Runtime Infrastructure - Bug #8206

Logging manager issues

01/19/2024 01:17 PM - Vladimir Tsichevski

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

Description

History

#2 - 01/19/2024 01:47 PM - Vladimir Tsichevski

As of trunk rev 14926, the following issues are known, which relate to legacy logging:

Parse **integer** logging configuration parameters:

1. when formatting error messages the parameter values are used where the parameter names are expected (first reported in #8195-41).
2. the maximum number of digits in an integer parameter is 10, should be 19.
3. configured log threshold size is not upper-limited, as it should (first reported in #8195-50).

Error reporting:

1. 1-2 dots are forcedly appended to error messages, there no dots expected.

4gl unit testing:

1. real parameters are not used to configure legacy logging, so logging configuration is impossible in 4gl unit testing (first reported in #8195-49).
2. a **fixed file name** is used for logging, which prevents proper testing of legacy logging itself in unit tests.

Logging:

1. sending **client** log records to unknown log file does not cause an error.

#3 - 01/22/2024 02:46 AM - Galya B

There is a change introduced in r14848 that deletes a substantial portion of the integer parameters parsing logic in LegacyLogManagerImpl.

```
revno: 14848 [merge]
committer: Theodoros Theodorou <tt@goldencode.com>
branch nick: trunk
timestamp: Wed 2023-11-29 01:56:22 +0200
```

```
message:
  Improved StartupParameters.java (refs #7732)



=== modified file 'src/com/goldencode/p2j/util/LegacyLogManagerImpl.java'
--- src/com/goldencode/p2j/util/LegacyLogManagerImpl.java    2023-08-08 20:49:39 +0000
+++ src/com/goldencode/p2j/util/LegacyLogManagerImpl.java    2023-10-03 22:57:25 +0000
@@ -18,6 +18,7 @@
 **                  file operations on log files client-side when the server-side filesystem config is
 **                  disabled. Entry types to be resolved case-insensitive. Appserver warnings added.
 ** 007 GBB 20230609 Fixes NPEs when configs not present.
+** 008 TT  20230915 Migrated fileSize and numLogFiles from String to Integer.
 */
 /*
 ** This program is free software: you can redistribute it and/or modify
@@ -206,7 +207,7 @@
          client = LogicalTerminal.getInstance().getLegacyLogManagerClient();
       }
       initialize(configs.getLogFile(), configs.getLevelString(), configs.getEntryTypes(),
-                 configs.getFileSizeString(), configs.getNumLogFilesString(), configs.getProcessId(),
+                 configs.getFileSize(), configs.getNumLogFiles(), configs.getProcessId(),
                  configs.getMaxInputCharacters(), configs.isAppserver(),
                  configs.isLoggerInitialized());
       configs.setLoggerInitialized();
@@ -222,9 +223,9 @@
     *          Default logging level.
     * @param   entryTypes
     *          List of comma-separated log entry types.
-    * @param   fileSizeString
+    * @param   fileSize
     *          File size in bytes used as threshold for file roll over.
-    * @param   numLogFilesString
+    * @param   numLogFiles
     *          Max number of log files.
     * @param   clientProcessId
     *          The id of the client process.
@@ -236,8 +237,8 @@
    private synchronized void initialize(String logFile,
                                         String levelString,
                                         String entryTypes,
-                                        String fileSizeString,
-                                        String numLogFilesString,
+                                        Integer fileSize,
+                                        Integer numLogFiles,
                                         long clientProcessId,
                                         int maxInputCharacters,
                                         boolean isAppserver,
@@ -274,20 +275,20 @@
          }
       }

-      if (hasContent(fileSizeString) && !this.setLogThreshold(fileSizeString))
+      if (fileSize != null && !this.setLogThreshold(fileSize))
       {
          LOG.log(Level.CONFIG, "There was an issue with log file threshold value. Initialization stops.");
          return;
       }

-      if (hasContent(numLogFilesString) && !this.setNumLogFiles(numLogFilesString))
+      if (numLogFiles != null && !this.setNumLogFiles(numLogFiles))
