Item 240
git_comments:

git_commits:

1. **summary:** NIFI-730: Added additional testing logic to unit tests
   message: NIFI-730: Added additional testing logic to unit tests

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Purge Queue from UI
   description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
2. **summary:** Purge Queue from UI
   description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
3. **summary:** Purge Queue from UI
   description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
4. **summary:** Purge Queue from UI
   description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
5. **summary:** Purge Queue from UI
   description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
6. **summary:** Purge Queue from UI
   description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
7. **summary:** Purge Queue from UI
   description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
8. **summary:** Purge Queue from UI
   description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
9. **summary:** Purge Queue from UI
   description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
10. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
11. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
12. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec),

then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.

13. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
14. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
15. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
16. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
17. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
18. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
19. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
20. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
21. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
22. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
23. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
24. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
25. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
    label: code-design
26. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
    label: code-design
27. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec),

then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.

28. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
29. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
30. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
31. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
32. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
33. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
34. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
35. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
36. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
37. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
38. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
39. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
40. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
41. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
42. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
43. **summary:** Purge Queue from UI

description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.

44. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
45. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
46. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
47. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
48. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
49. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
50. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
51. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
52. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
53. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
54. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
55. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.
    label: code-design
56. **summary:** Purge Queue from UI
    description: Making changes to connections between NiFi processors is difficult when data is queued up between those processors. A workaround to this data build up is to stop both processors, reduce/set the FlowFile Expiration of the connection to a low number (e.g., 2 sec), then start the receiving processor to age off the data. A more user-friendly solution is to provide a "Purge" or "Delete" context menu option on the queues to remove the data and/or age it off immediately.

jira_issues_comments:

1. Commit e0ac7cde372f428b0655465b7adc59ad41f8f270 in nifi's branch refs/heads/NIFI-730 from [~mcgilman] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=e0ac7cd] NIFI-730: - Starting to add support for deleting flow files from a queue by creating endpoints and starting to wire everything together. - Adding context menu item for initiating the request to drop flow files.
2. Commit b8c51dc35d1a7fdbf3e6449bbe297db667a1176c in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=b8c51dc] NIFI-730: Added methods for dropping queued flowfiles; refactored swap manager but have

