Base Language - Feature #3853

implement LOG-MANAGER runtime

01/09/2019 06:13 PM - Greg Shah

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Galya B	**% Done:**	60%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Runtime Infrastructure - Feature #7147: Make FWD log outputs consi...	**New**
Related to Runtime Infrastructure - Bug #7291: LOG-MANAGER to log on client-side	**Closed**
Related to Runtime Infrastructure - Bug #7462: fix 'AS' instead of '4GL' in L...	**New**

History

#1 - 01/09/2019 06:22 PM - Greg Shah

Log Manager Usage

The base implementation is already handled at conversion and it is stubbed out in the runtime. The biggest effort will be writing testcases identify the runtime locations for instrumentation. The used LOG-ENTRY-TYPES does not look very hard to implement.

LOG-MANAGER system handle

Attributes

LOGFILE-NAME
LOGGING-LEVEL
LOG-ENTRY-TYPES
LOG-THRESHOLD
ENTRY-TYPES-LIST
NUM-LOG-FILES
QUERY:BASIC-LOGGING

Methods

WRITE-MESSAGE
CLEAR-LOG
CLOSE-LOG

Some details about the customer application usage:

- LOGGING-LEVEL is set to 2, 3
  - 0 ("none"), means no logging
  - 1 ("errors"), this would log only 4GL error messages and ignores the LOG-ENTRY-TYPES setting
  - 2 ("basic"), includes "errors" plus categories defined in LOG-ENTRY-TYPES
- LOG-ENTRY-TYPES is set to one of the following
  - "QryInfo:2,4GLMessages" (this is used in non-appserver mode)
  - "QryInfo:2,ASDefault" (used in appserver mode)
  - "4GLTrace"
  - "4GLMessages"
  - At least in 1 location it is set to an arbitrary value.
  - 4GLTrace logs on execution of internal procedures, functions, events etc...
  - 4GLTrans logs on transactions and sub-transactions
  - QryInfo logs queries (open query and for each)
  - ASPlumbing logs unspecified appserver events
  - DB.Connects logs database connects/disconnects
  - 4GLMessages logs MESSAGE statement output (the same way that appserver and batch does this without the LOG-MANAGER)
  - session:debug-alert IS explicitly set so 4GLMessages does also include stack traces (yikes!)
  - ASDefault is the same as ASPlumbing + DB.Connects

#2 - 07/25/2019 12:26 AM - Mihai Popescu-Tiganea

I have made few tests who are described below:
NOTE: some procedures will generate a .log file with same name - for inspection and comparison - in log_manager/files directory

- Methods:
- clear-log() - will generate a file:
  - call method w/o setting a log file
  - call method after setting log file and add content - check sizes before and after
- close-log() - will generate a file:
  - call method w/o setting a log file
  - call method after setting a log file
  - call method write-message after close log
- write-message() - will generate a file:
  - call method w/o setting a log file
  - call method after set a log file compare sizes before and after
  - test default and explicit use of subsys expression parameter

- Attributes:
- entry-types-list - no file generated:
  - get list from log-manager
  - test if read-only
- logfile-name - generated file is deleted after test by test procedure:
  - use relative path:
    - check default value
    - create file, add info apply attribute -> check sizes
    - delete file, apply attribute check file then delete again
  - use absolute path:
    - check default value
    - create file, add info apply attribute -> check sizes
    - delete file, apply attribute check file then delete again
  - use wrong path syntax
  - call attribute after close-log() method
- logging-level - generated file is deleted after test by test procedure:
  - get default value
  - apply attribute w/o set log file
  - set logging level out of range
  - delete file used for tests
- log-threshold - no file generated
  - get default value
  - test if read only
- num-log-files - no file generated
  - get default value
  - test if read only
- log-entry-types - default and errors - will generate a file
  - apply attribute w/o set a log file
  - set a log file and apply with a value who is not in the list
- log-entry-types - 4glmessages - will generate a file
  - set logging level to 0 and 1 for this attribute
    - set session:debug-alert to true and false
  - set logging level to 2, 3 and 4 for this attribute
    - set session:debug-alert to true and false
    - apply message with view-as alert-box options
- log-entry-types - 4gltrace_events - will generate a file
  - set logging level from 2 to 4 and
    - call event on run with asynchronous
    - subscribe to an event - procedural way
    - publish event - procedural way
    - subscribe to an event - oo way
    - publish event - oo way
- log-entry-types - 4gltrace - will generate a file
  - set logging level from 2 to 4 and
    - call internal procedure
    - invoke internal function
    - call procedure from external procedure
    - call function from external procedure
    - call trigger
    - call static method from class
    - use method from class
    - instantiate a class
    - use class property
    - call procedure from appserver classic and asynchronous
    - delete object
- log-entry-types - 4gltrans - will generate a file
  - set logging level from 2 to 4 and

        - perform transaction in database
        - perform sub-transaction in database
        - return from sub-transaction with undo
- log-entry-types - db_connects - will generate a file
    - set logging level from 2 to 4 and
        - perform connects and disconnects from database
- log-entry-types - dynobjects - will generate a file
    - set logging level from 2 to 4 and
        - instantiate a lang object
        - run persistent procedure
        - perform an async request
        - memptr - set size
        - create dynamic handles for all options
- log-entry-types - fileid - will generate a file
    - set logging level from 2 to 4 and
        - open and close a file
- log-entry-types - qryinfo - will generate several files
    - set logging level from 2 to 4 and
        - client sort - find with where for non pk field
        - filter - for each/first/last
        - lock - find with no-lock/exclusive-lock/share/lock
        - multiple-index - where for pk and other filed
        - open-query - open dynamically for static and dynamic defined query
        - preselect - use this option on do and repeat
        - scrolling - use for open query
        - whole-index - apply where for non index fields

#3 - 12/03/2019 10:26 AM - Greg Shah

Consider the following log output examples.

```
[19/07/10@11:24:38.981+0300] P-004424 T-002588 1 4GL -- No entry types are activated
[19/07/10@11:24:38.981+0300] P-004424 T-002588 1 4GL -- No entry types are activated
[19/07/10@11:24:38.981+0300] P-004424 T-002588 1 4GL -- Log entry types activated: 4glmessages:2
[19/07/10@11:24:38.982+0300] P-004424 T-002588 2 4GL 4GLMESSAGE     logging-level:2 - debug-alert=true
[19/07/10@11:24:38.982+0300] P-004424 T-002588 2 4GL 4GLMESSAGE     ** ABL Debug-Alert Stack Trace **
[19/07/10@11:24:38.982+0300] P-004424 T-002588 2 4GL 4GLMESSAGE     --> C:\Progress\workspace\testcases\log_ma
nager\attributes\log_entry_types_4glmessages.p at line 177  (C:\Progress\workspace\testcases\log_manager\attri
butes\log_entry_types_4glmessages.r)
```

```
[19/07/11@14:22:04.773+0300] P-011956 T-007000 1 4GL -- No entry types are activated
[19/07/11@14:22:04.773+0300] P-011956 T-007000 1 4GL -- Log entry types activated: 4gltrans:2
[19/07/11@14:22:04.773+0300] P-011956 T-007000 3 4GL 4GLTRANS       BEGIN TRANS 250 [log_manager/common/transa
ctions.p @ 3]
[19/07/11@14:22:04.774+0300] P-011956 T-007000 3 4GL 4GLTRANS       END TRANS 250 [log_manager/common/transact
ions.p @ 16]
[19/07/11@14:22:04.774+0300] P-011956 T-007000 1 4GL -- Log entry types activated: 4gltrans:3
```

```
[19/07/11@14:22:04.774+0300] P-011956 T-007000 3 4GL 4GLTRANS       BEGIN TRANS 260 [log_manager/common/transa
ctions.p @ 3]
[19/07/11@14:22:04.774+0300] P-011956 T-007000 3 4GL 4GLTRANS       BEGIN SUB-TRANS 261 [log_manager/common/tr
ansactions.p @ 6]
[19/07/11@14:22:04.774+0300] P-011956 T-007000 3 4GL 4GLTRANS       BEGIN SUB-TRANS 262 [log_manager/common/tr
ansactions.p @ 7]
[19/07/11@14:22:04.774+0300] P-011956 T-007000 3 4GL 4GLTRANS       BEGIN SUB-TRANS 263 [log_manager/common/tr
ansactions.p @ 9]
[19/07/11@14:22:04.774+0300] P-011956 T-007000 3 4GL 4GLTRANS       UNDO SUB-TRANS 263 [log_manager/common/tra
nsactions.p @ 13]
[19/07/11@14:22:04.774+0300] P-011956 T-007000 3 4GL 4GLTRANS       END SUB-TRANS 262 [log_manager/common/tran
sactions.p @ 14]
[19/07/11@14:22:04.774+0300] P-011956 T-007000 3 4GL 4GLTRANS       END SUB-TRANS 261 [log_manager/common/tran
sactions.p @ 15]
[19/07/11@14:22:04.774+0300] P-011956 T-007000 3 4GL 4GLTRANS       END TRANS 260 [log_manager/common/transact
ions.p @ 16]
```

```
[19/07/15@13:41:17.751+0300] P-013120 T-006964 1 4GL -- Logging level set to = 2
[19/07/15@13:41:17.751+0300] P-013120 T-006964 1 4GL -- No entry types are activated
[19/07/15@13:41:17.751+0300] P-013120 T-006964 1 4GL ----------     Log file closed at user's request
[19/07/15@13:41:17.887+0300] P-013120 T-006964 1 4GL -- Logging level set to = 2
[19/07/15@13:41:17.887+0300] P-013120 T-006964 1 4GL -- No entry types are activated
[19/07/15@13:41:17.887+0300] P-013120 T-006964 1 4GL -- Log entry types activated: qryinfo:2
[19/07/15@13:41:17.895+0300] P-013120 T-006964 2 4GL QRYINFO        Query Info Logging turned on for query stD
efQuery handle 0 QueryId 1570746230280
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        Query Info Logging turned on for query  ha
ndle 1013 QueryId 1570746519680
[19/07/15@13:41:17.896+0300] P-013120 T-006964 1 4GL APPL          open statically - start
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        Query Plan:  log_manager/common/qryinfo_op
en_query.p line 22
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        QueryId: 1570746230280
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        Query Name: stDefQuery
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        Query Handle: 0
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        Type: Statically Opened Query
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        Prepared at Compile time
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        Client Sort: N
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        Scrolling: N
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        Table: C:\Progress\workspace\testcases\fwd
\fwd.customer
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO          Indexes: pk
[19/07/15@13:41:17.896+0300] P-013120 T-006964 1 4GL APPL          open statically - end
[19/07/15@13:41:17.896+0300] P-013120 T-006964 1 4GL APPL          open dynamic for static query - start
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        Query Statistics:  log_manager/common/qryi
nfo_open_query.p line 32
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        QueryId: 1570746230280
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        Query Name: stDefQuery
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        Query Handle: 1015
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        Times opened: 1
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        Used REPOSITION: N
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        DB Blocks accessed:
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO         C:\Progress\workspace\testcases\fwd\fwd :
 0
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        DB Reads:
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO         Table: C:\Progress\workspace\testcases\fw
d\fwd.customer : 0
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO        C:\Progress\workspace\testcases\fwd\fwd.cu
stomer Table:
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO         4GL Records: 0
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO         Records from server: 0
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO          Useful: 0
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO          Failed: 0
[19/07/15@13:41:17.896+0300] P-013120 T-006964 2 4GL QRYINFO         Select By Client: N
[19/07/15@13:41:17.897+0300] P-013120 T-006964 2 4GL QRYINFO        Query Plan:  log_manager/common/qryinfo_op
en_query.p line 32
[19/07/15@13:41:17.897+0300] P-013120 T-006964 2 4GL QRYINFO        QueryId: 1570746230280
[19/07/15@13:41:17.897+0300] P-013120 T-006964 2 4GL QRYINFO        Query Name: stDefQuery
[19/07/15@13:41:17.897+0300] P-013120 T-006964 2 4GL QRYINFO        Query Handle: 1015
[19/07/15@13:41:17.897+0300] P-013120 T-006964 2 4GL QRYINFO        Type: Dynamically Opened Query
[19/07/15@13:41:17.897+0300] P-013120 T-006964 2 4GL QRYINFO        PREPARE-STRING: for each fwd.customer wher
e customerName = "Ion Anton"
[19/07/15@13:41:17.897+0300] P-013120 T-006964 2 4GL QRYINFO        Prepared at Runtime
[19/07/15@13:41:17.897+0300] P-013120 T-006964 2 4GL QRYINFO        Client Sort: N
```

```
[19/07/15@13:41:17.897+0300] P-013120 T-006964 2 4GL QRYINFO        Scrolling: Y
```

