if the client is on the same host, the server cannot take over its functions.

#24 - 05/05/2023 08:25 AM - Galya B

Greg Shah wrote:

> In regard to locking, if we have to use OS level locks to get the right behavior with multiple processes, then so be it.

Please confirm which processes do we aim at:
1. one FWD server and its clients
2. multiple FWD servers and their clients on the same host
3. OE and FWD on the same host
4. everything above and more

#25 - 05/05/2023 08:32 AM - Greg Shah

Option 4 although it is unlikely that OE is ever involved on the same system. It is more likely that the installation may just have other processes involved in log access. For example, they may have scripts or log analysis tools or some other batch processes that they want to log to the same file.

#26 - 05/05/2023 08:48 AM - Galya B

Greg Shah wrote:

> Option 4 although it is unlikely that OE is ever involved on the same system. It is more likely that the installation may just have other processes involved in log access. For example, they may have scripts or log analysis tools or some other batch processes that they want to log to the same file.

Then the only generic option is to go for OS locks, but the Java FileLock API states these locks are not mandatory / exclusive, so some processes can decide to bypass them and not account for them.

> Whether or not a lock actually prevents another program from accessing the content of the locked region is system-dependent and therefore unspecified. The native file-locking facilities of some systems are merely advisory, meaning that programs must cooperatively observe a known locking protocol in order to guarantee data integrity. On other systems native file locks are mandatory, meaning that if one program locks a region of a file then other programs are actually prevented from accessing that region in a way that would violate the lock. On yet other systems, whether native file locks are advisory or mandatory is configurable on a per-file basis. To ensure consistent and correct behavior across platforms, it is strongly recommended that the locks provided by this API be used as if they were advisory locks.

OS locks will probably degrade the performance of logging to the level of Progress.

I was thinking more in line of syncing only FWD processes and that might have worked well. Now each write should be blind about the other writes and release the lock. Probably keeping a queue and writing bigger batches in longer intervals will be the only possible optimization.

#27 - 05/05/2023 08:54 AM - Galya B

Now lets confirm the other side of the matter. Do I understand correctly that when the context-local OSResourceManager says filesystem is enabled server-side, the client should send its logs and leave it to the server to write them on its host and if filesystem is not enabled, clients should write to log files themselves?

#28 - 05/05/2023 09:31 AM - Greg Shah

> Now lets confirm the other side of the matter. Do I understand correctly that when the context-local OSResourceManager says filesystem is enabled server-side, the client should send its logs and leave it to the server to write them on its host and if filesystem is not enabled, clients should write to log files themselves?

Yes, I think so.

Constantin: Can you think of any issues here?

#29 - 05/05/2023 10:17 AM - Galya B

I make it sound way too easy. I will have to rework a lot.

- Currently LegacyLogManagerImpl keeps a reference to the file and checks its size before each write and this is done by the Java implementation itself. Now it will have to be explicit, a rpc call. Actually there is plenty of such operations that will have to be done with rpc now.
- When rotation is enabled for two OE processes, only one of them can write to the same file sequence and it makes sense in any system actually, because it needs to track file size and sequence number. At the moment it works as expected, because the sync is done in one place. With client side writes, the implementation will break. Even if all the logic is still on the server, it will have to either always forbid two clients writing to the same rotated file (no matter the host) or identify the target locations as being on separate hosts and manage them separately.
- With client-side writes there might be lost logs. Even more so if queues and intervals of writing replace the current direct immediate writes (to save on lock/unlock files). The client throws an exception and exits. The server receives the exception and tries to write it back, but it's not possible.

#30 - 05/05/2023 10:25 AM - Galya B

All the logging tasks are trying to make distributed structures that are naturally singleton / monolith. Distributed architectures have their place, but in my opinion this concept doesn't work well, when we put it in one object. I mean it's a different scale, different level of complicating a feature.

#31 - 05/05/2023 10:36 AM - Galya B

A note: appserver clients will be an exception to any rule, no client-side writing for them, because it needs to be one file for all appserver clients.