not yet started swapping flowfiles in or out from within the flowfile queue
3. Commit 49a781df2d44859ec59672c2755b7346452cd74a in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=49a781d] NIFI-730: Implemented swapping in and out on-demand by the FlowFileQueue rather than in a background thread
4. Commit 9be37914ddb9c8c017cc4d6b3209340a19a7cb8d in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=9be3791] Merge branch 'NIFI-730' of https://git-wip-us.apache.org/repos/asf/nifi into NIFI-730
5. Commit 9be37914ddb9c8c017cc4d6b3209340a19a7cb8d in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=9be3791] Merge branch 'NIFI-730' of https://git-wip-us.apache.org/repos/asf/nifi into NIFI-730
6. Commit ad6af95d079c277bf2a7bc63277f14a91fddebf7 in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=ad6af95] NIFI-730: Fixed checkstyle violations
7. Commit 09d6fe5cdb93ed3660263165f909e088244796db in nifi's branch refs/heads/NIFI-730 from [~mcgilman] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=09d6fe5] NIFI-730: - Fixing checkstyle violations. - Renaming context menu item for emptying a queue.
8. Commit af78354d84dcd72a23f1101567729a04ee008110 in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=af78354] NIFI-730: Added additional parameters to dropFlowFiles
9. Commit 4b41aaab029518909b58bf069a0c8b2a2bdca91b in nifi's branch refs/heads/NIFI-730 from [~mcgilman] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=4b41aaa] NIFI-730: - Fixing checkstyle violations. - Wiring endpoints and framework model. - Lowering the max delay while polling from 8 seconds to 4 seconds.
10. Commit 72ff2a25d5455442d5aec27835f953e6ab36eca3 in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=72ff2a2] NIFI-730: Updated queue sizes appropriately during dropping of flowfiles
11. Commit afb76afcd0fd7d0c144a37621fdabc181bd42307 in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=afb76af] NIFI-730: Added error messages if we fail to drop FlowFiles from queue
12. Commit 77f7d7524cb8b07ed2976088f0e57d99233c8327 in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=77f7d75] NIFI-730: bug fixes and code cleanup for swap manager and flowfile queue
13. Commit 0af1acaafaf28844b31c41e276c4dbce18390acb in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=0af1aca] NIFI-730: Return DropFlowFileStatus object when calling cancel
14. Commit 5867193bc131962d05985f405ea199840a99feb9 in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=5867193] NIFI-730: Added additional testing logic to unit tests
15. Commit a2ae99f89965a3fe1bd6591204bdb187a377ae1c in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=a2ae99f] NIFI-730: Make cancel request actually cancel
16. Commit cad0e7cf0f7b5c8f0d4607c55a514e419a1caafc in nifi's branch refs/heads/NIFI-730 from [~mcgilman] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=cad0e7c] NIFI-730: - Adding progress bar for showing current status of drop request. - Allowing the user to cancel the current drop request.
17. Commit 98a04eec741d5ed72405c6549970de5f1281239e in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=98a04ee] NIFI-730: Set current queue size on drop flowfile request immediately when creating request
18. Commit 09a3f6dadd5ccff75590c53d071cc4ebe6c6175b in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=09a3f6d] NIFI-730: reordered states for drop flowfile request
19. Commit 39a050d2fdc2437576d860b968ab1ec78d18fa21 in nifi's branch refs/heads/NIFI-730 from [~mcgilman] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=39a050d] NIFI-730: - Adding emptying a queue when clustered.
20. Commit b8dbd1018cf0b2d09bf7371849cf8b588d5f3f7c in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=b8dbd10] NIFI-730: Updated details of Provenance Event for when a queue is emptied
21. Commit 2b4999c0185ad6009dda33badfef3ade6cb2fde6 in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=2b4999c] NIFI-730: do not interrupt swap thread if drop flowfiles is canceled
22. Commit a872403831a2e816ae41c5ac11d43f7067ce32cc in nifi's branch refs/heads/NIFI-730 from [~mcgilman] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=a872403] NIFI-730: - Updating the styling of the drop request status dialog. - Rendering any errors that may have occurred.
23. Commit 570202eb3059595394599300e810c22eb801d3cd in nifi's branch refs/heads/NIFI-730 from [~mcgilman] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=570202e] NIFI-730: - Fixing capitalization to remain consistent.
24. Reviewing
25. **body:** Code looks good and contrib all passes. For general cases, the flow has been working fine. I have yet to test against a clustered instance, but have run into some issues on my standalone. My flow for testing was simply a GenerateFlowFile sending to an UpdateAttribute. This seemed to work quite well overall. I then configured the connection between the two components to have an expiration of flowfiles after 30 seconds. I let this run for a while (queue ~2GB of 1 kB files) and then initiated a purge. At this point, I noticed significant swapping in the logs to the point where it seemed that was all that was occurring. {code} 2015-10-27 12:35:31,581 INFO [Drop FlowFiles for Connection 239dc171-e1d9-4782-8ead-e4f31bee3496] o.a.n.c.r.WriteAheadFlowFileRepository Repository updated to reflect that 10000 FlowFiles were swapped in to FlowFileQueue[id=239dc171-e1d9-4782-8ead-e4f31bee3496] 2015-10-27 12:35:32,007 INFO [Drop FlowFiles for Connection 239dc171-e1d9-4782-8ead-e4f31bee3496] o.a.n.c.r.WriteAheadFlowFileRepository Repository updated to reflect that 10000 FlowFiles were swapped in to FlowFileQueue[id=239dc171-e1d9-4782-8ead-e4f31bee3496] 2015-10-27 12:35:32,433 INFO [Drop FlowFiles for Connection 239dc171-e1d9-4782-8ead-e4f31bee3496] o.a.n.c.r.WriteAheadFlowFileRepository Repository updated to reflect that 10000 FlowFiles were swapped in to FlowFileQueue[id=239dc171-e1d9-4782-8ead-e4f31bee3496] 2015-10-27 12:35:32,829 INFO [Drop FlowFiles for Connection 239dc171-e1d9-4782-8ead-e4f31bee3496] o.a.n.c.r.WriteAheadFlowFileRepository Repository updated to reflect that 10000 FlowFiles were swapped in to FlowFileQueue[id=239dc171-e1d9-4782-8ead-e4f31bee3496] 2015-10-27 12:35:33,252 INFO [Drop FlowFiles for Connection 239dc171-e1d9-4782-8ead-e4f31bee3496] o.a.n.c.r.WriteAheadFlowFileRepository Repository updated to reflect that 10000 FlowFiles were swapped in to FlowFileQueue[id=239dc171-e1d9-4782-8ead-e4f31bee3496] 2015-10-27 12:35:33,678 INFO [Drop FlowFiles for Connection 239dc171-e1d9-4782-8ead-e4f31bee3496] o.a.n.c.r.WriteAheadFlowFileRepository Repository updated to reflect that 10000 FlowFiles were swapped in to FlowFileQueue[id=239dc171-e1d9-4782-8ead-e4f31bee3496] 2015-10-27 12:35:34,146 INFO [Drop FlowFiles for Connection 239dc171-e1d9-4782-8ead-e4f31bee3496] o.a.n.c.r.WriteAheadFlowFileRepository Repository updated to reflect that 10000 FlowFiles were swapped in to FlowFileQueue[id=239dc171-e1d9-4782-8ead-e4f31bee3496] 2015-10-27 12:35:34,587 INFO [Drop FlowFiles for Connection 239dc171-e1d9-4782-8ead-e4f31bee3496] o.a.n.c.r.WriteAheadFlowFileRepository Repository updated to reflect that 10000 FlowFiles were swapped in to FlowFileQueue[id=239dc171-e1d9-4782-8ead-e4f31bee3496] 2015-10-27 12:35:35,013 INFO [Drop FlowFiles for Connection 239dc171-e1d9-4782-8ead-e4f31bee3496] o.a.n.c.r.WriteAheadFlowFileRepository Repository updated to reflect that 10000 FlowFiles were swapped in to FlowFileQueue[id=239dc171-e1d9-4782-8ead-e4f31bee3496] 2015-10-27 12:35:35,431 INFO [Drop FlowFiles for Connection 239dc171-e1d9-4782-8ead-e4f31bee3496] o.a.n.c.r.WriteAheadFlowFileRepository Repository updated to reflect that 10000 FlowFiles were swapped in to FlowFileQueue[id=239dc171-e1d9-4782-8ead-e4f31bee3496] 2015-10-27 12:35:35,853 INFO [Drop FlowFiles for Connection 239dc171-e1d9-4782-8ead-e4f31bee3496] o.a.n.c.r.WriteAheadFlowFileRepository Repository updated to reflect that 10000 FlowFiles were swapped in to FlowFileQueue[id=239dc171-e1d9-4782-8ead-e4f31bee3496] {code} I had accidentally refreshed the page when trying to refresh stats of the flow, and then lost the ability to view the flow, only seeing the glowing drop. I tried stopping my instance via nifi.sh, but this seemed to fail and the background processes continued to run. Forcibly quitting, I was able to start NiFi again and let it run for a while. At startup, there were several errors concerning provenance, with several of the below exception: {code} 015-10-27 12:33:36,905 ERROR [main]