I have these questions about the content and formatting of the log entries.

- Is P-999999 an operating system pid?
- Is T-999999 an operating system thread id?
- If this was run under PASOE, would we see output (in a single log file) with the same P-999999 but different T-999999 values?
- The logs look like they were captured in Windows, what do those values look like on Linux?
- What is the one digit number that appears before the text 4GL? At first glance I thought it was the logging level associated with the log entry, but in the second example above the logging level is set to 4gltrans:2 but there is output listed as ... 3 4GL ....
- Why is there a difference in the indention level for the actual log messages? Are certain messages always indented the same amount or is there some meaning to the indention levels?

#4 - 12/04/2019 02:12 AM - Marian Edu

Greg Shah wrote:

> I have these questions about the content and formatting of the log entries.
>
> - Is P-999999 an operating system pid?

yes, not relevant in the log file imho but it's there

> - Is T-999999 an operating system thread id?

same, completely irrelevant though

> - If this was run under PASOE, would we see output (in a single log file) with the same P-999999 but different T-999999 values?

that might, they do say it's a multi-threaded agent but this is probably that is visible in a high load environment not in our development

> - The logs look like they were captured in Windows, what do those values look like on Linux?

not very different, the pid is still on six digits while the thread id is using ten... right now we don't have any linux environment setup for FWD

```
[19/12/04@07:00:36.802+0000] P-000309 T-2426324864 1 4GL -- Logging level set to
 = 2
[19/12/04@07:00:36.802+0000] P-000309 T-2426324864 1 4GL -- Log entry types acti
vated: qryinfo:4
[19/12/04@07:00:36.830+0000] P-000309 T-2426324864 2 4GL QRYINFO    Query Plan:
 test.p line 1
[19/12/04@07:00:36.830+0000] P-000309 T-2426324864 2 4GL QRYINFO    QueryId: 0x1
57f418
[19/12/04@07:00:36.830+0000] P-000309 T-2426324864 2 4GL QRYINFO    Type: FOR St
atement
[19/12/04@07:00:36.830+0000] P-000309 T-2426324864 2 4GL QRYINFO    Client Sort:
```

```
 N
[19/12/04@07:00:36.830+0000] P-000309 T-2426324864 2 4GL QRYINFO    Scrolling: N
[19/12/04@07:00:36.830+0000] P-000309 T-2426324864 2 4GL QRYINFO    Table: db/sp
orts2000.Customer
```

- What is the one digit number that appears before the text 4GL? At first glance I thought it was the logging level associated with the log entry, but in the second example above the logging level is set to 4gltrans:2 but there is output listed as ... 3 4GL ....

That is the log level, it does seems to be an error/bug there since according to this KB entry the begin/end transaction entries should be level 2.
https://knowledgebase.progress.com/articles/Article/P96551

- Why is there a difference in the indention level for the actual log messages? Are certain messages always indented the same amount or is there some meaning to the indention levels?

No idea but there seems to be always only one space between the first five entries then the lines with a specific type there is a space followed by the entry type and then a tab (actually 4 spaces) and whatever detail message that follows. For other non type specific lines there seems to be a space followed by at least two dashes, those aren't really important anyway.

#5 - 12/05/2019 02:40 PM - Greg Shah

As part of this task, we should ensure that the default appserver logging that occurs in the 4GL is fully supported in FWD. This must include suitable control over the log filename and the generation/formatting of the appserver log entries, including timestamps.

#6 - 04/28/2020 08:37 AM - Constantin Asofiei

I've seen something interesting in a logmanager test:

```
def var h as handle.
run cc.p on h asynchronous set h (output i as char).
```

You can inline the var definition, only if the RUN is an async request. I'm changing Marian's tests to explicitly define the var.

#7 - 04/28/2020 08:47 AM - Greg Shah

OK. Please create a task to record the problem so we don't forget it.

#8 - 04/28/2020 09:30 AM - Marian Edu

Constantin Asofiei wrote:

> I've seen something interesting in a logmanager test:
> [...]
>
> You can inline the var definition, only if the RUN is an async request. I'm changing Marian's tests to explicitly define the var.

What test does that exactly Constantin, inline variable definition isn't possible in 4gl... both variables used for `on` and `set` should be defined as handle before. The first one is for the remote server handle where the RPC is made and second is to save a handle to the remote async procedure to use it later. It is not mandatory to save that handle, especially if you event-procedure option is used. Also one can find all pending remote procedures using session system handle and walk the three starting from the first-procedure.

#9 - 04/28/2020 09:33 AM - Constantin Asofiei

Marian Edu wrote:

> What test does that exactly Constantin, inline variable definition isn't possible in 4gl... both variables used for `on` and `set` should be defined as handle before. The first one is for the remote server handle where the RPC is made and second is to save a handle to the remote async procedure to use it later. It is not mandatory to save that handle, especially if you event-procedure option is used. Also one can find all pending remote procedures using session system handle and walk the three starting from the first-procedure.

See log_manager/common/dyn_objects.p and profiler/procedure_class/procedure.p, this statement with charRsp:

```
run pipe_char.p on appsrv asynchronous set hreq event-procedure "onDone" (input 'test', output charRsp as char
acter).
```

#10 - 04/28/2020 09:36 AM - Marian Edu

Constantin Asofiei wrote:

> See log_manager/common/dyn_objects.p and profiler/procedure_class/procedure.p, this statement with charRsp:
> [...]

Yes, both variables are already defined... appsrv is inside the include file connect.i, that might have been the source for confusion, sorry for that.

#11 - 04/28/2020 09:36 AM - Constantin Asofiei

Marian Edu wrote:

> Constantin Asofiei wrote:
>
> > See log_manager/common/dyn_objects.p and profiler/procedure_class/procedure.p, this statement with charRsp:
> > [...]
>
> Yes, both variables are already defined... appsrv is inside the include file connect.i, that might have been the source for confusion, sorry for that.

See the charRsp parameter - that's the inline definition.

#12 - 04/28/2020 09:45 AM - Marian Edu

Constantin Asofiei wrote:

> See the charRsp parameter - that's the inline definition.

Ah, I see... that is a placeholder, it can be a local variable but for async calls when I see a local variable used as an output parameter I start looking for herrings, chances are the developer doesn't know the actual behaviour :(

```
For OUTPUT parameters of an asynchronous remote procedure call only, you can specify parameter-name AS primiti
ve-type-name as a prototype. The parameter-name is an arbitrary place-holder name and primitive-type-name must
 specify the ABL data type of the corresponding OUTPUT parameter in the asynchronous remote procedure. You can
 also specify OUTPUT parameters for an asynchronous remote procedure using a local field, variable, TABLE temp
-table-name, TABLE-HANDLE temp-table-handle, DATASET dataset-name, or DATASET-HANDLE dataset-handle. However,
note that the asynchronous remote procedure does not return any values to OUTPUT or INPUT-OUTPUT parameters on
 the RUN statement. These parameters are place holders only for values returned by the remote procedure to the
 specified event-internal-procedure.
```

#13 - 11/04/2020 06:34 AM - Greg Shah

Please see #4384-396 for some discussion about logging in the built-in OO classes. There will be work there to test and resolve any logging differences.