#32 - 05/05/2023 10:40 AM - Constantin Asofiei

Greg Shah wrote:

> > Now lets confirm the other side of the matter. Do I understand correctly that when the context-local OSResourceManager says filesystem is

> enabled server-side, the client should send its logs and leave it to the server to write them on its host and if filesystem is not enabled, clients should write to log files themselves?

Yes, I think so.

Constantin: Can you think of any issues here?

It makes sense; although originally server-side filesystem was meant for legacy streams and such, client logs (for i.e. LOG-MANAGER) are part of the legacy part. As a side note, we still have the FWD client logs, which will also be on server-side, if server-side filesystem is configured.

#33 - 05/05/2023 10:46 AM - Galya B

Constantin Asofiei wrote:

> As a side note, we still have the FWD client logs, which will also be on server-side, if server-side filesystem is configured.

At the moment that depends on a logging config though.

#34 - 05/05/2023 11:05 AM - Galya B

So how to make LOG-MANAGER distributed?
- The server has the attribute getters, setters, receives logs from different systems and WRITE-MESSAGE;
- The client has the initialization configs, produces the error logs and will need to do file system checks and writes;
- Processes trying to write to the same rotated file should be orchestrated somehow. Everything else is OS lock dependent.

I guess we want to use OS resource 'filesystem' and hide the details of where the operations are executed. So the client will still send the error logs and initialization configs to the server as it does now. Then the server will write it back as filesystem operations. It will be tricky to check the file size before each write, if writes are not immediate. If writes are immediate, then there will be a lot of OS lock/release operations. The current sync logic in LegacyLogWriteExecutor will be removed completely.

Is this how we imagine using #4065?

#35 - 05/05/2023 11:15 AM - Galya B

The other option would be to not use #4065 and filesystem OS resource.

The code in LOG-MANAGER remains the same. When server-side is enabled, it operates like at the moment. When client-side is enabled, the server context-local is another LOG-MANAGER impl, that is an empty shell proxying to the client getter/setter/write calls. On the client resides the actual LOG-MANAGER now. OS locks are still applied, but orchestration of rotated files may need additional though.

That is basically having two modes of operation of LOG-MANAGER.

#36 - 05/05/2023 11:24 AM - Galya B

With option B. syncing the errors of the LOG-MANAGER itself between the server and the client, will also have to be reworked, because they are expected to be thrown server-side (I'll have to look into it).

#37 - 05/08/2023 02:38 PM - Greg Shah

> I guess we want to use OS resource 'filesystem' and hide the details of where the operations are executed. So the client will still send the error logs and initialization configs to the server as it does now. Then the server will write it back as filesystem operations.

I'm not sure this is the way we should go. The filesystem resource is meant for 4GL compatible features such has streams support (e.g. PUT STREAM s ...). Although the LOG-MANAGER does write to the file system, it does so in a lower level manner since we have to support things like log rotation. I don't want to complicate it with trying to reuse the filesystem support.

> The other option would be to not use #4065 and filesystem OS resource.

I think we would use the same configuration capability but the low level implementation does not strictly need to be the same as #4065.

> When client-side is enabled, the server context-local is another LOG-MANAGER impl, that is an empty shell proxying to the client getter/setter/write calls. On the client resides the actual LOG-MANAGER now. OS locks are still applied, but orchestration of rotated files may need additional though.

I think this is mostly the right idea. I do think we might want to use state sync to avoid round trips to the client.

#38 - 05/09/2023 02:11 AM - Galya B

Greg Shah wrote:

> I do think we might want to use state sync to avoid round trips to the client.

With state sync the client and the server will keep each queues of events + log records. On state sync both sources will have to be merged and sorted by time before executed in the order. On shutdown of the client its buffer can be saved to the file, but if it's a disconnect or an unexpected exception on the server, the queue on the server will be lost. This is a note on implementation.

#39 - 05/09/2023 05:21 AM - Galya B

At the moment LOG-MANAGER logs all messages with the client pid. Do we want to differentiate between server source and client source by setting different pids? Is yes, then System.loadLibrary("p2j") will always be executed.