o.a.n.p.PersistentProvenanceRepository Failed to read Provenance Event File ./provenance_repository/43401826.prov.gz due to {} java.io.EOFException: Unexpected end of ZLIB input stream at java.util.zip.InflaterInputStream.fill(InflaterInputStream.java:240) ~[na:1.8.0_60] at java.util.zip.InflaterInputStream.read(InflaterInputStream.java:158) ~[na:1.8.0_60] at java.util.zip.GZIPInputStream.read(GZIPInputStream.java:117) ~[na:1.8.0_60] at java.io.BufferedInputStream.fill(BufferedInputStream.java:246) ~[na:1.8.0_60] at java.io.BufferedInputStream.read(BufferedInputStream.java:265) ~[na:1.8.0_60] at org.apache.nifi.stream.io.ByteCountingInputStream.read(ByteCountingInputStream.java:41) ~[nifi-utils-0.3.1-SNAPSHOT.jar:0.3.1-SNAPSHOT] at java.io.DataInputStream.readUnsignedShort(DataInputStream.java:337) ~[na:1.8.0_60] at java.io.DataInputStream.readUTF(DataInputStream.java:589) ~[na:1.8.0_60] at java.io.DataInputStream.readUTF(DataInputStream.java:564) ~[na:1.8.0_60] at org.apache.nifi.provenance.StandardRecordReader.<init>(StandardRecordReader.java:91) ~[nifi-persistent-provenance-repository-0.3.1-SNAPSHOT.jar:0.3.1-SNAPSHOT] at org.apache.nifi.provenance.serialization.RecordReaders.newRecordReader(RecordReaders.java:106) ~[nifi-persistent-provenance-repository-0.3.1-SNAPSHOT.jar:0.3.1-SNAPSHOT] at org.apache.nifi.provenance.PersistentProvenanceRepository.recover(PersistentProvenanceRepository.java:548) [nifi-persistent-provenance-repository-0.3.1-SNAPSHOT.jar:0.3.1-SNAPSHOT] at org.apache.nifi.provenance.PersistentProvenanceRepository.initialize(PersistentProvenanceRepository.java:219) [nifi-persistent-provenance-repository-0.3.1-SNAPSHOT.jar:0.3.1-SNAPSHOT] at org.apache.nifi.controller.FlowController.<init>(FlowController.java:406) [nifi-framework-core-0.3.1-SNAPSHOT.jar:0.3.1-SNAPSHOT] at org.apache.nifi.controller.FlowController.createStandaloneInstance(FlowController.java:349) [nifi-framework-core-0.3.1-SNAPSHOT.jar:0.3.1-SNAPSHOT] at org.apache.nifi.spring.FlowControllerFactoryBean.getObject(FlowControllerFactoryBean.java:63) [nifi-framework-core-0.3.1-SNAPSHOT.jar:0.3.1-SNAPSHOT] at org.springframework.beans.factory.support.FactoryBeanRegistrySupport.doGetObjectFromFactoryBean(FactoryBeanRegistrySupport.java:168) [spring-beans-4.1.6.RELEASE.jar:4.1.6.RELEASE] {code} I tried a few times to recreate this, but could not get the same result as above. Not sure if there is a way around it, but ultimately, the blocking on the operation and being able to access the UI, which could be confusing for other people trying to access a given instance. Is it feasible to heartbeat from the user issuing the request to the background processing and causing the termination to occur if I should navigate away? This wouldn't solve the issue of the blocking, but would at least prevent losing control of the process should I navigate away. Thoughts on any of the above?
label: code-design