#15 - 10/29/2021 04:53 AM - Constantin Asofiei

A note for the implementation: considering that the Java client associated with a Web client runs on the same machine with other users (even the same username), logfile-name or the folder where this is output needs to be unique, and not collide with another user's log file.

The FWD implementation can make sure to add a numeric suffix so that it does not collide (like a log rotation), but the application logic should add at least the application username or otherwise create a folder structure to output these logs.

#17 - 06/05/2022 04:13 AM - Constantin Asofiei

-clientlog startup parameter is part of this task:
https://docs.progress.com/bundle/openedge-abl-troubleshoot-applications-117/page/Specifying-the-log-filename.html

#18 - 06/05/2022 04:14 AM - Constantin Asofiei

In 6129a/13924 I've added experimental WRITE-MESSAGE and LOGFILE-NAME, to just dump the messages to server's log.

#19 - 06/06/2022 12:29 PM - Greg Shah

As part of this task we should also support bootstrap configuration overrides that provide an equivalent feature for these 4GL command line options:

- logginglevel
- logfile
- logentrytypes
- logthreshold
- logname
- numlogfiles

#22 - 11/29/2022 08:34 AM - Greg Shah

- % Done changed from 0 to 30

- Assignee set to Galya B

#23 - 12/19/2022 06:35 AM - Galya B

Greg Shah wrote:

> The base implementation is already handled at conversion and it is stubbed out in the runtime.

Does it mean the logic creating messages for logging is already implemented? Can you point me to what classes to look in for it?

> The used LOG-ENTRY-TYPES does not look very hard to implement.

The entry types for the client context are as follows: 4GLMessages, 4GLTrace, 4GLTrans, AiaMgmt, AiaProp, AiaRqst, AiaUbroker, AiaDefault, ASDefault, ASPlumbing, DB.Connects, DS.Cursor, DS.QryInfo, DynObjects.DB, DynObjects.XML, DynObjects.Other, DynObjects.Class, DynObjects.UI, FileID, IgnoredOps, MsgrTrace, NSPlumbing, ProEvents.UI.Char, ProEvents.UI.Command, ProEvents.Other, QryInfo, SAX, Temp-tables, TTStats, UBroker.Basic, UBroker.ClientFSM, UBroker.ServerFSM, UBroker.ClientMsgStream, UBroker.ServerMsgStream, UBroker.ClientMsgQueue, UBroker.ServerMsgQueue, UBroker.ClientMemTrace, UBroker.ServerMemTrace, UBroker.ThreadPool, UBroker.Stats,

UBroker.AutoTrim, UBroker.All, WSADefault, DS.Performance

Do we have that separation in the logic handled at conversion?

#24 - 12/19/2022 06:08 PM - Greg Shah

> The base implementation is already handled at conversion and it is stubbed out in the runtime.

Does it mean the logic creating messages for logging is already implemented? Can you point me to what classes to look in for it?

In the testcases project, there is an example of using WRITE-MESSAGE() (log_manager/methods/write_message.p). I assume that is what you are talking about as "creating messages"?

If you convert that program:

java -classpath p2j/build/lib/p2j.jar com.goldencode.p2j.convert.ConversionDriver F2+M0+CB log_manager/methods/write_message.p 2>&1 | tee "cvt_$(date +%Y%m%d_%H%M%M%S).log"

You can then look at the converted code in src/com/goldencode/testcases/log_manager/methods/WriteMessage.java and compare it to the 4GL.

For example, this section of 4GL code:

```
//set a log file , get size, call method , get size and compare sizes
log-manager:logfile-name = 'log_manager/files/write_message.log'.

//get initial file size
file-info:file-name = 'log_manager/files/write_message.log'.
initFileSize = file-info:file-size.

//write message with default subsys-expression
logicalRsp = log-manager:write-message ('test message') no-error.
```

corresponds to this Java code:

```
        // set a log file , get size, call method , get size and compare sizes
        LegacyLogOps.logMgr().setLogFileName(new character("log_manager/files/write_message.log"));

        // get initial file size
        FileSystemOps.initFileInfo(new character("log_manager/files/write_message.log"));
        initFileSize.assign(FileSystemOps.fileInfoGetSize());

        // write message with default subsys-expression
        silent(() -> logicalRsp.assign(LegacyLogOps.logMgr().writeMessage("test message")));
```

From there I expect you can find the runtime classes in FWD which are referenced.

> The used LOG-ENTRY-TYPES does not look very hard to implement.

> The entry types for the client context are as follows: 4GLMessages, 4GLTrace, 4GLTrans, AiaMgmt, AiaProp, AiaRqst, AiaUbroker, AiaDefault, ASDefault, ASPlumbing, DB.Connects, DS.Cursor, DS.QryInfo, DynObjects.DB, DynObjects.XML, DynObjects.Other, DynObjects.Class, DynObjects.UI, FileID, IgnoredOps, MsgrTrace, NSPlumbing, ProEvents.UI.Char, ProEvents.UI.Command, ProEvents.Other, QryInfo, SAX, Temp-tables, TTStats, UBroker.Basic, UBroker.ClientFSM, UBroker.ServerFSM, UBroker.ClientMsgStream, UBroker.ServerMsgStream, UBroker.ClientMsgQueue, UBroker.ServerMsgQueue, UBroker.ClientMemTrace, UBroker.ServerMemTrace, UBroker.ThreadPool, UBroker.Stats, UBroker.AutoTrim, UBroker.All, WSADefault, DS.Performance

In case this was not clear: we only need a subset of these types. We will implement those types which are documented above in #3853-1.

> Do we have that separation in the logic handled at conversion?

I don't think this is feasible. When the LOG-ENTRY-TYPES is assigned, it can be assigned from any character expression. Since we cannot rely upon this to be a string literal, we cannot know the value of LOG-ENTRY-TYPES until runtime. This means that only the runtime can know what needs to be logged.

More importantly, as far as I know, the logging can all be hidden inside the FWD runtime. Even things like 4GLTrace are not logging each line of code in the 4GL program, it is logging in response to various invocation mechanisms like the RUN statement. Each of these has some backing implementation in FWD and inside those backing runtime locations we will have to put logging code that matches the same result as the 4GL would generate. This means the timing of when the output occurs, outputting only that the exact same "event" that as the 4GL would log, ensuring that the format of the log entry is the same. I don't expect the converted code to change as a result of this task, since we already have the conversion of these 4GL elements implemented.

#25 - 12/23/2022 08:30 AM - Galya B

LOG-MANAGER:WRITE-MESSAGE supports by default text up to 15000 characters (actually it should be less, since that is for the whole statement in 4GL). If more than that it throws:

```
** More than 15000 characters in a single statement--use -inp parm. (135)
** .\test.p Could not understand line 16. (193)
** Unable to run startup procedure test.p. (492)
```

There is the param -inp described here as:

> The number of characters allowed in a single ABL statement. The default is 15000 characters.

I cannot find anything related to it in our code. I guess we don't have that 4GL limitation. Do we show the same error when LOG-MANAGER:WRITE-MESSAGE tries to write more than 15000 chars? If yes, then we'll have to also add support for startup command -inp.

#26 - 01/03/2023 08:33 AM - Greg Shah

> Do we show the same error when LOG-MANAGER:WRITE-MESSAGE tries to write more than 15000 chars? If yes, then we'll have to also add support for startup command -inp.

For now, I don't see a reason to implement this limitation. Since this is only present depending on command line arguments, it is not something that can be assumed to always exist. If the error doesn't exist, the only consequence is there is no error. I don't see any functional reason that application code would need to fail if log entries were over 15000 characters long. For now, ignore this.

#27 - 01/04/2023 03:10 AM - Galya B

Rotation and testing
A substantial part of LOG-MANAGER's logic is related to handling rotation of log files and is based on two startup params:

- -logthreshold activating rotation and setting the max file size in bytes,
- -numlogfiles defining the max number of log files to keep. The rest are removed regularly.

Both of them don't have setters and cannot be set programmatically which makes sense, otherwise changing rotation logic runtime would be a strange one. The problem is testing it automatically (in our setup). That's why there is nothing related to rotation in testcases. A very substantial portion of the whole logic is verifying startup params and delivering rotation based on that.

We need a separate script for each test to run it with the same predefined startup args in both Progress envs (where it can be started manually) and in our own pre-configured VM. For most cases VM is actually an overkill, a lightweight Docker container can run a lot of tests quickly in batch / appserver - agent mode. But the key here is to have pairs of script - test, where the script starts the test in a client with the expected startup args.

Log message chars limit
Actually on second thought keeping the 15000 chars per message limit makes sense in the context of file rotation, where the allowed min file size is 500_000 bytes. I guess there might have been some memory concerns taken in consideration when Progress made it a requirement for statements, but logging big chunks of text in small files might not be a good idea either. Even if we don't like their limit, we still need to impose a common sense one - message.length < logThreshold (and in that case we'll have to get into charset as well), so I would advice to keep the original one.

Multiple clients, one file
Now about the multi-threading / multi-processing part. Progress LOG-MANAGER has two modes:

- multiple sessions / processes writing to the same log file, when rotation is not enabled (-logthreshold not defined).
- only one session / process writing to the log files, when rotation is enabled. If process A starts with -clientlog client.log -logthreshold 500000 and process B starts after that with the same -clientlog client.log -logthreshold 500000. Process A is the only one able to write to files in the sequence like client.000001.log, while process B doesn't show any signs of error and just continues without logging. Check here . It's better documented somewhere else, but I cannot find it now. Anyways I've tested it and it's confirmed.