       {
          LOG.log(Level.CONFIG, "There was an issue with log files number. Initialization stops.");
          return;
       }

       if (!hasContent(logFile) &&
-         (hasContent(levelString) || hasContent(fileSizeString) || hasContent(numLogFilesString)
 || hasContent(entryTypes)))
+         (hasContent(levelString) || fileSize != null || numLogFiles != null || hasContent(entryTypes)))
       {
          recordOrThrowError( 11068, "Parameters for logging were specified but -clientlog was not set. " +
             "Logging parameters will be ignored. Specify -logginglevel 0 if you want to keep logging " +
@@ -312,7 +313,7 @@
```

```
          " logFile=" + logFile +
          " loggingLevel=" + loggingLevel +
          " logThreshold=" + logThreshold +
-         " numLogFiles=" + numLogFiles +
+         " numLogFiles=" + this.numLogFiles +
          " activatedTypeLevelPairs=" + activatedTypeLevelPairs);
    }

@@ -1414,43 +1415,11 @@
     *
     * Can be set only command line on initialization.
     *
-    * @param    fileSizeString
+    * @param    fileSize
     *           The log file size threshold for rolling over to a new file.
     */
-   private boolean setLogThreshold(String fileSizeString)
+   private boolean setLogThreshold(int fileSize)
    {
-      fileSizeString = fileSizeString.trim();
-      if (!fileSizeString.matches(NUMBER_REGEX))
-      {
-         recordOrThrowError(11996, "The -logthreshold parameter requires a numeric argument.", false);
-         return false;
-      }
-
-      if (fileSizeString.length() > 10)
-      {
-         recordOrThrowError(5049, "The -logginglevel parameter has too many digits.", false);
-         return false;
-      }
-
-      long fileSizeLong;
-      try
-      {
-         fileSizeLong = Long.parseLong(fileSizeString);
-      }
-      catch (NumberFormatException e)
-      {
-         recordOrThrowError(11996, "The -logthreshold parameter requires a numeric argument.", false);
-         return false;
-      }
-
-      if (fileSizeLong < 0 || fileSizeLong > MAX_LOG_FILE_THRESHOLD_BYTES)
-      {
-         recordOrThrowError(5049, "The -logginglevel parameter has too many digits.", false);
-         return false;
-      }
-
-      int fileSize = (int) fileSizeLong;
-
       if (fileSize == 0)
       {
          this.logThreshold = fileSize;
@@ -1493,50 +1462,20 @@
     * @param    numLogFilesString
     *           The number of log files in total.
     */
-   private boolean setNumLogFiles(String numLogFilesString)
+   private boolean setNumLogFiles(int numLogFiles)
    {
-      numLogFilesString = numLogFilesString.trim();
-      if (!numLogFilesString.matches(NUMBER_REGEX))
-      {
-         recordOrThrowError(11996,  "The -numlogfiles parameter requires a numeric argument.", false);
-         return false;
-      }
-
-      if (numLogFilesString.length() > 10)
-      {
-         recordOrThrowError(5049, "The -numlogfiles parameter has too many digits.", false);
-         return false;
-      }
-
-      long numLogFilesLong;
```

```
-       try
-       {
-          numLogFilesLong = Long.parseLong(numLogFilesString);
-       }
-       catch (NumberFormatException e)
-       {
-          recordOrThrowError(11996,  "The -numlogfiles parameter requires a numeric argument.", false);
-          return false;
-       }
-
-       if (numLogFilesLong < 0 || numLogFilesLong > 2_147_483_647)
-       {
-          recordOrThrowError( 5049, "The -logginglevel parameter has too many digits.", false);
-          return false;
-       }
-
-       if (numLogFilesLong == 1)
+       if (numLogFiles == 1)
        {
          recordOrThrowError(11067, "-numlogfiles cannot be set to 1", false);
          return false;
        }

-       if (numLogFilesLong > MAX_NUM_LOG_FILES)
+       if (numLogFiles > MAX_NUM_LOG_FILES)
        {
          recordOrThrowError( 14416, "-numlogfiles cannot be set to a value greater than 999999", false);
          return false;
        }
-       this.numLogFiles = (int) numLogFilesLong;
+       this.numLogFiles = numLogFiles;
        if (this.logWriter != null)
        {
          // only after initialization
```

The changes need to be reverted and the relevant parameters need to be stored in StartupParameters as strings. This will fix the first part of the issues described in the task.