26. **body:** [~aldrin] The messages about the flowfiles being swapped in is expected, as you had 2 million FlowFiles queued up. Since it's swapping in 10K at a time, you're talking about 200 of those log messages. This is all happening in the background, though. Not sure why it would cause your browser to stop responding. Is it possible that garbage collection was hitting your instance really hard? I imagine there's likely a good bit of garbage collection going on as it just created and swapped out 2 million FlowFiles and is now actively swapping them back in. The Provenance error, I believe, is entirely unrelated and was just the result of when you restarted NiFi. I created a ticket, NIFI-1076 to address that. Looking into it, it appears to be simply a poorly handled condition where we should log a warning or an info level message and move on. We certainly don't need a stack trace for that case.
label: code-design
27. The browser not responding I can consistently recreate. # Queue up many files on a connection # initialize an empty on that connection # refresh the browser (page level not just stats) # UI is blocked from loading until, seemingly, that emptying process completes. Interacting with the REST API while the UI is blocking, however, is responsive and immediate. Will attach a thread dump
28. [~aldrin] - not sure that I understand the statement "Interacting with the REST API while the UI is blocking, however, is responsive and immediate." Does that indicate that you are interacting with the REST API without using the UI? I.e., a curl command for instance?
29. Correct. Doing a `watch` on curl -s localhost:8080/nifi-api/controller/status | jq . continuously returns a new clientId every second, even while the UI hangs on the droplet. Watching that output, such as the below, for the queue to go to zero and then refreshing the UI, the canvas loads instantaneously
30. Digging in more, the initial page loads and returns the core HTML resources on a request via curl as well when requested. It seems like one of the JS calls is blocking on page load.
31. Loading a JS resource is blocking? Or some Ajax requests are taking a long time? We show the drop until we've received responses from a number of key end points (the flow, the users authorities, etc).
32. Sorry, poor wording. It seems that the requests are taking an indefinite amount of time. They do not successfully complete until the empty process has finished for the scenario laid out above.
33. [~markap14] Based off the thread dump that Aldrin provided it appears that getting the backpressure configuration for the connection that is being emptied is blocking while doing so. This will cause any request that happens to return those values in the response (GET group, GET/PUT connection) to block. This will also affect other users. So if Aldrin initiates a long-running empty request and then you attempt to navigate to that group or configure that connection it will block. I am not familiar with the threading in that class, but do we need to use a read/write lock on those configuration values? Possibly replace with Atomic references? [~aldrin] Great catch!
34. Commit edf238e004e45566374fcb520555cf4cc30ed8f7 in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=edf238e] NIFI-730: Do not require a Read Lock in order to obtain backpressure configuration values for FlowFileQueue's
35. [~aldrin] - That was a great catch! I pushed a fix for that. Seems to be working well.
36. Reviewing new updates
37. Overall code, contrib and build are all good. Tested both a single instance and clustered with various configurations on the queue. The single instance seemed to be fine regardless of configuration. There is some interesting behavior in terms of statistics where if I proceed to purge a queue from one view of the UI and have the UI open in another window, initiating another purge from that other window the second window's request will not start until the purge in the first UI completes and will not update any of the stats in that window until that process begins. I made a local cluster with a manager and two nodes. I create a flow consisting of a GenerateFlowFile connected to a non-running UpdateAttribute. The connection between them had a backpressure of 500k flowfiles. I did the same approach as laid out above with the associated backpressure and this resulted in both nodes getting disconnected with stacktraces as shown below: {code} 015-10-31 16:43:19,200 INFO [NiFi Web Server-123] org.apache.nifi.web.filter.RequestLogger Attempting request for (anonymous) GET http://localhost:8080/nifi-api/controller/proce ss-groups/81b0bf63-3127-4eec-830b-28d325d8ce6c/connections/9d12660c-71ad-4895-acae-d0045987b90b/drop-requests/447fbe61-f0ed-3487-b5df-14e96c20251d (source ip: 127.0.0.1) 2015-10-31 16:43:19,206 ERROR [NiFi Web Server-123] o.a.nifi.web.api.config.ThrowableMapper An unexpected error has occurred: java.lang.ArithmeticException: / by zero. Returning Internal Server Error response. java.lang.ArithmeticException: / by zero at org.apache.nifi.web.api.dto.DtoFactory.createDropRequestDTO(DtoFactory.java:332) ~[classes/:na] at org.apache.nifi.web.StandardNiFiServiceFacade.getFlowFileDropRequest(StandardNiFiServiceFacade.java:2107) ~[classes/:0.3.1-SNAPSHOT] at org.apache.nifi.web.StandardNiFiServiceFacade$$FastClassBySpringCGLIB$$358780e0.invoke(<generated>) ~[spring-core-4.1.6.RELEASE.jar:0.3.1-SNAPSHOT] at org.springframework.cglib.proxy.MethodProxy.invoke(MethodProxy.java:204) ~[spring-core-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invokeJoinpoint(CglibAopProxy.java:717) ~[spring-aop-