Our implementation of LOG-MANAGER runs purely on the server. When a client (gui, chui, batch) hits StandardServer it's on a new thread created specifically for that client. LegacyLogOps delivers a new instance of LegacyLogManagerImpl using the mechanism of ThreadLocal. Accessing LOG-MANAGER through LegacyLogOps makes it effectively singleton for the thread.

So in 4GL a session is a process, in FWD a session is a thread. By default having multiple sessions / threads writing to the same file without synchronization leads to broken files. The original no-rotation behavior in 4GL allows writing to the same file though. Also the other part is allowing only one session to write log files when rotation is enabled - we still need a common code to orchestrate that. Logging is a high throughput process and I'm definitely not thinking of creating a bottleneck, but it might be a good idea to still do something about it. I don't have the solution right now, I need to do some experiments to see what will work, especially with ThreadLocal as it is implemented. **But let me know what are your thoughts?**

Close log on client exit
Somewhat related to multi-threading is the solution I've implemented to closing the LOG-MANAGER, when the client exits. I've noticed that's when ContextLocal gets cleaned up, so I've added a cleanup '@Override for local (ContextLocal<WorkArea>) in LegacyLogOps that calls LOG-MANAGER:CLOSE-LOG.

#28 - 01/04/2023 04:58 AM - Galya B

Current clientlog implementation and -debugalert

Currently the code uses -clientlog startup param to enable logging of error messages that should be enabled by a different startup param -debugalert (i.e. SESSION:DEBUG-ALERT) and further expanded by -errorstack (i.e. SESSION:ERROR-STACK-TRACE).

-debugalert enables auto logging of:

> ABL stack trace for error messages (ABL errors and .NET Exceptions) and Alert-box messages

-errorstack:

> includes the contents of the CallStack property for the unhandled error

We have both of these behaviors now coming directly with -clientlog and independent of the parameters/configs.

Currently these logs are generated client-side (as far as I understand) in ThinClient and AlertBoxGuiImpl, but they seem to root from ErrorManager.

The proper LOG-MANAGER implementation obviously interferes with that use of 4GL -clientlog param. What we have can be covered by the proper implementation of -debugalert and -errorstack. Instead of trying to sneak in two different approaches I would recommend that we stay away from adding multiple responsibilities to something with a clear definition in 4GL and instead focus on completing -debugalert and -errorstack behavior depending on LegacyLogManagerImpl.

I will need a few days to come up with solution for it. **Do you want me to proceed with it and under what task?**

P.S. This also involves changes to logging level and entry types, which are set automatically with -debugalert presence.

P.S.S. The easiest quickest patch to complete #5753 would be to just switch the dependency from -clientlog to -debugalert (with a hardcoded path to the log file) and open a -debugalert separate task (maybe #4065 or this one #3853) to migrate the logic to LOG-MANAGER.

#29 - 01/04/2023 07:55 AM - Galya B

Galya Bogdanova wrote:

> P.S.S. The easiest quickest patch to complete #5753 would be to just switch the dependency from -clientlog to -debugalert (with a hardcoded

> path to the log file) and open a -debugalert separate task (maybe #4065 or this one #3853) to migrate the logic to LOG-MANAGER.

I've just added a new startup param -debugalertlog and switched the file name (-clientlog) passed to ThinClient#initClientLog by ClientCore. The change is quite simple and keeps both log files until proper -debugalert handling is implemented. Pushed to 5753a (commit 14468).

#30 - 01/04/2023 08:04 AM - Galya B

Galya Bogdanova wrote:

> open a -debugalert separate task (maybe #4065 or this one #3853) to migrate the logic to LOG-MANAGER.

I'm fine with continuing work under #5753 as that's where this got broken, but the changes will get beefy and I'm not sure how easy it will be for a code review if I continue adding to it.

#31 - 01/04/2023 12:29 PM - Greg Shah

- File can_multiple_clients_output_to_the_same_file_using_log_manager_psc_knowledgebase_article_p160687.pdf added

For logthreshold and numlogfiles and the other items from #3853-19, we should be able to configure these via 2 mechanisms:

1. directory.xml
   - We probably need some kind of override for setting the server-wide default values.
   - We definitely need account/group level control over the same.
   - I think this can be done with our normal multi-stage lookup process (see Utils.getDirectoryNode*() methods when DirScope.BOTH is passed).
2. bootstrap configuration (which can be in the bootstrap config file and/or on the command line)

Constantin: Any thoughts on this?

> We need a separate script for each test to run it with the same predefined startup args in both Progress envs (where it can be started manually) and in our own pre-configured VM. For most cases VM is actually an overkill, a lightweight Docker container can run a lot of tests quickly in batch / appserver - agent mode. But the key here is to have pairs of script - test, where the script starts the test in a client with the expected startup args.

In FWD we can handle this with bootstrap config values. The FWD server can stay the same and each test can launch a client session with whatever config overrides are needed.

> Actually on second thought keeping the 15000 chars per message limit makes sense in the context of file rotation

OK, go ahead with that. You'll have to add the -inp equvalent to the rest of the configuration values so that customers can override it when needed.

> Our implementation of LOG-MANAGER runs purely on the server.

It is certainly preferable to do this. However, we know there can be cases when this is simply wrong and the log files must be on the client side. We

need to support both.

> When a client (gui, chui, batch) hits StandardServer it's on a new thread created specifically for that client. LegacyLogOps delivers a new instance of LegacyLogManagerImpl using the mechanism of ThreadLocal. Accessing LOG-MANAGER through LegacyLogOps makes it effectively singleton for the thread.

We should not use ThreadLocal. It is not guaranteed that all log operations will be on the same thread. I think that was done as a quick and dirty approach, but we need this to be ContextLocal.

> in FWD a session is a thread. By default having multiple sessions / threads writing to the same file without synchronization leads to broken files. The original no-rotation behavior in 4GL allows writing to the same file though. Also the other part is allowing only one session to write log files when rotation is enabled - we still need a common code to orchestrate that. Logging is a high throughput process and I'm definitely not thinking of creating a bottleneck, but it might be a good idea to still do something about it.

We must solve this properly. The behavior needs to match the 4GL implementation while implementing full thread-safety. This will require that there be locking/mutex semantics OR that we post messages into a thread safe queue and have a single worker thread that handles writing to log files. I think this worker thread idea could be simple enough that we have 1 thread which writes to any number of log files. This would work if the message posted into the queue was an object that stored the log entry as well as the log file to be written to.

If that single thread approach becomes a bottleneck, we can implement a thread pool that can do the same job.

> Somewhat related to multi-threading is the solution I've implemented to closing the LOG-MANAGER, when the client exits. I've noticed that's when ContextLocal gets cleaned up, so I've added a cleanup '@Override for local (ContextLocal<WorkArea>) in LegacyLogOps that calls LOG-MANAGER:CLOSE-LOG.

This is the right idea. We'll have to review the implementation to confirm it follows all needed conventions.

> Instead of trying to sneak in two different approaches I would recommend that we stay away from adding multiple responsibilities to something with a clear definition in 4GL and instead focus on completing -debugalert and -errorstack behavior depending on LegacyLogManagerImpl.

Agreed. Let's get this right. We want the same behavior as is implemented by the 4GL in these cases.

> I will need a few days to come up with solution for it. Do you want me to proceed with it and under what task?

Yes, go ahead with this in this same task. This seems to be part of the LOG-MANAGER implementation.

> > open a -debugalert separate task (maybe #4065 or this one #3853) to migrate the logic to LOG-MANAGER.

> I'm fine with continuing work under #5753 as that's where this got broken, but the changes will get beefy and I'm not sure how easy it will be for a code review if I continue adding to it.

Do this in a 2nd phase. We can move ahead with the core implementation and get all of that reviewed and then merged into 3821c (or whatever). Then in a 2nd phase, we open a new task branch just for this part of the cleanup.

#32 - 01/04/2023 12:33 PM - Greg Shah

From Galya via email:

> I've tested gui, batch, web gui clients and they are working well with LOG-MANAGER. But I've noticed appserver (scheduled batch) doesn't use LOG-MANAGER. It's not related to the startup command, because log manager should work with programmatically set configs (which it doesn't now). It might be that log manager initialization in StandardServer#standardEntry somehow doesn't affect the instance appserver uses. I don't completely understand what's going on after AppServerManager.startAppServer().

Does the 4GL have an implicit set of settings for LOG-MANAGER in an appserver process? If so, then we need to implement the equivalent.

Constantin will have more thoughts on this.

#33 - 01/04/2023 12:46 PM - Constantin Asofiei

Greg Shah wrote:

> From Galya via email:
>
> > I've tested gui, batch, web gui clients and they are working well with LOG-MANAGER. But I've noticed appserver (scheduled batch) doesn't use LOG-MANAGER. It's not related to the startup command, because log manager should work with programmatically set configs (which it doesn't now). It might be that log manager initialization in StandardServer#standardEntry somehow doesn't affect the instance appserver uses. I don't completely understand what's going on after AppServerManager.startAppServer().

For appservers, any initialization code needs to be written in Agent.prepare. This is a duplicate of any initialization code which you add to StandardServer.standardEntry, after the AppServerManager.startAppServer() test, for non-appserver clients.

#34 - 01/05/2023 03:06 AM - Constantin Asofiei

Greg Shah wrote:

> For logthreshold and numlogfiles and the other items from #3853-19, we should be able to configure these via 2 mechanisms:
>
> 1. directory.xml
>   - We probably need some kind of override for setting the server-wide default values.
>   - We definitely need account/group level control over the same.
>   - I think this can be done with our normal multi-stage lookup process (see Utils.getDirectoryNode*() methods when DirScope.BOTH is passed).
> 2. bootstrap configuration (which can be in the bootstrap config file and/or on the command line)
>
> Constantin: Any thoughts on this?