#4 - 01/22/2024 02:48 AM - Galya B

Vladimir Tsichevski wrote:

> Error reporting:
>
> 1. 1-2 dots are forcedly appended to error messages, there no dots expected.

Can you give examples of error messages, how to produce them and what is the expected vs actual outcome.

#5 - 01/22/2024 02:53 AM - Galya B

Theodoros, the change was introduced for #7732. The two parameters in concern are only used in LegacyLogManagerImpl. Can you elaborate on the reasons for the change of the type?

#6 - 01/31/2024 05:57 PM - Theodoros Theodorou

My goal was to remove the duplicate code which checks if the integer startup parameters are valid. I worked on the StartupParameters.java, which includes many integer startup parameters. The lines removed from setLogThreshold() and setNumLogFiles() contain the same logic, as all integer startup parameters I believe.

```
xString = xString.trim();
if (!xString.matches(NUMBER_REGEX))
{
   recordOrThrowError(11996, "The -x parameter requires a numeric argument.", false);
   return false;
}

if (xString.length() > 10)
{
   recordOrThrowError(5049, "The -x parameter has too many digits.", false);
   return false;
}

long xLong;
try
{
   xLong = Long.parseLong(xString);
}
catch (NumberFormatException e)
{
   recordOrThrowError(11996, "The -x parameter requires a numeric argument.", false);
   return false;
}

if (xLong < 0 || xLong > 2_147_483_647)
{
   recordOrThrowError(5049, "The -x parameter has too many digits.", false);
   return false;
}

int x = (int) xLong;
```

I accidentally introduced some bugs, and I started working on resolving them.

#7 - 01/31/2024 06:09 PM - Theodoros Theodorou

- Assignee set to Theodoros Theodorou

- Status changed from New to WIP

#8 - 02/01/2024 02:48 AM - Galya B

A found issue with the numeric parameters in LegacyLogManagerImpl is supposed to stop the initialization process. How is this supposed to work, when the parsing is done outside?

#9 - 02/01/2024 12:46 PM - Theodoros Theodorou

Galya B wrote:

> A found issue with the numeric parameters in LegacyLogManagerImpl is supposed to stop the initialization process. How is this supposed to work, when the parsing is done outside?

Under which circumstances should it stop the initialization process? With what values?

For example, if the following parameter is passed in the client.xml, then an error is shown and the initialization is stopped. This is handled inside IntegerStartupParameter.java. I think this is the default behavior for every integer startup parameter that we pass non-numeric values.

```
<client>
   <cmd-line-option logthreshold="ccc"/>
</client>
```

#10 - 02/01/2024 03:21 PM - Theodoros Theodorou

Vladimir Tsichevski wrote (in #8195-50):

> Also, I found additional minor issues:
>
> 1. FWD reports if the maximum number of digits in integer parameters as more than 10, in OE the limit is 19 digits
> 2. In FWD Error messages are ended with two dots, in OE they have no dots at all.
> 3. OE silently limits the log threshold by 2147483647. FWD does neither check nor limit the upper bound (I am not sure whether we should fix this).

OK, this is getting interesting... I created a file named a.p which contains message log-manager:log-threshold. I tried to run my file using proenv and the command prowin -p a.p -logthreshold [number]. I got the following results using different numbers:

number -> output printed
1234567 -> 1234567
1234567890 -> 1234567890
12345678901 -> Error: The -logthreshold parameter has too many digits.
123456789012 -> Error: The -logthreshold parameter has too many digits.
1234567890123 -> 1912276171
12345678901234 -> 1942892530
123456789012345 -> Error: The -logthreshold parameter has too many digits.
1234567890123456 -> Error: The -logthreshold parameter has too many digits.