#40 - 05/09/2023 07:14 AM - Greg Shah

> but if it's a disconnect or an unexpected exception on the server, the queue on the server will be lost

Understood. On a positive note, only one of the sync directions will be used in any given session.

> Do we want to differentiate between server source and client source by setting different pids?

Not at this time. I think that would be more confusing than helpful.

#41 - 05/26/2023 08:32 AM - Galya B

How are supposed getters to work with state sync? I can create events for initialization, attr setters, write log record, clear and close LOG-MANAGER, but if these are synced with state sync, how is a getter call from the server working? It should be rpc that waits for the accumulated queue of events to get executed before returning, basically forcing exec of the queue and waiting for the file operations... Is this how I should implement it?

P.S. But if the getter is rpc, how can I be sure the state sync for all events before the getter has already reached the client? Even if I can check if the server queue is empty, what do I do if it's not?

#42 - 05/26/2023 08:45 AM - Galya B

Now that I think more about it, the whole merge-two-sources-of-events-and-execute-them thing is problematic. It can't be on schedule, because it should be in order of occurrence and the server logs may still be on the server. If it's on every state sync received, how can I be sure the logs generated by the client in the meantime should all be executed. The latest client logs might be a result of an event happening after another server event has already been executed on the server during the state sync.

Also there are some client logs happening after the server finishes the state sync. Do these wait for client process finish to be written to the file? Not sure if this can affect long running processes like the appserver client.

#43 - 05/26/2023 10:31 AM - Greg Shah

The state sych honors the order of operations because of these facts:

1. The execution of code is effectively single threaded even though it is implemented in a highly threaded multi-process manner. This means that if the flow of control is on one side of a given user's session (e.g. the client) then no code is executing in that user's session on the other side (e.g. the server). The flow of control only shifts back and forth in well architected ways that require the involvement of the RPC mechanism (either call or return).
2. The RPC mechanism is built on top of a messaging based low level protocol. That protocol has hooks implemented for the state sync. State accumulates only on the side of the network in which the control flow is currently executing. That state will accumulate there until the next time the control flow shifts to the other side (either a call or return).
3. The hooks package up the changes and transparently append them to the message associated with the call or return that shifts the flow of control. These are gathered by the protocol itself, so the caller/returner doesn't know anything about it.
4. Before the call or return is actually allowed to proceed on the other side, the protocol transparently applies the changes so that when the call or return executes, it sees the synchronized state. This maintains all order of operations as one would expect.

#44 - 05/30/2023 05:00 AM - Galya B

State sync doesn't seem to work well. On top of the the six getters, the following methods should also return a value immediately: logical clearLog(), logical writeMessage(), logical closeLog(), because they are legacy LOG-MANAGER method implementations, so they should also be called with rpc. Most importantly writeMessage should return. If writeMessage uses rpc state, then sync won't be useful.

I can mark the calls to writeMessage coming from the FWD systems (at the moment just ErrorManager) and still use state sync for them, but this will lead to some long-term issues:

- LOG-MANAGER future development (adding subsys logs) should be inline with the idea of marking the source instead of using the OE interface freely.
- writeMessage will always return true for FWD calls (subsys logging), so that logs can be handled by the client asynchronously.
- We'll end up with:
  - 3 setters and init method using state sync (each of them critical to proper LOG-MANAGER behavior);
  - 6 getters, close and clear methods using rpc;
  - writeMessage using rpc for direct LOG-MANAGER:WRITE-MESSAGE calls from converted procedures and state sync for subsys logging.

What do you think?

#45 - 05/30/2023 07:51 AM - Constantin Asofiei

Galya, is there any condition which can be raised by any APIs which would use state sync? If this is so, then we must use RPC for them, otherwise the application execution flow will not be matched.

#46 - 05/30/2023 08:09 AM - Galya B

There are no Java exceptions thrown, but there are plenty of OE errors in initialization and the setters.