4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:157) ~[spring-aop-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.springframework.aop.aspectj.MethodInvocationProceedingJoinPoint.proceed(MethodInvocationProceedingJoinPoint.java:85) ~[spring-aop-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.apache.nifi.web.NiFiServiceFacadeLock.getLock(NiFiServiceFacadeLock.java:139) ~[classes/:0.3.1-SNAPSHOT] at sun.reflect.GeneratedMethodAccessor66.invoke(Unknown Source) ~[na:na] at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[na:1.8.0_60] at java.lang.reflect.Method.invoke(Method.java:497) ~[na:1.8.0_60] at org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethodWithGivenArgs(AbstractAspectJAdvice.java:621) ~[spring-aop-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.springframework.aop.aspectj.AbstractAspectJAdvice.invokeAdviceMethod(AbstractAspectJAdvice.java:610) ~[spring-aop-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.springframework.aop.aspectj.AspectJAroundAdvice.invoke(AspectJAroundAdvice.java:68) ~[spring-aop-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:179) ~[spring-aop-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(ExposeInvocationInterceptor.java:92) ~[spring-aop-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:179) ~[spring-aop-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.springframework.aop.framework.CglibAopProxy$DynamicAdvisedInterceptor.intercept(CglibAopProxy.java:653) ~[spring-aop-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.apache.nifi.web.StandardNiFiServiceFacade$$EnhancerBySpringCGLIB$$d357524c.getFlowFileDropRequest(<generated>) ~[spring-core-4.1.6.RELEASE.jar:0.3.1-SNAPSHOT] at org.apache.nifi.web.api.ConnectionResource.getDropRequest(ConnectionResource.java:1030) ~[classes/:na] at org.apache.nifi.web.api.ConnectionResource$$FastClassBySpringCGLIB$$b8cd4334.invoke(<generated>) ~[spring-core-4.1.6.RELEASE.jar:na] at org.springframework.cglib.proxy.MethodProxy.invoke(MethodProxy.java:204) ~[spring-core-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.invokeJoinpoint(CglibAopProxy.java:717) ~[spring-aop-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:157) ~[spring-aop-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.springframework.security.access.intercept.aopalliance.MethodSecurityInterceptor.invoke(MethodSecurityInterceptor.java:64) ~[spring-security-core-3.2.7.RELEASE.jar: 3.2.7.RELEASE] at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:179) ~[spring-aop-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.springframework.aop.framework.CglibAopProxy$DynamicAdvisedInterceptor.intercept(CglibAopProxy.java:653) ~[spring-aop-4.1.6.RELEASE.jar:4.1.6.RELEASE] at org.apache.nifi.web.api.ConnectionResource$$EnhancerBySpringCGLIB$$155c0ae4.getDropRequest(<generated>) ~[spring-core-4.1.6.RELEASE.jar:na] at sun.reflect.GeneratedMethodAccessor97.invoke(Unknown Source) ~[na:na] at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[na:1.8.0_60] at java.lang.reflect.Method.invoke(Method.java:497) ~[na:1.8.0_60] at com.sun.jersey.spi.container.JavaMethodInvokerFactory$1.invoke(JavaMethodInvokerFactory.java:60) ~[jersey-server-1.19.jar:1.19] {code}