12345678901234567 -> 1015724747
123456789012345678 -> 1567312886
1234567890123456789 -> Error: The -logthreshold parameter has too many digits.
12345678901234567890 -> 2112466034
123456789012345678901 -> Error: The -logthreshold parameter has too many digits.
1234567890123456789012 -> 793205908
12345678901234567890123 -> Error: The -logthreshold parameter has too many digits.
123456789012345678901234 -> 2011179506
...
1234567890123467890123456 7 -> 1134812039

I suppose the assumption for the limit of the 19 digits is incorrect. I think OE does some magic and generates some number, but the results above do not indicate the fixed number 2147483647.

Does anyone understand what is going on? Can anyone confirm my findings?

#11 - 02/01/2024 03:58 PM - Vladimir Tsichevski

Theodoros Theodorou wrote:

> Does anyone understand what is going on? Can anyone confirm my findings?

OE parses the number to a 64-bit integer, and interprets the lower 32 bits of the result as a signed integer. If the resulting value is negative (the bit 31 is set), OE reports error, otherwise the 32-bit result is used.

#12 - 02/02/2024 02:46 AM - Galya B

Theodoros Theodorou wrote:

> Galya B wrote:
>
> > A found issue with the numeric parameters in LegacyLogManagerImpl is supposed to stop the initialization process. How is this supposed to work, when the parsing is done outside?
>
> Under which circumstances should it stop the initialization process? With what values?

This is in the original code. The validation stops the initialization of LegacyLogManagerImpl.

> For example, if the following parameter is passed in the client.xml, then an error is shown and the initialization is stopped. This is handled inside IntegerStartupParameter.java. I think this is the default behavior for every integer startup parameter that we pass non-numeric values.
>
> [...]

IntegerStartupParameter.getIntegerParameter does:

```
        PostInitErrorManager.addError(() -> recordOrThrowError(11996, text, false));
        return null;
```

The error is thrown after initialization and the value is considered null for the time being, which is not stopping the logger initialization.

> Does anyone understand what is going on? Can anyone confirm my findings?

Values above 2_147_483_647 bytes (2.1 GB) are a mistake in configuration, because they can't be contained into integers anyways, so it should be OK to use the original validation.

I think what this sub-task needs is to return the validation of string parameters to LegacyLogManagerImpl and if there is any common validation with IntegerStartupParameter reuse the methods.

#13 - 02/02/2024 09:37 AM - Theodoros Theodorou

Galya B wrote:

> The error is thrown after initialization and the value is considered null for the time being, which is not stopping the logger initialization.

There is a method called PostInitErrorManager.checkForErrors() which stops the application if there are any errors and can be called anywhere in the application and as many times as we want. It can be placed right before the logger initialization.

> I think what this sub-task needs is to return the validation of string parameters to LegacyLogManagerImpl and if there is any common validation with IntegerStartupParameter reuse the methods.

Ok I can do it if you believe it is better. Should I proceed?

#14 - 02/02/2024 09:46 AM - Galya B

Theodoros Theodorou wrote:

> Galya B wrote:
>
> > The error is thrown after initialization and the value is considered null for the time being, which is not stopping the logger initialization.
>
> There is a method called PostInitErrorManager.checkForErrors() which stops the application if there are any errors and can be called anywhere in the application and as many times as we want. It can be placed right before the logger initialization.

I expect PostInitErrorManager to run after the initialization as the name suggests, even if you put it on every second line. Then so much will have happened in LOG-MANAGER already.

> Ok I can do it if you believe it is better. Should I proceed?

If you're able to return the original behavior, it doesn't matter how it is done. Test for some of the issues Vladimir noticed in #8206-2. In every case LegacyLogManagerImpl.initialize should not continue execution with faulty logThreshold and numLogFiles, while at the same time the order of validations need to be preserved, since this is replica of the original behavior.

#15 - 02/02/2024 11:29 AM - Theodoros Theodorou

Galya B wrote:

> I expect PostInitErrorManager to run after the initialization as the name suggests, even if you put it on every second line. Then so much will have happened in LOG-MANAGER already.
>
> In every case LegacyLogManagerImpl.initialize should not continue execution with faulty logThreshold and numLogFiles, while at the same time the order of validations need to be preserved, since this is replica of the original behavior.

Currently, PostInitErrorManager.checkForErrors() is called first and LegacyLogManagerImpl.initialize() is called second. If logThreshold and numLogFiles are faulty, checkForErrors() will show the errors and terminate the application. Is this the desired order?

