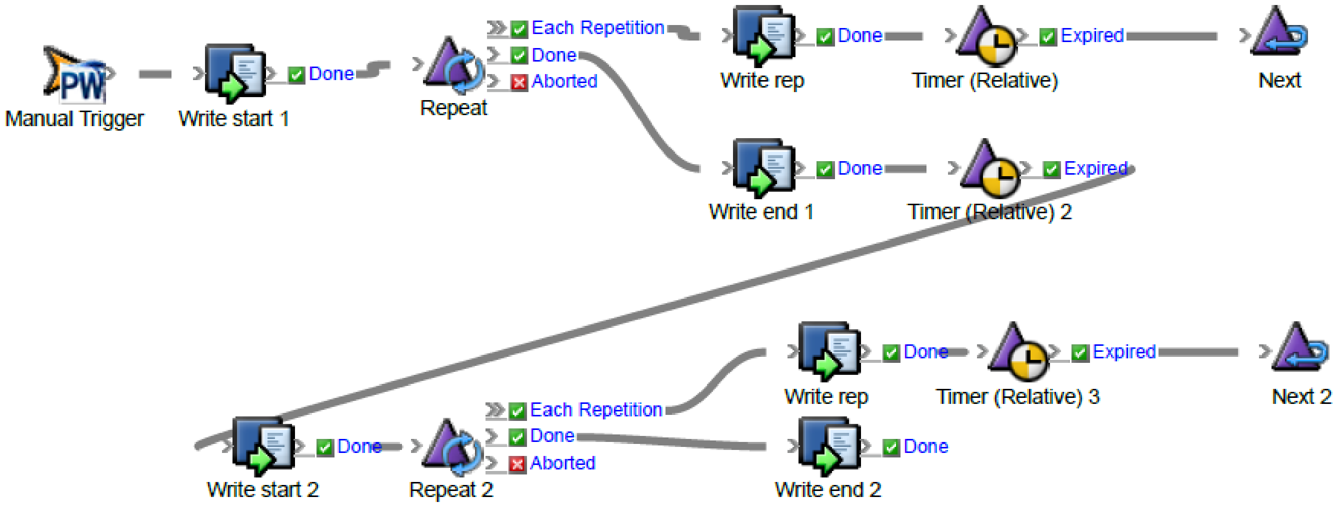
Examples of using limiting rule set concurrency

No concurrency control

The following example rule set is composed of two loops. Each loop uses the **Write To File** action to write several lines of text to a text file with a small delay between the writing of each line.



If a single instance is run, the output looks like this:

```
************************* Starting Loop 1 *****************************
Loop 1 - Iteration: 1
Loop 1 - Iteration: 2
Loop 1 - Iteration: 3
************************* End of Loop 1 *******************************
####################### Start of Loop 2 ###############################
Loop 2 - iteration: 1
Loop 2 - iteration: 2
Loop 2 - iteration: 3
####################### End of Loop 2 #################################
```

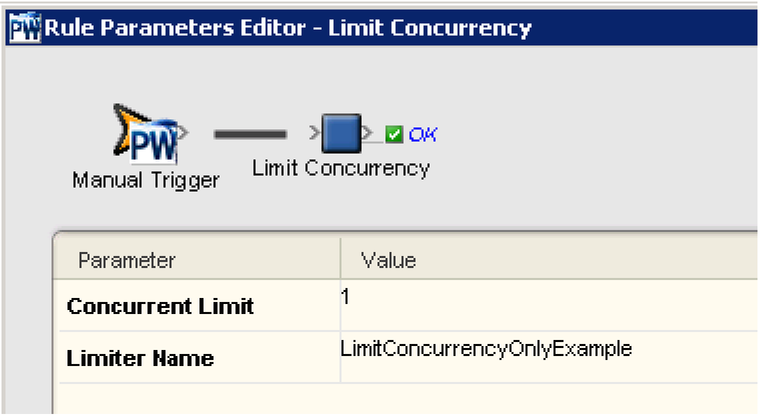
If however multiple instances are run in quick succession, the output looks like this:

```
************************* Starting Loop 1 *************************
Loop 1 - Iteration: 1
************************* Starting Loop 1 *************************
Loop 1 - Iteration: 1
************************* Starting Loop 1 *************************
Loop 1 - Iteration: 1
Loop 1 - Iteration: 2
Loop 1 - Iteration: 2
Loop 1 - Iteration: 2
Loop 1 - Iteration: 3
Loop 1 - Iteration: 3
Loop 1 - Iteration: 3
************************* End of Loop 1 ***************************
************************* End of Loop 1 ***************************
************************* End of Loop 1 ***************************
####################### Start of Loop 2 ###########################
Loop 2 - iteration: 1
####################### Start of Loop 2 ###########################
Loop 2 - iteration: 1
####################### Start of Loop 2 ###########################
Loop 2 - iteration: 1
Loop 2 - iteration: 2
Loop 2 - iteration: 2
Loop 2 - iteration: 2
Loop 2 - iteration: 3
Loop 2 - iteration: 3
Loop 2 - iteration: 3
####################### End of Loop 2 #############################
####################### End of Loop 2 #############################
####################### End of Loop 2 #############################
```

Because each instance is writing to the file at the same time, the output from the instances is interleaved.

