Aggressive Log Report

Although we have quite nice Error Message Dialog in midPoint there are some occasions when this is not enough to diagnose the problem. Such (relative rare) occasions usually hits about a programming bug: wrong exception handling, wrong error reporting a wild variety (usually stupid) programming mistakes. If the usual troubleshooting steps do not work, there is a time for a brutal solution.

The best way is to show midPoint team how to replicate the problem. But some problems cannot be replicated. Everything works perfectly fine in our environment but fails in your environment. If there is no associated error message (or the message is incomplete) and the logs do not show any errors or any other suspicious activity there is still one way how to let us know what is going on: aggressive logging.

Let midPoint log everything for a short period of time and deliver the log to the midPoint team. We should be able to figure out what is happening. But it will be very difficult for us to find our way through megabytes of logs. Therefore there is a procedure how to reduce the amount of logging just to the required minimum.

Turn off all Threads

If your problem is not related to tasks then the best strategy is to turn them off. The tasks wake up once in a while and create additional "noise" in the logs. Even if you have no active tasks, the task manager itself is producing some noise. Turning off the tasks is not intended for production systems as it may impact the operation, E.g. it will stop live synchronization or workflows. But it is a good strategy for this kind of "last resort" debugging.

To turn off the tasks go to the **Server Tasks** page. Click **Stop all threads** red button on the bottom of the page.

From this point on all tasks and other threads will be stopped. Only the operations that you explicitly do in the GUI or through other interfaces will be executed on this midPoint node. You can reactivate threads when you are done with this procedure later by clicking the other button in the dialog.

Increase Logging Granularity

Navigate to the **Configuration > Logging** page. The goal is to increase logging to the maximum. Click **Add component logger** and then select **All** and **TRACE** options. Don't forget to click **Save**. MidPoint will produce considerable amount of log records from this point on.

If you know in which component is the problem located you can experiment with this setting. But if all else fails the All/TRACE will do. The explanation of logging levels can be found on Log Levels page.

Operation Trace

MidPoint can trace the execution of any operation using a "trace" mechanism. When enabled it records entry and exit of operations between midPoint subsystems. This is usually intended for developers but it may come handy when troubleshooting really devious problems. It can give other log records a *perspective* by showing where exactly the error happened and what has midPoint tries to do before that.

Trace can also be used to locate performance problems. The trace messages contain indications of how long the operation took. Therefore it can be used to roughly locate a bottleneck if some of the operations are slow.

The trace can be configured in **Configuration > Logging** GUI page by setting the **Log subsystem entry/exit** option.

Prepare to Execute an Operation

Use the GUI to navigate to the page where you intend to do the operation. Do all the steps except for the very last step. E.g. fill in the form with all data but **do not click Save yet**.

Trim the Logfiles

The log now contains a lot of "noise" caused by navigation through the GUI, preparing the operation, etc. We usually do not need that. And if all that is in the logs it makes it difficult for us to find the exact point when the problematic operation started. Therefore the best strategy is to get rid of that noise entirely. The logfiles are usually in the `logs` subdirectory of your application server. The logfile of midpoint is `midpoint.log`.

But it is a bit tricky. The server has the logfile still open. If you are on UNIX-like system it just cannot be deleted as the server will still write to the deleted file. The usual trick is to trim the file, reducing its size to zero. There are several commands that can be used in UNIX to do it. Choose any one that you like:

```
cp /dev/null midpoint.log
```

```
echo -n > midpoint.log
```

On MS Windows systems with PowerShell installed it is possible to use the following:

```
powershell clear-content midpoint.log
```

If there's no PowerShell, you can use the following Java code, found e.g. in the Log Viewer Utilities package:

```
public static void main(String[] args) throws IOException {
    RandomAccessFile file = new RandomAccessFile("midpoint.log", "rw");
    file.setLength(0);
    file.close();
    System.out.println("Done.");
}
```

Execute the Operation

Switch back to the browser and execute the last step of the prepared operation, e.g. by clicking "Save".

Gather the Logs

There should be a complete trace of the operation in the logfiles. Copy the logfiles to any safe location. Please make sure you **do not execute any other operation** before the logfiles are copied. Doing so will add additional noise at the end of the logfiles.

Bug Report

Send the gathered logfiles to the midPoint team with all the other relevant details which usually are:

- What operation have you tried
- If there is a form or other input tell us how it was filled in (e.g. an XML snippet used to import, etc.)
- What kind of resource definition was used (we prefer if you replicate the problem with one of our samples, see Creating a Bug Report.
- Any other special settings that you feel can influence the outcome (custom schema, strange things in expressions, etc.)
- If the operation produced an error message in GUI include that error message as well
- ... and of course the gathered log files

Clean Up

Turn the logging back to normal using **Configuration > Logging** page. Turn the threads on using the dialog in **Server Tasks** page.