Yes, I think that's about right. We should consider having similar/identical names for these settings, for both directory and bootstrap cases. Galya: please post what logmanager related config needs to added.

#35 - 01/05/2023 03:50 AM - Galya B

Constantin Asofiei wrote:

> Galya: please post what logmanager related config needs to added.

4GL LOG-MANAGER configs:

Startup param	Programmatic getter	Programmatic setter	Description	Standalone clients - client bootstrap	Web and scheduled clients - directory.xml
-clientlog	LOG-MANAGER :LOGFILE-NAM E	LOG-MANAGER :LOGFILE-NAM E	Absolute or relative path to log file. Enables logging.	client:cmd-line-o ption:clientlog	clientConfig/cfgO verrides
-logginglevel	LOG-MANAGER :LOGGING-LEV EL	LOG-MANAGER :LOGGING-LEV EL	0 off; 1 errors; 2 (default sometimes) basic; 3 verbose; 4 extended	client:cmd-line-o ption:loggingleve l	clientConfig/cfgO verrides
-logthreshold	LOG-MANAGER :LOG-THRESHO LD	none	Enables file rotation. Defines max file size: 0 (default) for no limit; 500,000 - 2,147,483,647 for bytes (488 KB - 2 GB)	client:cmd-line-o ption:logthreshol d	clientConfig/cfgO verrides
-numlogfiles	LOG-MANAGER :NUM-LOG-FILE S	none	Max number of log files between 000001 - 999999. Files get deleted, when exceeded.	client:cmd-line-o ption:numlogfiles	clientConfig/cfgO verrides
-logentrytypes	LOG-MANAGER :LOG-ENTRY-T YPES	LOG-MANAGER :LOG-ENTRY-T YPES	Comma separated list of entry types (optional :logginglevel), example: QryInfo:2,4GLMe ssages	client:cmd-line-o ption:logentrytyp es	clientConfig/cfgO verrides
-debugalert	SESSION:DEBU G-ALERT	SESSION:DEBU G-ALERT	ABL stack trace for error messages (ABL errors and .NET Exceptions) and Alert-box messages. Affects LOG-MANAGER log level and entry types.	client:cmd-line-o ption:debugalert	clientConfig/cfgO verrides
-errorstack	SESSION:ERRO R-STACK-TRAC E	SESSION:ERRO R-STACK-TRAC E	Allows error objects to save the ABL call stack in the CallStack property of an error object. Not recommended in production. Affects the content of log	client:cmd-line-o ption:errorstack	clientConfig/cfgO verrides

			records enabled by -debugalert.		
-inp	none	none	Max number of input character for a single statement. Default 15000. Max 2147483647.	client:cmd-line-option:inp	clientConfig/cfgOverrides

I should have made the necessary changes on branch 5753a.

Having no setters for rotation related configs, means tests should rely on configurations outside of them, which is not easy to automate without a dedicated process.

#36 - 01/05/2023 04:47 AM - Galya B

Greg Shah wrote:

> Does the 4GL have an implicit set of settings for LOG-MANAGER in an appserver process? If so, then we need to implement the equivalent.
>
> Constantin will have more thoughts on this.

Update: The issue is not in the configs or LOG-MANAGER initialization. If all relevant startup params are removed LOG-MANAGER should still get enabled by this procedure and should produce logs:

```
LOG-MANAGER:LOGFILE-NAME = "/home/gbb/testcases/deploy/bgr.log".
LOG-MANAGER:WRITE-MESSAGE("Hello ABL").
```

It does for all other types of clients, only appserver_process doesn't produce anything. So I'm thinking there is something wrong with how the LOG-MANAGER handle is used in this mode, but I have no clue what to look for.

P.S. I compared the client logs of the scheduled batch (appserver_process) and web and it seems web starts ContextHandler, while appserver_process doesn't. The LOG-MANAGER handle accesses the instance of LegacyLogManagerImpl through LegacyLogOp's ContextLocal<WorkArea>, so it seems to be related.

#37 - 01/05/2023 06:47 AM - Galya B

A few observations in relationship to max log message length (tested):

- By default statement length is limited to 15000 characters.
- With -inp (or Input Characters) statement length can be set to max 2147483647 characters (~2GB).
- But LOG-MANAGER:WRITE-MESSAGE works for up to 32767 characters text (32K bytes per record). After that messages are completely ignored. That limit is documented in Inputoutput-limits . A side notice: Java FileWriter (I use in LegacyLogManagerImpl implementation) uses UTF-16 by default and that will be 64K bytes.
- Logging alerts from MESSAGE VIEW-AS ALERT-BOX is limited to 3000 characters. Bigger messages are cut to the 3000 character.
- With messages of type ALERT-BOX having length more than 65000 characters, the app crashes consistently or sometimes gives stget: out of storage (1450) error.

The conclusion is: Although Progress have the -inp startup param, the apps have other limitations and perform poorly with very long full texts.
I'm about to implement -inp. Do we want also the 32767 chars limit for LOG-MANAGER:WRITE-MESSAGE? If not, I'll still have to implement the common sense validation of checking for rotation file size limit (logRecord.getBytes().length > logThreshold).

#38 - 01/05/2023 07:48 AM - Greg Shah

> Do we want also the 32767 chars limit for LOG-MANAGER:WRITE-MESSAGE?

Yes. Also, please implement the limit of 3000 in MESSAGE VIEW-AS ALERT-BOX.

> With messages of type ALERT-BOX having length more than 65000 characters, the app crashes consistently

Is this different than MESSAGE VIEW-AS ALERT-BOX? Or is this just the case where the 4GL handles the really long messages badly and doesn't get the chance to truncate to 3000?

#39 - 01/05/2023 07:49 AM - Galya B

Greg Shah wrote:

> > Our implementation of LOG-MANAGER runs purely on the server.
>
> It is certainly preferable to do this. However, we know there can be cases when this is simply wrong and the log files must be on the client side.

> We need to support both.

I will figure out ways to implement it both server-side and client-side, when I see the case. I still haven't got into implementing log entry types (#3853), so that might be where it needs to be extended. We'll discuss it further, when I get familiar with these requirements.

> > LegacyLogOps delivers a new instance of LegacyLogManagerImpl using the mechanism of ThreadLocal.

> We should not use ThreadLocal. It is not guaranteed that all log operations will be on the same thread. I think that was done as a quick and dirty approach, but we need this to be ContextLocal.

It is ContextLocal, which creates an instance of the log manager wrapper through class Fallback extends ThreadLocal<T> get() for standalone clients and some cases in server clients. It's guaranteed that different clients will operate on different threads with different instances of the log manager. But maybe you mean one client might operate on multiple threads? 4GL apps are single threaded. Do you mean that FWD implementation is different?

> This would work if the message posted into the queue was an object that stored the log entry as well as the log file to be written to.

This sounds good at first, but there is a ton of validation and error messages going on in LegacyLogManagerImpl and the queue will have to be quite smart (taking over log manager's responsibilities) or sending back a lot of feedback.

#40 - 01/05/2023 07:51 AM - Galya B

Greg Shah wrote:

> Is this different than MESSAGE VIEW-AS ALERT-BOX? Or is this just the case where the 4GL handles the really long messages badly and doesn't get the chance to truncate to 3000?

It's still MESSAGE VIEW-AS ALERT-BOX. When the message is 65000+ chars the whole app crashes. When it's less, it's processed and truncated to 3000 in the log file.

#41 - 01/05/2023 07:57 AM - Galya B

Greg Shah wrote:

> We must solve this properly. The behavior needs to match the 4GL implementation while implementing full thread-safety.

I'll get back to you with some options.

#42 - 01/05/2023 08:10 AM - Greg Shah

> It is ContextLocal, which creates an instance of the log manager wrapper through class Fallback extends ThreadLocal<T> get() for standalone clients and some cases in server clients.

We don't really use the standalone mode. It will probably be removed at some point. As long as ContextLocal is used, it will be correct on the server-side.

> But maybe you mean one client might operate on multiple threads? 4GL apps are single threaded. Do you mean that FWD implementation is different?

Yes, FWD is different. We simulate single-threaded mode in a multi-threaded environment. There are 2 modes for a non-virtual client session: conversation mode and dispatcher mode. In conversation mode, all normal processing happens on the "conversation" thread, which is a kind of server-side dedicated "main" thread for the session. However, even in conversation mode there are still ways that code can be invoked on non-conversation threads in a session. This mostly happens in response to an asynchronous interruption like CTRL-C in a ChUI session which generates a STOP condition. In non-conversation mode, we process all logic on dispatcher threads so it can be a different thread each time. You can have a mixture of conversation mode and dispatcher mode sessions in the same server, it is an attribute that is defined in the bootstrap config at the moment the session is started.

Understanding FWD Threading has some useful information related to this.

> > This would work if the message posted into the queue was an object that stored the log entry as well as the log file to be written to.
>
> This sounds good at first, but there is a ton of validation and error messages going on in LegacyLogManagerImpl and the queue will have to be quite smart (taking over log manager's responsibilities) or sending back a lot of feedback.

I would think that the LegacyLogManagerImpl would do all the work until it decides that it actually needs to write a message out, then it can post to the queue. But you will know better as you analyze things.

#43 - 01/05/2023 08:10 AM - Greg Shah

> Is this different than MESSAGE VIEW-AS ALERT-BOX? Or is this just the case where the 4GL handles the really long messages badly and doesn't get the chance to truncate to 3000?

It's still MESSAGE VIEW-AS ALERT-BOX. When the message is 65000+ chars the whole app crashes. When it's less, it's processed and truncated to 3000 in the log file.