It just came to my mind why I initially had all entry type issues thrown as errors and then we had to convert them to show only... Man, I said this LOG-MANAGER is tricky... I have to prove it yet, but my bet is that it handles it differently. I mainly tested with command line startup args, then Constantin tested programmatically. It throws errors on init, but then the setters only show warnings.

#47 - 05/30/2023 08:11 AM - Galya B

Galya B wrote:

> but there are plenty of OE errors in initialization and the setters.

... which is (as we already know) java RuntimeException...

#48 - 05/30/2023 08:12 AM - Constantin Asofiei

Galya B wrote:

> ... but there are plenty of OE errors in initialization and the setters.

This is what I meant. It may work if all the validation/checks (which can raise OE errors) are on FWD server side, and the RPC API (or state sync) will include only the final call/state, which is 'safe'.

#49 - 05/30/2023 08:19 AM - Galya B

Constantin Asofiei wrote:

> This is what I meant. It may work if all the validation/checks (which can raise OE errors) are on FWD server side, and the RPC API (or state sync) will include only the final call/state, which is 'safe'.

It sounds good, but the problem it all revolves around the file system.
switchLogFile does file system operations, it's called from setLogFileName, initialize, writeMessage and clearLog.

#50 - 05/30/2023 08:24 AM - Galya B

Greg, it's not that I'm lazy, but it really doesn't make sense to have LOG-MANAGER client-side. It's way too intertwined with both the server life-cycle and the client file system and that's why it works on the the same process/host. Splitting them looks like a hell of a headache. I mean if it's really critical, we'll pull it off somehow, but it's like a whole new feature on itself.

#51 - 05/30/2023 11:37 AM - Galya B

Greg, I think LOG-MANAGER client-side implementation will end up:

- very fragile;
- with multiple rpc calls for one LOG-MANAGER method execution.

Do you want me to continue nevertheless?

#52 - 05/30/2023 12:21 PM - Greg Shah

Pause the work while I consider what we lose by dropping this.

#53 - 05/31/2023 09:06 AM - Greg Shah

We need to move ahead with this task. We cannot assume that all customers can use server-side logging for LOG-MANAGER. It would break compatibility for some use cases.

#54 - 05/31/2023 12:25 PM - Galya B

Then what about making the design as simple and and clear as possible.

- LOG-MANAGER will live on the server and throw exceptions like it knows best, but some file system checks, closeLog and clearLog will be rpc calls to the client.
- On LOG-MANAGER:WRITE-MESSAGE writeLog will execute immediately as rpc.
- Subsys log writes will be sent with state sync and merged and sorted with the log queue generated client-side. writeLog will always return true server-side for FWD/subsys calls (auto logging).
- To save some calls to the client, the check for the file size on each log record write will have to depend on a server-side counter, counting bytes and hoping nothing else writes to the same file on the client side. That is to ensure rotation is on time.
- Rotation to a new file will create an rpc call to the client.
- Without rotation each write will lock the file and release it immediately after.
- With rotation the client will obtain the file lock initially and release it at the end of the process / file log span, because with rotation two processes can't write to the same file in theory. That implementation relies on OS specifics and will not work for exclusive lock anywhere, but it's the best we can do in Java.

I'm sure I haven't thought it well through, but that is the rough plan.

#55 - 05/31/2023 02:09 PM - Greg Shah

I'm fine with the plan.

#56 - 06/01/2023 07:16 AM - Galya B

Can we have clients running in the server JVM? Basically my question is if ThinClient static fields will always be used by only one client instance.

#57 - 06/01/2023 09:46 AM - Greg Shah

> Can we have clients running in the server JVM? Basically my question is if ThinClient static fields will always be used by only one client instance.

Currently, this is the case. We do rely upon this in many places. We have an idea that it is possible to move all of this into the server and it would be useful for web UI cases but we have no plans to implement that mode at this time. It is OK to rely upon this assumption.

#58 - 06/05/2023 01:20 PM - Galya B

A few notes:

- Java file lock is respected by other Java processes on Windows and Linux and they will wait indefinitely for their turn if implemented with FileLock.lock(), which means this allows consecutive writes.
- Java file lock is respected by Progress LOG-MANAGER on Windows, but the Progress process doesn't wait the lock to be released and instead skips writing to the file.
- On Linux simple echo redirected to the file cmd line will not respect the Java file lock.
- On Windows simple echo redirected to the file cmd line will respect the Java file lock and display The process cannot access the file because it is being used by another process..