#16 - 02/02/2024 05:01 PM - Theodoros Theodorou

Vladimir Tsichevski wrote:

> As of trunk rev 14926, the following issues are known, which relate to legacy logging:
>
> Parse **integer** logging configuration parameters:
>
> 1. when formatting error messages the parameter values are used where the parameter names are expected (first reported in #8195-41).

Done

> 1. the maximum number of digits in an integer parameter is 10, should be 19.

I think this assumption is incorrect. It can accept more than 19 digits.

> 1. configured log threshold size is not upper-limited, as it should (first reported in #8195-50).
>
>   OE parses the number to a 64-bit integer, and interprets the lower 32 bits of the result as a signed integer. If the resulting value is negative (the bit 31 is set), OE reports error, otherwise the 32-bit result is used.

Galya B wrote:

> Values above 2_147_483_647 bytes (2.1 GB) are a mistake in configuration, because they can't be contained into integers anyways, so it should be OK to use the original validation.

I kept the original validation logic.

> Error reporting:
>
> 1. 1-2 dots are forcedly appended to error messages, there no dots expected.

Done

> 4gl unit testing:
>
> 1. real parameters are not used to configure legacy logging, so logging configuration is impossible in 4gl unit testing (first reported in #8195-49).
> 2. a **fixed file name** is used for logging, which prevents proper testing of legacy logging itself in unit tests.

Indeed, inside the UnitTestServer.java, a new LegacyLogManagerConfigs("unittestserver.log"... with a fixed log file name is created. Do you want me to add startupParameters.getClientLog() in its place, as we use in StandardServer.java? How should we handle the case where -clientlog is not set? Should I initialize it as "unittestserver.log" if startupParameters.getClientLog() is null?

> Logging:
>
> 1. sending **client** log records to unknown log file does not cause an error.

I am not aware of this functionality, I was not the one who implemented it. Can you please give me more details on how to recreate it?

#17 - 02/09/2024 06:31 PM - Theodoros Theodorou

Kind reminder to check the comments above so we can proceed merging the bug fixes

#18 - 02/12/2024 02:04 AM - Galya B

Theodoros Theodorou wrote:

1. the maximum number of digits in an integer parameter is 10, should be 19.

I think this assumption is incorrect. It can accept more than 19 digits.

If the explanation of Vladimir about how OE parses numbers and cuts integers out of longs, then it might be true. Otherwise Integer is max 2147483647, so 10 digits.

1. configured log threshold size is not upper-limited, as it should (first reported in #8195-50).

   OE parses the number to a 64-bit integer, and interprets the lower 32 bits of the result as a signed integer. If the resulting value is negative (the bit 31 is set), OE reports error, otherwise the 32-bit result is used.

If this is confirmed, then I guess we can recreate the unexpected behavior of cut numbers.

Error reporting:

1. 1-2 dots are forcedly appended to error messages, there no dots expected.

Done

I still need an example from Vladimir, because the messages I tested were well formatted back when I worked on it, so the question in #8206-4 still stands.

4gl unit testing:

1. real parameters are not used to configure legacy logging, so logging configuration is impossible in 4gl unit testing (first reported in #8195-49).
2. a **fixed file name** is used for logging, which prevents proper testing of legacy logging itself in unit tests.

Indeed, inside the UnitTestServer.java, a new LegacyLogManagerConfigs("unittestserver.log"... with a fixed log file name is created. Do you want me to add startupParameters.getClientLog() in its place, as we use in StandardServer.java? How should we handle the case where -clientlog is not set? Should I initialize it as "unittestserver.log" if startupParameters.getClientLog() is null?

I think it should be like in StandardServer. A default value will fail the LOG-MANAGER tests, so don't use it.

Logging:

1. sending **client** log records to unknown log file does not cause an error.

I am not aware of this functionality, I was not the one who implemented it. Can you please give me more details on how to recreate it?