OK, then this one is "easy" since we don't need to replicate their crashes. :)

#44 - 01/10/2023 09:20 AM - Galya B

Logging level rules:

Startup params/4GL setters	Auto set logging level
none	2
-logentrytypes (invalid)	0
-clientlog	1
-clientlog + LOG-MANAGER:LOGFILE-NAME	1
LOG-MANAGER:LOGFILE-NAME	2
-clientlog + -logthreshold + -numlogfiles	1
-clientlog + -logentrytypes	2

4GL setter cannot be used to disable logging by LOG-MANAGER:LOGGING-LEVEL = 0.. This code doesn't change the logging level.

#45 - 01/12/2023 06:40 AM - Galya B

I'm trying to find the difference in the log files when SESSION:ERROR-STACK-TRACE (-errorstack) is enabled, but for now without success. It is supposed to include:

> the contents of the CallStack property for the unhandled error.

A certain procedure test.p throws an error in a nested function throwerrornested called by another function throwerror. This is the log record:

pro -b -p test.p -clientlog "test.log"

```
[23/01/12@12:24:22.580+0100] P-002940 T-013736 1 4GL -- (Procedure: 'throwerrornested test.p' Line:14) Connect
ion failure for host localhost port 5162 transport UDP. (9407)
```

pro -b -p test.p -clientlog "test.log" -debugalert

```
[23/01/12@12:23:45.153+0100] P-007656 T-004072 1 4GL -- Connection failure for host localhost port 5162 transp
ort UDP. (9407)
[23/01/12@12:23:45.153+0100] P-007656 T-004072 1 4GL -- ** ABL Debug-Alert Stack Trace **
[23/01/12@12:23:45.153+0100] P-007656 T-004072 1 4GL -- --> throwerrornested test.p at line 14  (.\test.p)
[23/01/12@12:23:45.153+0100] P-007656 T-004072 1 4GL --     throwerror test.p at line 19  (.\test.p)
[23/01/12@12:23:45.153+0100] P-007656 T-004072 1 4GL --     test.p at line 44  (.\test.p)
```

The problem is pro -b -p test.p -clientlog "test.log" -debugalert -errorstack returns the same as pro -b -p test.p -clientlog "test.log" -debugalert. How is this CallStack property different from ABL Debug-Alert Stack Trace? Any idea what I can try to produce it?

#46 - 01/12/2023 10:05 AM - Greg Shah

Marian: Can you offer any guidance?

#47 - 01/13/2023 02:43 AM - Marian Edu

Greg Shah wrote:

> Marian: Can you offer any guidance?

- -errorstack (session:error-stack-trace) has no effect on the client log (log manager), if enabled it will just attach the stack trace (full) to the error object CallStack when catch is used (no effect on standard error handling, no-error/error-status).
- -debugalert (session:debug-alert) does affect the client log and it is mentioned both on the debugalert and clientlog the fact that when that is set to true the stack trace (top entry only) will be written to the log file, not the full trace and that is not saved in the generated error object CallStack.

Bref, using -errostack has no effect on the client log so don't bother with it in this case.

#48 - 01/13/2023 02:52 AM - Galya B

Marian Edu wrote:

> Bref, using -errostack has no effect on the client log so don't bother with it in this case.

Thank you for the confirmation. This is what I observe as well. It's just that the documentation mixes both UI and clientlog and sounds misleading. Good that we cleared this up.

#49 - 01/13/2023 08:42 AM - Galya B

When -clientlog or LOG-MANAGER:LOGFILE-NAME is defined logging level 1 is always enabled (except if explicitly set to 0 on startup with -logginglevel 0 or value of startup param -logentrytypes was invalid). Logging level 1 creates log records for all types of MESSAGE "" VIEW-AS ALERT-BOX and unhandled errors.

Example procedure lvl1.p:

```
DEF VAR h AS HANDLE NO-UNDO.

FUNCTION ThrowErrorNested RETURNS INTEGER ():
  CREATE SERVER h.
  h:CONNECT("").
  RETURN 1.
END.

FUNCTION ThrowError RETURNS INTEGER ():
  RETURN ThrowErrorNested().
END.

MESSAGE "MESSAGE" VIEW-AS ALERT-BOX MESSAGE.
MESSAGE "QUESTION" VIEW-AS ALERT-BOX QUESTION.
MESSAGE "INFORMATION" VIEW-AS ALERT-BOX INFORMATION.
MESSAGE "ERROR" VIEW-AS ALERT-BOX ERROR.
MESSAGE "WARNING" VIEW-AS ALERT-BOX WARNING.
MESSAGE "GENERIC" VIEW-AS ALERT-BOX.

ThrowError().
```

Logs produced when run with pro -b -p lvl1.p -clientlog lvl1.log:

```
[23/01/13@14:29:59.301+0100] P-008112 T-008220 1 4GL -- Logging level set to = 1
[23/01/13@14:29:59.358+0100] P-008112 T-008220 1 4GL -- MESSAGE
[23/01/13@14:29:59.358+0100] P-008112 T-008220 1 4GL -- QUESTION
[23/01/13@14:29:59.358+0100] P-008112 T-008220 1 4GL -- INFORMATION
[23/01/13@14:29:59.358+0100] P-008112 T-008220 1 4GL -- ERROR
[23/01/13@14:29:59.358+0100] P-008112 T-008220 1 4GL -- WARNING
[23/01/13@14:29:59.358+0100] P-008112 T-008220 1 4GL -- GENERIC
[23/01/13@14:29:59.387+0100] P-008112 T-008220 1 4GL -- (Procedure: 'ThrowErrorNested lvl1.p' Line:5) Connecti
on failure for host localhost port 5162 transport UDP. (9407)
[23/01/13@14:29:59.393+0100] P-008112 T-008220 1 4GL -- (Procedure: 'ThrowErrorNested lvl1.p' Line:5) Applicat
ion server connect failure. (5468)
```

Logs produced when run with pro -b -p lvl1.p -clientlog lvl1.log -debugalert:

```
[23/01/13@14:30:41.274+0100] P-013624 T-012048 1 4GL -- Logging level set to = 1
[23/01/13@14:30:41.341+0100] P-013624 T-012048 1 4GL -- MESSAGE
[23/01/13@14:30:41.341+0100] P-013624 T-012048 1 4GL -- ** ABL Debug-Alert Stack Trace **
[23/01/13@14:30:41.341+0100] P-013624 T-012048 1 4GL -- --> lvl1.p at line 13  (.\lvl1.p)
[23/01/13@14:30:41.341+0100] P-013624 T-012048 1 4GL -- QUESTION
[23/01/13@14:30:41.341+0100] P-013624 T-012048 1 4GL -- ** ABL Debug-Alert Stack Trace **
[23/01/13@14:30:41.341+0100] P-013624 T-012048 1 4GL -- --> lvl1.p at line 14  (.\lvl1.p)
[23/01/13@14:30:41.341+0100] P-013624 T-012048 1 4GL -- INFORMATION
[23/01/13@14:30:41.341+0100] P-013624 T-012048 1 4GL -- ** ABL Debug-Alert Stack Trace **
[23/01/13@14:30:41.341+0100] P-013624 T-012048 1 4GL -- --> lvl1.p at line 15  (.\lvl1.p)
[23/01/13@14:30:41.341+0100] P-013624 T-012048 1 4GL -- ERROR
[23/01/13@14:30:41.341+0100] P-013624 T-012048 1 4GL -- ** ABL Debug-Alert Stack Trace **
[23/01/13@14:30:41.341+0100] P-013624 T-012048 1 4GL -- --> lvl1.p at line 16  (.\lvl1.p)
[23/01/13@14:30:41.341+0100] P-013624 T-012048 1 4GL -- WARNING
[23/01/13@14:30:41.342+0100] P-013624 T-012048 1 4GL -- ** ABL Debug-Alert Stack Trace **
[23/01/13@14:30:41.342+0100] P-013624 T-012048 1 4GL -- --> lvl1.p at line 17  (.\lvl1.p)
[23/01/13@14:30:41.342+0100] P-013624 T-012048 1 4GL -- GENERIC
[23/01/13@14:30:41.342+0100] P-013624 T-012048 1 4GL -- ** ABL Debug-Alert Stack Trace **
[23/01/13@14:30:41.342+0100] P-013624 T-012048 1 4GL -- --> lvl1.p at line 18  (.\lvl1.p)
[23/01/13@14:30:41.361+0100] P-013624 T-012048 1 4GL -- Connection failure for host localhost port 5162 transp
ort UDP. (9407)
[23/01/13@14:30:41.361+0100] P-013624 T-012048 1 4GL -- ** ABL Debug-Alert Stack Trace **
[23/01/13@14:30:41.361+0100] P-013624 T-012048 1 4GL -- --> ThrowErrorNested lvl1.p at line 5  (.\lvl1.p)
[23/01/13@14:30:41.361+0100] P-013624 T-012048 1 4GL --     ThrowError lvl1.p at line 10  (.\lvl1.p)
[23/01/13@14:30:41.361+0100] P-013624 T-012048 1 4GL --     lvl1.p at line 20  (.\lvl1.p)
[23/01/13@14:30:41.368+0100] P-013624 T-012048 1 4GL -- Application server connect failure. (5468)
[23/01/13@14:30:41.368+0100] P-013624 T-012048 1 4GL -- ** ABL Debug-Alert Stack Trace **
[23/01/13@14:30:41.368+0100] P-013624 T-012048 1 4GL -- --> ThrowErrorNested lvl1.p at line 5  (.\lvl1.p)
[23/01/13@14:30:41.368+0100] P-013624 T-012048 1 4GL --     ThrowError lvl1.p at line 10  (.\lvl1.p)
[23/01/13@14:30:41.368+0100] P-013624 T-012048 1 4GL --     lvl1.p at line 20  (.\lvl1.p)
```