#59 - 06/06/2023 10:48 AM - Galya B

- % Done changed from 0 to 90

#60 - 06/07/2023 04:40 AM - Galya B

How do I configure OSResourceManager.getInstance().isServerSideFileSystem() in directory.xml?

#61 - 06/07/2023 09:31 AM - Galya B

Galya B wrote:

> How do I configure OSResourceManager.getInstance().isServerSideFileSystem() in directory.xml?

I want to add this to the wiki. If someone can recall where to show me the configs?

#62 - 06/07/2023 09:33 AM - Galya B

- Status changed from WIP to Review

- % Done changed from 90 to 100

7291a rebased on trunk r14616.

7291a r14626 tested and ready for review.

#63 - 06/07/2023 09:34 AM - Galya B

Do I add the file headers before or after the review?

#64 - 06/07/2023 02:21 PM - Greg Shah

Galya B wrote:

> Galya B wrote:
>
> > How do I configure OSResourceManager.getInstance().isServerSideFileSystem() in directory.xml?
>
> I want to add this to the wiki. If someone can recall where to show me the configs?

The following snippet would enable memptr, native library calls and filesystem support (which includes streams access) on the server side. It can be added at the account, group, server or global default level in the directory:

```
<node class="string" name="server-side-resources">
   <node-attribute name="value" value="memptr,library,filesystem">
</node>
```

As can be seen in the javadoc for Utils.getDirectoryNodeWorker(), there is a hierarchy of lookups that occurs:

1. If a user/process or group node is present:
    1. /server/<serverID>/runtime/<account_or_group>/server-side-resources.<project> (honors the project token, if configured)
    2. /server/<serverID>/runtime/<account_or_group>/server-side-resources
2. If no user/process or group nodes are present, then these server-specific nodes are checked:
    1. /server/<serverID>/runtime/default/server-side-resources.<project> (honors the project token, if configured)
    2. /server/<serverID>/runtime/default/server-side-resources
3. If no server node exists, these are checked (it is the global default area for all servers):
    1. /server/default/runtime/<account_or_group>/server-side-resources.<project> (honors the project token, if configured)
    2. /server/default/runtime/<account_or_group>/server-side-resources
4. Finally, if no user/process or group nodes are present in the global default area, then these are checked:
    1. /server/default/runtime/default/server-side-resources.<project> (honors the project token, if configured)
    2. /server/default/runtime/default/server-side-resources.

#65 - 06/07/2023 07:19 PM - Greg Shah

> Do I add the file headers before or after the review?

Generally, before. It helps explain the changes which makes it easier to review.

#66 - 06/08/2023 06:28 AM - Galya B

7291a r14627 file headers.

#67 - 06/08/2023 06:43 AM - Constantin Asofiei

Review for 7291a/14627:

- we don't usually use static imports except for interface constants. Without qualifying a static method, the reader needs to find the import and determine what class is from. Please remove import static com.goldencode.p2j.util.Utils.*; and such. Greg, this is not documented in the coding standards, please correct me if I'm wrong.
- ThinClient calls logManagerService.write(sstate.logManagerRecords);, but sstate.logManagerRecords can be null. write doesn't protect against this.
- Utils.pollAll needs to return null if there is no message, there is no reason to transfer an empty array to the remote side.

#68 - 06/08/2023 06:56 AM - Greg Shah

> Greg, this is not documented in the coding standards, please correct me if I'm wrong.

I think this is correct. An exception would be cases where there is a significant value in the reduction of boilerplate code that outweighs the ambiguity added by removing the class names. We do this in the converted code, for example. In the absence of a clear case otherwise, we do try to avoid such usage.

#69 - 06/08/2023 07:54 AM - Galya B

Constantin, please go ahead with reviewing the rest of the code, when you have time. I'll fix all, when you're done.