I also don't know what Vladimir means here. Check the code in LegacyLogManagerImpl:

```
         File newLogFile = new File(logFileConfig);
         if (!isWritable(newLogFile))
         {
            recordOrShowWarning(11076, "Cannot open log file " + logFileConfig + ", errno 3", false);
            recordOrThrowError(4487, "Unable to open file for PSEUDO-WIDGET.", true);
            return;
         }
```

#19 - 02/12/2024 06:54 AM - Vladimir Tsichevski

Galya B wrote:

> I still need an example from Vladimir, because the messages I tested were well formatted back when I worked on it, so the question in #8206-4 still stands.

1. In OE there is no dot at the end of the message:

```
      if (fileSize < MIN_LOG_FILE_THRESHOLD_BYTES)
      {
         recordOrThrowError(11065, "-logthreshold must be set to a value between 500000 and 2147483647.",
                                                                                                  ^^^^^^
                            false);
         return false;
      }
```

2. Here another dot is added to all log manager error messages. If the original messages also contains a dot, two dots are present as the result.

```
   static void recordOrThrowLogManagerError(int num, String text, boolean prefix)
   {
      recordOrThrowError(new int[] { num }, new String[] { text }, prefix, false, true, true);
                                                                                 ^^^^
   }
```

I can be wrong (needs to be checked), but in OE no log manager error messages are terminated by a dot.

All this was fixed in my change we decided to revert.

#20 - 02/12/2024 11:53 AM - Galya B

Vladimir Tsichevski wrote:

> Galya B wrote:
>
> > I still need an example from Vladimir, because the messages I tested were well formatted back when I worked on it, so the question in #8206-4 still stands.
>
> 1. In OE there is no dot at the end of the message:
> [...]
> 2. Here another dot is added to all log manager error messages. If the original messages also contains a dot, two dots are present as the result.
> [...]
>
> I can be wrong (needs to be checked), but in OE no log manager error messages are terminated by a dot.
>
> All this was fixed in my change we decided to revert.

Yes, this needs to be checked for all messages.

#21 - 02/13/2024 04:47 PM - Theodoros Theodorou

Galya B wrote:

> Theodoros Theodorou wrote:
>
> > 1. the maximum number of digits in an integer parameter is 10, should be 19.
>
> I think this assumption is incorrect. It can accept more than 19 digits.
>
> If the explanation of Vladimir about how OE parses numbers and cuts integers out of longs, then it might be true. Otherwise Integer is max 2147483647, so 10 digits.
>
> > 1. configured log threshold size is not upper-limited, as it should (first reported in #8195-50).
> >
> > OE parses the number to a 64-bit integer, and interprets the lower 32 bits of the result as a signed integer. If the resulting value is negative (the bit 31 is set), OE reports error, otherwise the 32-bit result is used.

> If this is confirmed, then I guess we can recreate the unexpected behavior of cut numbers.

OE can accept numbers of more than 19 digits. I was able to recreate the exact same behavior by using Java's BigInteger. The change has been pushed to the branch 8206a rev no 14961.

#22 - 02/13/2024 04:51 PM - Theodoros Theodorou

Vladimir Tsichevski wrote:

> Galya B wrote:
>
> > I still need an example from Vladimir, because the messages I tested were well formatted back when I worked on it, so the question in #8206-4 still stands.
>
> 1. In OE there is no dot at the end of the message:
> [...]
> 2. Here another dot is added to all log manager error messages. If the original messages also contains a dot, two dots are present as the result.
> [...]
>
> I can be wrong (needs to be checked), but in OE no log manager error messages are terminated by a dot.
>
> All this was fixed in my change we decided to revert.

You were right. I made the changes and pushed them to 8206a rev no 14962. Please check them and let me know if the changes are ok

#23 - 02/14/2024 01:22 AM - Galya B

Theodoros Theodorou wrote:

> Galya B wrote:
>
> > Theodoros Theodorou wrote:
> >
> > 1. the maximum number of digits in an integer parameter is 10, should be 19.

I think this assumption is incorrect. It can accept more than 19 digits.

If the explanation of Vladimir about how OE parses numbers and cuts integers out of longs, then it might be true. Otherwise Integer is max 2147483647, so 10 digits.

1. configured log threshold size is not upper-limited, as it should (first reported in #8195-50).