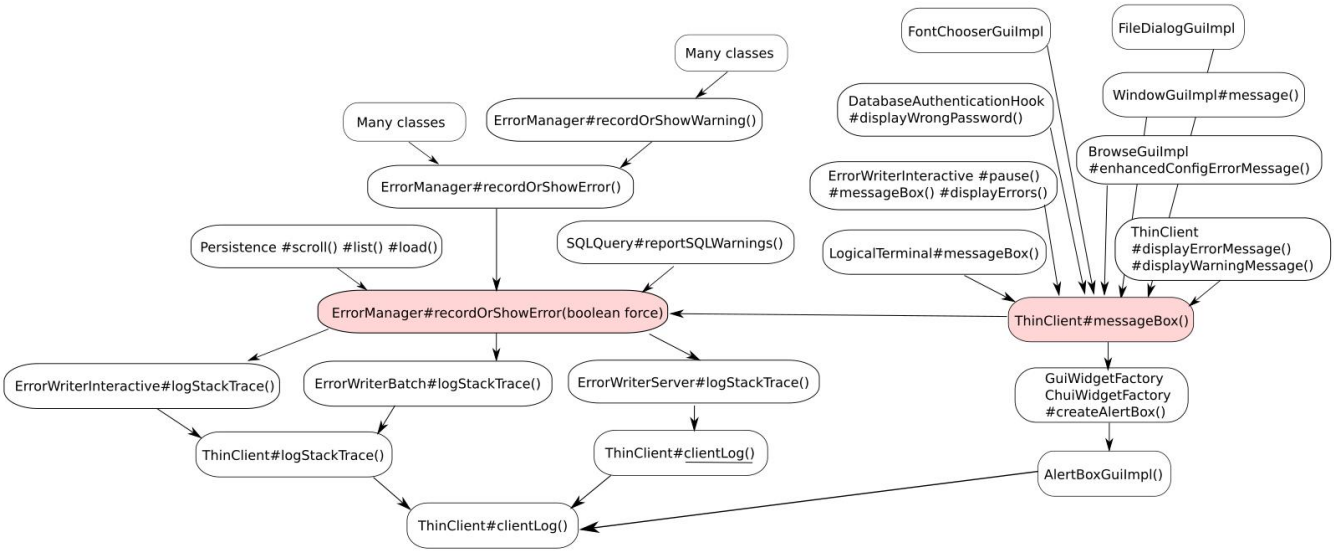
Currently ThinClient#clientLog() doesn't log these. There seem to be several conditions up in the call stack that seem to interfere with this function being called. So the LOG-MANAGER implementation for logging level one will not rely much on what's already in the code.

#50 - 01/17/2023 09:13 AM - Galya B

- File clientlog-old.png added

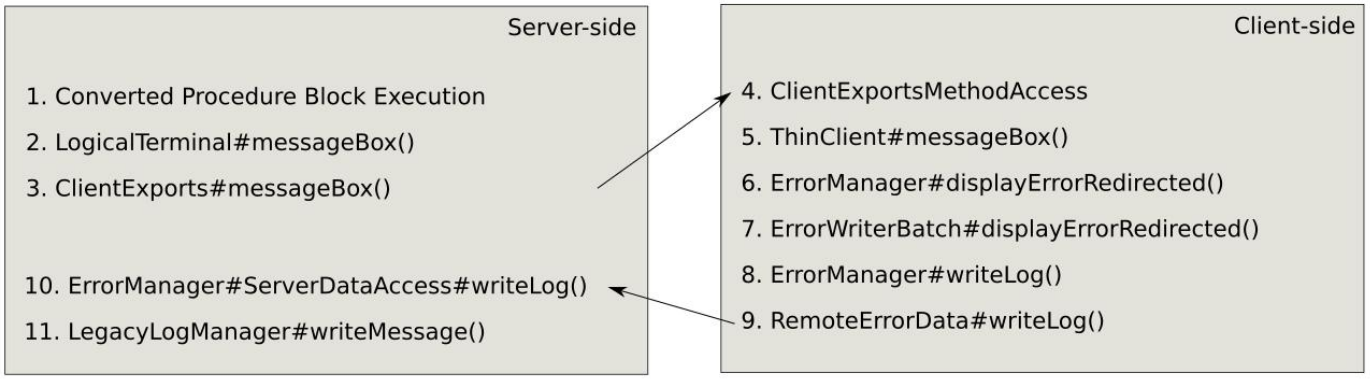
- File clientlog-new.png added

This diagram shows the current state of the call stack producing error logs (not a complete view):

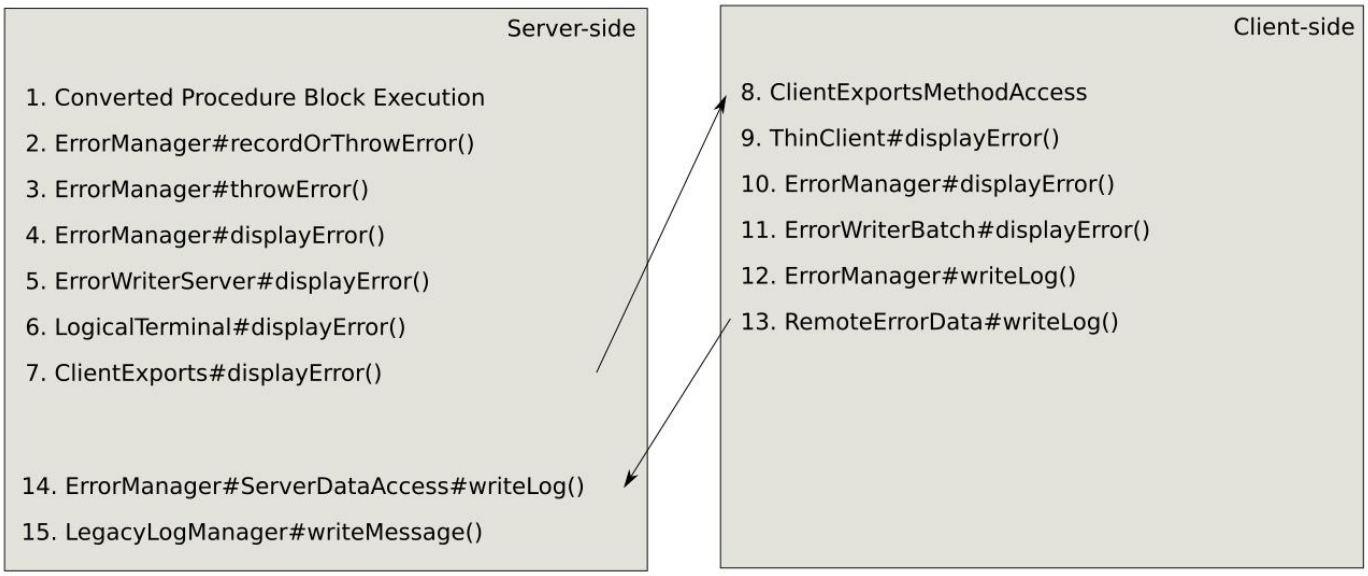


This is the call stack for two specific scenarios of producing an error log with my latest changes to branch 5753b:

VIEW-AS ALERT-BOX Call Stack In Batch



Unhandled 4GL Numbered Error Call Stack In Batch



#51 - 01/25/2023 01:06 PM - Galya B

I've found an unexpected behavior with batch execution. stdout is displayed using ErrorManager.displayError(lineOut); in com.goldencode.p2j.ui.client.chui.driver.batch.BatchPrimitives#updateScreen() L276. That causes our implementation of LOG-MANAGER to write all stdout (display, etc.) as errors in the log files in batch. Is ErrorWriterBatch used as the default screen writer for batch?
This is what ErrorWriterBatch#displayError() does (as of active branch 5753b):

```
113      if (client != null && client.isRedirected())
114      {
115         client.displayErrorMessage(errmsg, false, null, logCallStack);
116      }
117      else
118      {
119         ErrorManager.writeLog(errmsg, logCallStack);
120      }
121
122      if (PlatformHelper.isUnderWindowsFamily() && !isBatchInBackground() && !FileChecker.isConsoleRedirect
ed())
123      {
124         // TODO: In Windows the native message box should be displayed here.
125      }
126
127      // then log it to redirected console
128      System.out.println(errmsg);
```

It executes lines 119 and 128, which is expected with errors, but not with regular messages in batch.

I would like to replace the call in BatchPrimitives#updateScreen() with System.out.println(lineOut);. Who can confirm if it's safe?

#52 - 01/27/2023 08:04 AM - Galya B

- File fix-batch-output-as-errors.diff added

I've found a workaround fix for batch writing all stdout through ErrorWriterBatch#displayError(), which messed up LOG-MANAGER error logging. Here is the diff attached. It's also committed as r14485 in 5753b.

#53 - 02/27/2023 08:39 AM - Greg Shah

- Related to Feature #7147: Make FWD log outputs consistent added

#54 - 03/22/2023 11:01 AM - Galya B

Note: To check if entry types are case-sensitive in configs.

#55 - 03/30/2023 04:25 AM - Galya B

The tests Constantin did for #5701#note-86 made me think about the appserver agent logs.

The implementation in #5753 is as follows: By default no log files are generated. LOG-MANAGER should be enabled by configuring a log file name for the appserver_process:

```
<node class="container" name="">
  <node class="container" name="server">
    <node class="container" name="default">
        <node class="container" name="appserver_process">
          <node class="container" name="clientConfig">
            <node class="string" name="cfgOverrides">
              <node-attribute name="value" value="client:cmd-line-option:clientlog=appserver_process.log"/>
            </node>
          </node>
        </node>
    </node>
  </node>
</node>
```

All other configs applicable to LOG-MANAGER are available for as described in Logs wiki. All agents of the appserver will write to the same log files as it is in OE.