#70 - 06/08/2023 09:45 AM - Constantin Asofiei

Galya B wrote:

> Constantin, please go ahead with reviewing the rest of the code, when you have time. I'll fix all, when you're done.

I've looked through all the changes, I don't see anything else wrong. Did you test again all client types (ChUI/GUI, batch and appserver)?

#71 - 06/08/2023 09:46 AM - Galya B

Constantin Asofiei wrote:

> I've looked through all the changes, I don't see anything else wrong. Did you test again all client types (ChUI/GUI, batch and appserver)?

Only batch, appserver and web. I'll see gui and chui next.

#72 - 06/08/2023 10:37 AM - Galya B

Constantin Asofiei wrote:

- Utils.pollAll needs to return null if there is no message, there is no reason to transfer an empty array to the remote side.

null is the abbreviation of NullPointerException, so I'll return Optional from pollAll method and make sure to transfer null with the state sync.

r14628 up. I'm testing gui/chui now.

#73 - 06/08/2023 11:40 AM - Galya B

Pushed a tiny NPE fix in r14629. Tested with gui and chui.

When do I merge?

#74 - 06/08/2023 02:53 PM - Greg Shah

You can merge now.

#75 - 06/09/2023 03:22 AM - Galya B

Task branch 7291a was merged to trunk as rev 14620 and archived. Email sent.

#76 - 06/09/2023 06:32 AM - Galya B

- Status changed from Review to Test

#77 - 06/09/2023 09:03 AM - Galya B

Do I create a whole new branch just for a null check?

#78 - 06/09/2023 09:27 AM - Greg Shah

No, just post the full diff here for a quick review.

#79 - 06/09/2023 09:45 AM - Galya B

- File npe-fix.patch added

Diff attached. Fixes two NPEs when configs not present.

#80 - 06/09/2023 09:50 AM - Greg Shah

In the future, when you post something like this that is expected to be committed immediately, please include history entries.

You can add the history entries now and commit it to trunk. You will still need to send the merge email but the subject will have to be changed a bit.

#81 - 06/09/2023 09:51 AM - Galya B

Greg Shah wrote:

> In the future, when you post something like this that is expected to be committed immediately, please include history entries.

Yep, I saw I missed it after posting the diff.

> You will still need to send the merge email but the subject will have to be changed a bit.

I can't recall if we have a standard one in this situation.

#82 - 06/09/2023 09:53 AM - Galya B

I can name it merge notification for task 7291 if there is no standard one.

#83 - 06/09/2023 09:59 AM - Galya B

Fix merged to trunk as rev 14621.

#84 - 06/09/2023 10:01 AM - Greg Shah

> I can name it merge notification for task 7291 if there is no standard one.

That works.

#85 - 06/11/2023 05:59 AM - Constantin Asofiei

Galya, I missed something. Appserver logging must not be forced on server-side.

```
=== modified file 'src/com/goldencode/p2j/util/LegacyLogManagerConfigs.java'
--- old/src/com/goldencode/p2j/util/LegacyLogManagerConfigs.java        2023-06-08 10:23:19 +0000
+++ new/src/com/goldencode/p2j/util/LegacyLogManagerConfigs.java        2023-06-11 09:51:04 +0000
@@ -253,7 +253,6 @@
      */
     public void setAppServer(String appServerName)
     {
-       this.isServerSideFileSystem = true;
        this.appServerName = appServerName;
        if (logFile == null)
        {
```

#86 - 06/12/2023 02:32 AM - Galya B

Constantin Asofiei wrote:

> Galya, I missed something. Appserver logging must not be forced on server-side.

Well, yes. Can you fix it in one of your branches? I think I won't have any other merges soon.

#87 - 06/13/2023 02:15 AM - Galya B

I'll merge it with 7415c when ready.

#88 - 06/27/2023 09:03 AM - Greg Shah

- Status changed from Test to Closed

The last issue was fixed in task branch 7415c which was merged to trunk as rev 14631.

Files

npe-fix.patch	1.28 KB	06/09/2023	Galya B