   OE parses the number to a 64-bit integer, and interprets the lower 32 bits of the result as a signed integer. If the resulting value is negative (the bit 31 is set), OE reports error, otherwise the 32-bit result is used.

If this is confirmed, then I guess we can recreate the unexpected behavior of cut numbers.

OE can accept numbers of more than 19 digits. I was able to recreate the exact same behavior by using Java's BigInteger. The change has been pushed to the branch 8206a rev no 14961.

I guess you keep into consideration the LOG-MANAGER errors:

- 11065, "-logthreshold must be set to a value between 500000 and 2147483647."
- 5049, "The -numlogfiles parameter has too many digits." for pro -b -p procedure.p -clientlog file.log -numlogfiles 12345678901.
- 5049, "The -logginglevel parameter has too many digits." for pro -b -p procedure.p -clientlog file.log -logthreshold 12345678901.

#24 - 02/14/2024 07:52 AM - Vladimir Tsichevski

Theodoros Theodorou wrote:

I made the changes and pushed them to 8206a rev no 14962. Please check them and let me know if the changes are ok

Review of the changes 8206a revs. 14960..14962

StartupParameters:

1. no history entry
2. import com.goldencode.p2j.directory.DirectoryService is not used.

IntegerStartupParameter:

1. import com.goldencode.p2j.main.* is never used.
2. the way to check an integer parameter with this regexp is incorrect:

```
   private Integer getIntegerParameter(String parameter) {

      String param = parameter.trim();
      if (!param.matches("-?\\d+(\\.\\d+)?"))
      {
```

   It allows floating point numbers to be used, which is no allowed in OE.
   It does not allow using the + sign, which is allowed in OE.
   I think, it is better to use Integer.parseInt() and catch exceptions.
3. Negative values are considered as error here. I gather, the IntegerStartupParameter class is to be used for reading **any** Progress integer startup parameter, and I am not sure negative values are forbidden for **all** command-line parameters in Progress.

UnitTestServer:

1. no history entry
2. the parameters passed to LegacyLogManagerConfigs are still wrong: they must be configurable.

ErrorManager:

1. no history entry
2. throws ErrorConditionException: in Javadocs is missing description.

LegacyLogManagerConfigs:

1. no history entry

LegacyLogManagerImpl:

1. no history entry
2. setNumLogFiles(int numLogFiles): the return type is not described in javadocs

#25 - 02/14/2024 04:52 PM - Theodoros Theodorou

Vladimir Tsichevski wrote:

> IntegerStartupParameter:
>
> 1. the way to check an integer parameter with this regexp is incorrect:
>   [...]
>   It allows floating point numbers to be used, which is no allowed in OE.
>   It does not allow using the + sign, which is allowed in OE.
>   I think, it is better to use Integer.parseInt() and catch exceptions.

You are right, the regex was incorrect. I used BigInteger to catch exceptions.

> 1. Negative values are considered as error here. I gather, the IntegerStartupParameter class is to be used for reading **any** Progress integer startup parameter, and I am not sure negative values are forbidden for **all** command-line parameters in Progress.

I have this in mind. If we ever encounter negative integer startup parameters in the future, I can think of many solutions. We can just create a new class named SignedIngegerStartupParameter or maybe add a flag boolean acceptOnlyPositiveNumbers. There is no reason to take any action now.

I have addressed all issues in 8206a rev no 14964. Please review again.

#26 - 02/15/2024 02:14 PM - Theodoros Theodorou

Galya B wrote:

> I guess you keep into consideration the LOG-MANAGER errors:
>
> - 11065, "-logthreshold must be set to a value between 500000 and 2147483647."
> - 5049, "The -numlogfiles parameter has too many digits." for pro -b -p procedure.p -clientlog file.log -numlogfiles 12345678901.
> - 5049, "The -logginglevel parameter has too many digits." for pro -b -p procedure.p -clientlog file.log -logthreshold 12345678901.

Yes, I doublechecked them and they look fine.

#27 - 02/15/2024 03:00 PM - Theodoros Theodorou

Rebased 8206a from trunk revision 14986. The latest revision in this branch is now 14992.