38. This was from the nifi-user.log on each of the nodes with corresponding disconnect requests in the manager's nifi-app.log
39. Thanks [~aldrin]. Will address the exception and resubmit. I'm pretty sure the delay in purging while outstanding requests are processing is by design. The second UI should have indicated that it was waiting. Let me know if this was not the case. Thanks!
40. Right you are about the notification. Didn't even notice the different text. Good thinking :D
41. Commit f5727cfb0fe73b7226fe9eca96fd594f8d423ffa in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=f5727cf] NIFI-730: Ensure that we always populate queue counts when initiating a Drop FlowFile request
42. Commit 5a04021dd7a61cc7c76b24405288713116bcd682 in nifi's branch refs/heads/NIFI-730 from [~mcgilman] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=5a04021] NIFI-730: - Updating logic now that original is guaranteed to be non null. - Always reporting 100% once the drop request has completed.
43. [~aldrin] A fix has been pushed for the exception you were seeing. Thanks!
44. Reviewing
45. Build, contrib, and code all look good and the additions seem to close out any quirks uncovered throughout the process. Nice work guys. +1
46. Commit e4cebba3c7868010c2b9fd994b850f40b81aa044 in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=e4cebba] Merge branch 'master' into NIFI-730
47. Commit dbf0c7893fef964bfbb3a4c039c756396587ce12 in nifi's branch refs/heads/NIFI-730 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=dbf0c78] NIFI-730: Completed merging of master
48. Commit edf238e004e45566374fcb520555cf4cc30ed8f7 in nifi's branch refs/heads/master from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=edf238e] NIFI-730: Do not require a Read Lock in order to obtain backpressure configuration values for FlowFileQueue's
49. Commit f5727cfb0fe73b7226fe9eca96fd594f8d423ffa in nifi's branch refs/heads/master from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=f5727cf] NIFI-730: Ensure that we always populate queue counts when initiating a Drop FlowFile request
50. Commit 5a04021dd7a61cc7c76b24405288713116bcd682 in nifi's branch refs/heads/master from [~mcgilman] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=5a04021] NIFI-730: - Updating logic now that original is guaranteed to be non null. - Always reporting 100% once the drop request has completed.
51. Commit e4cebba3c7868010c2b9fd994b850f40b81aa044 in nifi's branch refs/heads/master from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=e4cebba] Merge branch 'master' into NIFI-730
52. Commit dbf0c7893fef964bfbb3a4c039c756396587ce12 in nifi's branch refs/heads/master from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=dbf0c78] NIFI-730: Completed merging of master
53. Commit e4cebba3c7868010c2b9fd994b850f40b81aa044 in nifi's branch refs/heads/NIFI-274 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=e4cebba] Merge branch 'master' into NIFI-730
54. Commit dbf0c7893fef964bfbb3a4c039c756396587ce12 in nifi's branch refs/heads/NIFI-274 from [~markap14] [https://git-wip-us.apache.org/repos/asf?p=nifi.git;h=dbf0c78] NIFI-730: Completed merging of master