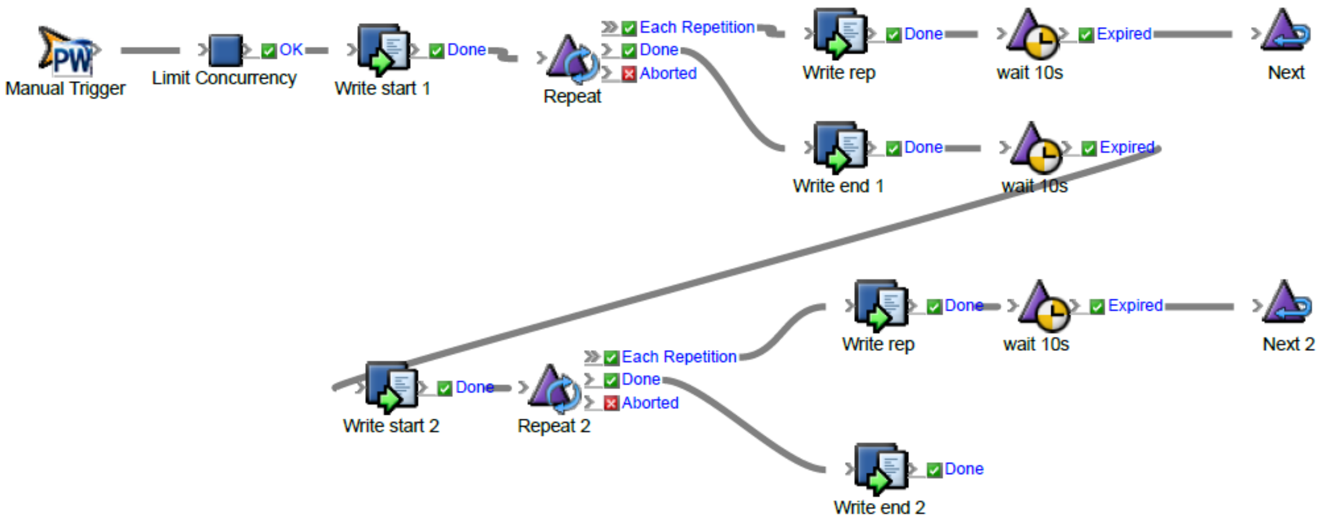
Limit Concurrency only

In the following example only the **Limit Concurrency** action is used. It is configured to allow only a single instance to run. The limiter is named `LimitConcurrencyOnlyExample`.



Parameter	Value
Concurrent Limit	1
Limiter Name	LimitConcurrencyOnlyExample

From the time the **Limit Concurrency** action executes until the rule chain completes, no other instances of the rule chain will run. They will instead queue at the **Limit Concurrency** action until the currently executing instances completes. If there are multiple rule chain instances queued at the **Limit Concurrency** action, they will be released in the same order that they were queued:



If the above rule set is enabled in a job and triggered three times in quick succession, a text file, which contains the following output, is generated:

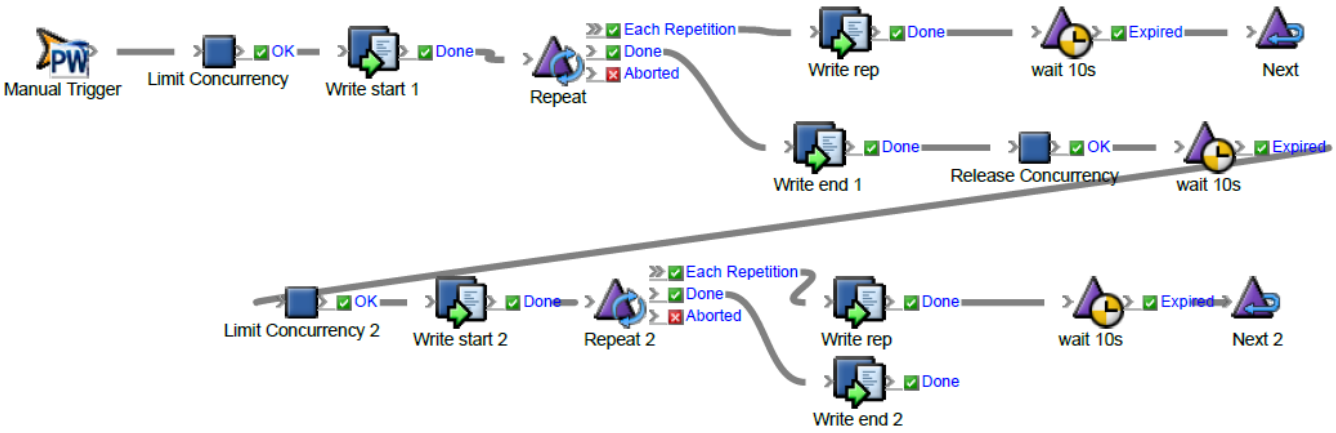
```
************************ Starting Loop 1 ************************
```

```
Loop 1 - Iteration: 1
Loop 1 - Iteration: 2
Loop 1 - Iteration: 3
************************* End of Loop 1 ***************************
##################### Start Loop 2 ###############################
Loop 2 - repetition: 1
Loop 2 - repetition: 2
Loop 2 - repetition: 3
##################### End of Loop 2 ###############################
************************* Starting Loop 1 *************************
Loop 1 - Iteration: 1
Loop 1 - Iteration: 2
Loop 1 - Iteration: 3
************************* End of Loop 1 ***************************
###################### Start Loop 2 ##############################
Loop 2 - repetition: 1
Loop 2 - repetition: 2
Loop 2 - repetition: 3
###################### End of Loop 2 ##############################
************************* Starting Loop 1 *************************
Loop 1 - Iteration: 1
Loop 1 - Iteration: 2
Loop 1 - Iteration: 3
************************* End of Loop 1 ***************************
##################### Start Loop 2 ###############################
Loop 2 - repetition: 1
Loop 2 - repetition: 2
Loop 2 - repetition: 3
##################### End of Loop 2 ###############################
```

Note that only one instance of the chain was allowed to write to the file at once. Each instance completed both loops before the next instance was allowed to proceed. This is because when each instance acquires the limiter at the **Limit Concurrency** action, it holds it until it completes both loops. At the end of the second loop the limiter is automatically released because it is the end of the rule chain. The next instance waiting at the **Limit Concurrency** action is then able to acquire the limiter and proceed.

Limit Concurrency and Release Concurrency

In the following example both the **Limit Concurrency** action and the **Release Concurrency** action are used. The regions of the rule chain delimited by the **Limit Concurrency** and **Release Concurrency** action pair will be limited to a single instance running. Any other instances will queue at the **Limit Concurrency** action until the currently executing instances reaches the **Release Concurrency** action. The next waiting instance will then begin executing the actions within the **Limit Concurrency** to **Release Concurrency** region.



If the above rule set is enabled in a job and triggered three times in quick succession, a text file, which contains the following output, is generated:

```
************************* Starting Loop 1 *************************
Loop 1 - Iteration: 1
Loop 1 - Iteration: 2
Loop 1 - Iteration: 3
************************* End of Loop 1 ***************************
************************* Starting Loop 1 *************************
Loop 1 - Iteration: 1
Loop 1 - Iteration: 2
Loop 1 - Iteration: 3
************************* End of Loop 1 ***************************
************************* Starting Loop 1 *************************
Loop 1 - Iteration: 1
Loop 1 - Iteration: 2
Loop 1 - Iteration: 3
************************* End of Loop 1 ***************************
##################### Start Loop 2 ###############################
Loop 2 - repetition: 1
Loop 2 - repetition: 2
Loop 2 - repetition: 3
##################### End of Loop 2 ###############################
###################### Start Loop 2 ##############################
Loop 2 - repetition: 1
Loop 2 - repetition: 2
Loop 2 - repetition: 3
###################### End of Loop 2 ##############################
##################### Start Loop 2 ###############################
Loop 2 - repetition: 1
Loop 2 - repetition: 2
Loop 2 - repetition: 3
##################### End of Loop 2 ###############################
```

Note that as in the previous example, the output within the loop is not interleaved as it was in the first example. But it is different from the previous example in that each instance was allowed to complete the first loop before the first instance was allowed to proceed with the second loop. This is because between the two loops the instance is releasing and re-acquiring the limiter which results in it effectively going to the end of the queue and having to wait its turn for the limiter again.