#28 - 04/29/2024 04:33 PM - Theodoros Theodorou

- Status changed from WIP to Review

#29 - 05/02/2024 10:53 AM - Greg Shah

- Status changed from Review to Internal Test

- % Done changed from 0 to 100

Code Review Task Branch 8206a Revisions 14987 through 14992

I have no objections to the changes.

As far as I know, this task is complete.

Vladimir/Galya: I understand that you are OK with these changes as well?

#30 - 05/02/2024 10:54 AM - Greg Shah

- Assignee changed from Theodoros Theodorou to Galya B

With Theodoros gone, this will need to be rebased, tested and merged by someone else.

#31 - 05/02/2024 01:24 PM - Vladimir Tsichevski

Greg Shah wrote:

> Code Review Task Branch 8206a Revisions 14987 through 14992
>
> Vladimir/Galya: I understand that you are OK with these changes as well?

No, the initialization of LegacyLogManagerConfigs in UnitTestServer is still "undercooked", as Galya would say. Please, see the reverted change in 8195a:

```
revno: 14939
committer: vvt@goldencode.com
branch nick: 8195a
timestamp: Fri 2024-01-19 19:04:06 +0300
message:
  The previous change 14929 reverted to 14928
------------------------------------------------------------
revno: 14938
author: Vladimir Tsichevski <vvt@wowa.org>
committer: vvt@goldencode.com
branch nick: 8195a
timestamp: Thu 2024-01-18 19:54:15 +0300
message:
  Fixed issues listed in #8195-49 and #8195-50
```

#32 - 05/03/2024 01:21 AM - Galya B

- Assignee changed from Galya B to Vladimir Tsichevski

- % Done changed from 100 to 80

Actually Vladimir is better familiar with these issues and has more insight on what needs to be achieved.

Vladimir, please commit the necessary changes and let me know if you need a review.

#33 - 05/03/2024 06:29 AM - Vladimir Tsichevski

- % Done changed from 80 to 90

- Status changed from Internal Test to WIP

- Assignee changed from Vladimir Tsichevski to Galya B

OK, I'll rebase the branch and add the missing change.

#34 - 05/03/2024 06:30 AM - Vladimir Tsichevski

- Assignee changed from Galya B to Vladimir Tsichevski

#35 - 05/03/2024 09:21 AM - Vladimir Tsichevski

- Status changed from WIP to Review

- % Done changed from 90 to 100

Log manager initialization fixed in 8206a rev. 15197. Please, review.

#36 - 05/03/2024 09:44 AM - Galya B

Vladimir Tsichevski wrote:

> Log manager initialization fixed in 8206a rev. 15197. Please, review.

I'm not sure if this is supposed to work for big numbers IntegerStartupParameter:

```
int integer = bigInteger.intValue();
if (integer < 0)
{
   String text = "The -" + name + " parameter has too many digits";
   PostInitErrorManager.addError(() -> recordOrThrowError(5049, text, false));
   return null;
}
```

BigInteger.intValue() (https://docs.oracle.com/javase/8/docs/api/java/math/BigInteger.html#intValue):

> This conversion is analogous to a narrowing primitive conversion from long to int as defined in section 5.1.3 of The Java™ Language Specification: if this BigInteger is too big to fit in an int, only the low-order 32 bits are returned.

So if the number is bigger than max int, this message won't be shown.

#37 - 05/03/2024 10:55 AM - Vladimir Tsichevski

Galya B wrote:

> Vladimir Tsichevski wrote:
>
> > Log manager initialization fixed in 8206a rev. 15197. Please, review.
>
> I'm not sure if this is supposed to work for big numbers IntegerStartupParameter:
> [...]
> BigInteger.intValue() (https://docs.oracle.com/javase/8/docs/api/java/math/BigInteger.html#intValue):
>
> > This conversion is analogous to a narrowing primitive conversion from long to int as defined in section 5.1.3 of The Java™ Language Specification: if this BigInteger is too big to fit in an int, only the low-order 32 bits are returned.
>
> So if the number is bigger than max int, this message won't be shown.

This is the response on the #8206-11 finding and needs to be tested. I am not sure it we have to implement this strange behavior in FWD though.