A complete implementation should:

- review if current cfgOverrides are compliant to AppServer Agent Logging Setting properties ; In OE the configs are entered in the graphical env for setting up appservers;
- subsystem code AS as in Logs;
- take into consideration that LOG-MANAGER handles are dealt with in a specific way, when used in procedures running on appservers, because the agent itself already has a log file. Check the log in Logs (**LOGFILE-NAME is not a setable attribute for PSEUDO-WIDGET.). This behavior should be cross-checked with OE and modifications applied;
- consider removing ouputToFile for appservers, because after #7236 is fixed, it should contain the same output as the appserver log file.

#56 - 04/03/2023 03:47 AM - Galya B

Answering to #5753#note-113 :

Constantin Asofiei wrote:

From this, we know:

- changing LOGFILE-NAME does not raise an ERROR condition on the appserver
- LOGGING-LEVEL can be set
- State-reset does not change the LOG-MANAGER to its initial value - I think what needs to be done in FWD is to:
  - in Agent.prepare, if Agent.logManager (a new field) is null, initialize the LogManager's context-local state and save the this instance at Agent.logManager
  - in Agent.prepare, if Agent.logManager (a new field) is not null, initialize the LogManager's context-local state with the saved one

Does this make sense? We will need to check the other LOG-MANAGER attributes (beside LOGGING-LEVEL and LOGFILE-NAME) to see if they can be changed on the appserver.

- On appserver LOGFILE-NAME doesn't change anything, but shows a warning message, it's not an error indeed.
- LOGGING-LEVEL can be set, but does it take actual effect is something to consider when entry types with higher level log levels are tested and implemented.
- State-reset - What I implemented in 5753d r14519 is a different approach with the same result - on every Agent.prepare LegacyLogManagerImpl.initiate is called and if it's a new instance, then the flag isInstantiated is false and the work gets done.
- The other attributes are:
  - logentrytypes - not fully implemented yet.
  - numlogfiles, logthreshold - don't have setters, settable only with startup configs.

#57 - 04/03/2023 05:34 AM - Galya B

Appservers have the following logging configs in OE Explorer:

- srvrLogFile - by default {WorkPath}\%appservername%.server.log.
- srvrLogAppend - by default 1.
- srvrLoggingLevel - by default Basic or 2.
- srvrLogEntryTypes - by default ASPlumbing,DB.Connects activated.
- srvrNumLogFiles - by default 3.
- srvrLogWatchdogInterval ? - by default 60
- srvrLogThreshold - by default 0
- srvrLogEntries - ? by default 0

#58 - 04/03/2023 06:11 AM - Galya B

LOG-ENTRY-TYPES and LOGGING-LEVEL are LOG-MANAGER attributes that can be changed in the internal procedure running on the appserver agent, but they don't change the configurations in the OE Explorer. It's still to be determined if this is a long lived change and if it affects all agents running at that time on the appserver as well as those running after the disconnect.

#59 - 04/03/2023 10:35 AM - Galya B

- % Done changed from 30 to 60

#60 - 04/21/2023 01:59 AM - Galya B

- Related to Bug #7291: LOG-MANAGER to log on client-side added

#61 - 05/31/2023 11:57 AM - Galya B

Here are my startup args tests. Now with focus on error thrown or shown. The current impl will have to be revised.

msg.p contains only MESSAGE "TEST".. If TEST shows on the screen, it means the client doesn't thrown an error and the execution continues.

```
pro -p msg.p -b
pro -p msg.p -clientlog msg.log -b
TEST

pro -p msg.p -clientlog /invalid/path -b
Cannot open log file /invalid/path, errno 3 (11076)
TEST

pro -p setname.p -b
(set programmatically) LOG-MANAGER:LOGFILE-NAME="/invalid/path".
Cannot open log file /invalid/path, errno 3 (11076)
** Unable to open file for PSEUDO-WIDGET. (4487)
TEST

IN APPSERVER
pro -p setnameappserver.p -b
(set programmatically) LOG-MANAGER:LOGFILE-NAME="name.log".
LOGFILE-NAME is not a setable attribute for PSEUDO-WIDGET. (4052)
TEST

pro -p msg.p -clientlog msg.log -logentrytypes Invalid -b
pro -p msg.p -logentrytypes Invalid -clientlog msg.log -b
Ignoring unknown log entry type: Invalid (11069)
TEST

pro -p msg.p -logentrytypes 4GLTrace:1 -clientlog msg.log -b
Logging level for 4GLTrace must be 2 or higher (11071)
TEST

pro -p msg.p -logentrytypes 4GLTrace -clientlog msg.log -logginglevel 1 -b
Logging level greater than 1 must be specified either for 4GLTrace or for all types by using -logginglevel. (1
1072)
TEST

pro -p msg.p -logentrytypes 4GLTrace -clientlog msg.log -b
--
(set programmatically)
LOG-MANAGER:LOGGING-LEVEL=1.
MESSAGE LOG-MANAGER:LOGGING-LEVEL.
MESSAGE "TEST".
--
2
TEST

pro -p msg.p -clientlog msg.log -logginglevel abc -b
The -logginglevel parameter requires a numeric argument. (11996)

pro -p msg.p -clientlog msg.log -logginglevel -1 -b
pro -p msg.p -clientlog msg.log -logginglevel 2147483648 -b
The -logginglevel parameter has too many digits. (5049)

pro -p setloglevel.p -b -clientlog msg.log -b
(set programmatically) LOG-MANAGER:LOGGING-LEVEL=-1.
**Attribute LOGGING-LEVEL for the PSEUDO-WIDGET has an invalid value of -1. (4057)
TEST

pro -p setloglevel.p -b -clientlog msg.log -b
(set programmatically) LOG-MANAGER:LOGGING-LEVEL=2147483648.
**Attribute LOGGING-LEVEL for the PSEUDO-WIDGET has an invalid value of -2147483648. (4057)
TEST

pro -p msg.p -logginglevel 1 -b
```

```
Parameters for logging were specified but -clientlog was not set. Logging parameters will be ignored. Specify
-logginglevel 0 if you want to keep logging disabled at startup. (11068)
TEST

pro -p setloglevel.p -b
(set programmatically) LOG-MANAGER:LOGGING-LEVEL=5.
**Cannot set attribute LOGGING-LEVEL because log file name was not specified at startup. (11078)
TEST

pro -p setloglevel.p -b
(set programmatically) LOG-MANAGER:LOGGING-LEVEL=5.

pro -p msg.p -clientlog msg.log -logthreshold -1 -b
pro -p msg.p -clientlog msg.log -logthreshold 2147483648 -b
The -logthreshold parameter has too many digits. (5049)

pro -p msg.p -clientlog msg.log -logthreshold abc -b
The -logthreshold parameter requires a numeric argument. (11996)

pro -p msg.p -clientlog msg.log -logthreshold 499999 -b
-logthreshold must be set to a value between 500000 and 2147483647 (11065)

pro -p msg.p -clientlog msg.log -numlogfiles -1 -b
pro -p msg.p -clientlog msg.log -numlogfiles 2147483648 -b
The -numlogfiles parameter has too many digits. (5049)

pro -p msg.p -clientlog msg.log -numlogfiles 1 -b
-numlogfiles cannot be set to 1 (11067)

pro -p msg.p -clientlog msg.log -numlogfiles 1000000 -b
-numlogfiles cannot be set to a value greater than 999999 (14416)

pro -p clearlog.p -b
(programmatically) LOG-MANAGER:CLEAR-LOG().
Cannot clear log because there is no log file open (14333)
TEST

pro -p writemsg.p -b
(programmatically) LOG-MANAGER:WRITE-MESSAGE("MSG").
Cannot write message to log, as there is no log open (14332)
TEST
```

#62 - 06/22/2023 10:36 AM - Constantin Asofiei

- Related to Bug #7462: fix 'AS' instead of '4GL' in LOG-MANAGER log lines and add an incompatible FWD and FWD-AS for 4GL and AS texts added

#63 - 06/26/2023 11:03 AM - Galya B

Note: This task should include proper implementation of basic-logging that is already supported in the 4GL syntax.

#64 - 06/26/2023 11:25 AM - Greg Shah

- % Done changed from 60 to 0

Galya B wrote:

> Note: This task should include proper implementation of basic-logging that is already supported in the 4GL syntax.

Do you mean logging-level 2 (basic) or query:basic-logging?

#65 - 06/26/2023 12:02 PM - Galya B

Greg Shah wrote:

> Galya B wrote:
>
> > Note: This task should include proper implementation of basic-logging that is already supported in the 4GL syntax.
>
> Do you mean logging-level 2 (basic) or query:basic-logging?

https://docs.progress.com/bundle/abl-reference/page/BASIC-LOGGING-attribute.html

#66 - 06/26/2023 01:25 PM - Greg Shah

QUERY:BASIC-LOGGING is already listed in #3853-1. Whether we implement a new set of logging instrumentation for it or we repurpose existing query logging, we will have to determine later.

#67 - 08/07/2023 08:50 AM - Galya B

- Assignee deleted (Galya B)

#68 - 08/07/2023 08:52 AM - Greg Shah

- Assignee set to Galya B

#69 - 08/07/2023 08:53 AM - Greg Shah

- % Done changed from 0 to 60

Files

can_multiple_clients_output_to_the_same_file_using_log_manager_psc_knowledgebase_article_p160687.pdf	79.7 KB	01/24/2023	Greg Shah
clientlog-old.png	216 KB	01/17/2023	Galya B
clientlog-new.png	273 KB	01/17/2023	Galya B
fix-batch-output-as-errors.diff	3.48 KB	01/27/2023	Galya B