Aeron Initial Commercial Release

Test Plan

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author(s) |
| 04/06/15 | 0.1 | Draft of Aeron Test Plan | M.Tiddens,A.Gladney |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[Purpose of Document](#h.ngtox7hjt99y)

[Scope](#h.j2h78vt3uhqg)

[Strategy](#h.rgknft3rosvx)

[Test Scenarios](#h.tusjr2akmko6)

[Resources](#h.f0nrfyiycg7p)

### Purpose of Document

This document describes the plan for testing Aeron, a new high-performance open source message transport library. The goal is to ensure the Aeron product is ready to be released commercially.

### Scope

This test plan documents the high level functional, stress, and performance test cases which will be executed during the test cycle. It also defines the test methodologies and evaluation criteria that must be met in order to ensure the accurate, repeatable, and uniform testing of the aeron transport . This test plan does not provide specific test equipment configurations or detailed test instructions by which to execute the tests.

### Strategy

In an effort to validate Aeron three levels/methodologies of testing will be used: Functional, Stress, and Performance.

The functional tests will focus on verification of what would be considered the correct behavior under normal circumstances. The PublisherTool and SubscriberTool test applications will be the primary applications used for this level of testing.

The stress tests will put an emphasis on availability and error handling under a certain load. It will help to determine if the system will perform sufficiently under certain conditions. The Thwacker, Publisher, and Subscriber test tools will be used for this level of testing. They will be used in certain configurations that will be pushing certain operating limits and verifying correct behavior.

The performance tests will demonstrate that the system meets the performance criteria. The Ping/Pong and Throughputency Pub/Sub Tools will be used for this level of testing. The initial focus will be to capture the current Aeron baseline performance in a select few scenarios that we will be using to compare to future releases.

### 

Where appropriate the test scenarios will be automated. The automated test tool will use the pairwise testing technique/method (i.e. test multiple possible discrete combinations of input parameters allowing the framework to run a smaller number of tests while ensuring full coverage). Tests will be executed manually during the automation development phase.

### 

### 

### 

### 

### 

### Test Scenarios

Special Note: Assume Media Drive(s) are actively running for each test unless otherwise stated.

**id0000: Basic Test**

*Summary:* A publisher sends messages to a single subscriber.

A subscriber is started and sends a subscription using a particular stream id.

A publisher is started and receives a subscription for the stream id of interest, the publisher then sends data.

*Expected Results:* The subscriber receives all messages sent by the publisher. Verify no memory leaks are reported.

*idNOTES*: Repeat test scenario using the various config options documented in the [TestPLanning spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) for the ‘basic test’.

**id0005: Basic Test**

*Summary:* A subscriber is started after the publisher.

Start a single publisher. Wait a few seconds and start a single subscriber using the stream ID of interest.

*Expected Results:* Before the subscriber is started, the senders offer will fail. After the subscriber is known, the sender will be allowed to send and the subscriber should receive all messages

*NOTES*: Repeat test scenario using the various config options documented in the [TestPLanning spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) for the ‘BasicTest’.

**id0010: Loss Test**

*Summary:* A subscriber is started, and a publisher attempts to send a SETUP message/FRAME to the subscriber. Drop the message.

*Expected Results*: ?? Time out occurs?

*NOTES*: Repeat test scenario using the various config options documented in the [TestPLanning spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) for the ‘LossTest’.

**i**

**id0025: Loss Test**

*Summary:* Subscriber disconnects and reconnects when unrecoverable loss occurs.

While a publisher is actively sending messages to subscriber, the driver associated with the subscriber will periodically lose a percentage of messages as documented in the [TestPlanning Spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) under the ‘LossTest’ config options.

The subscriber can not keep up or recover (NAKs will NOT be sent for the missed messages).

*Expected Results:* The subscriber which is behind will disconnect and reconnect. It should start receiving messages from the live stream.

*NOTE:* This scenario will be re-executed using the various config options documented in the [TestPlanning SpreadSheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) in the ‘LossTest’ section.

**id0030: Loss Test**

*Summary:* Subscriber disconnects and reconnects when unrecoverable loss occurs.

While a publisher is actively sending messages to a multiple subscribers, the driver associated with one of the subscribers will periodically lose a percentage of messages as documented in the [TestPlanning Spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) under the ‘LossTest’ config options.

The subscriber can not keep up or recover (NAKs will NOT be sent for the missed messages).

*Expected Results:* The subscriber which is behind will disconnect and reconnect. It should start receiving messages from the live stream. Verify all other subscribers are NOT affected.

*NOTE:* This scenario will be re-executed using the various config options documented in the [TestPlanning SpreadSheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) in the ‘LossTest’ section.

**id0035: SlowRcvTest**

*Summary:* A publisher sends messages to a slow subscriber on a different machine.

A fast publisher on machine A is sending messages to the a subscriber on machine B.

The publisher is sending too fast for the subscriber (i.e the rate of consumption of the subscriber is slower than the rate of publication of the publisher).

*Expected Results:* The data sent by the publisher exceeds the initial term window length (this is the window between the drivers?). The rate of publication will decrease until the subscriber is able to catch up.

*NOTES:* This scenario will be re-executed using the various config options documented in the [TestPlanning SpreadSheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) in the ‘SlowRcvTest’ section.

**id0040: SlowRcvTest**

*Summary:* A publisher sends messages to a slow subscriber on the same machine.

A fast publisher is sending messages to the a subscriber on the same machine.

The publisher is sending too fast for the subscriber (i.e the rate of consumption of the subscriber is slower than the rate of publication of the publisher).

*Expected Results:* The data sent by the publisher exceeds the publisher term window length. Will the publisher stop sending until the subscriber catches up or will the rate of publication decrease until the subscriber catches up

*NOTES:* This scenario will be re-executed using the various config options documented in the [TestPlanning SpreadSheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) in the ‘SlowRcvTest’ section.

**id0045: SlowMRcvTest**

*Summary:* A publisher sends messages to multiple subscribers (at least 3) on different machines. The publisher is sending too fast for one of the subscribers (i.e. the rate of consumption of the subscriber is slower than the rate of publication of the publisher).

*Expected Results:* The data sent by the publisher exceeds the initial term window length (this is the window between the drivers?). (How can this be verified)? The publisher and other subscribers are not affected by the slow subscriber ( the publisher will continue to publish messages at the same rate and the subscribers will continue to consume messages at the same rate.)

Will the slow subscriber report unrecoverable loss?. The slow subscriber will disconnect and reconnect to the driver. It should start receiving messages from the live stream.

*NOTES:* This scenario will be re-executed using the various config options documented in the [TestPlanning SpreadSheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) in the ‘SlowRcvTest’ section.

**id0050: SlowMRcvTest**

*Summary:* A publisher sends messages to multiple subscribers on the same machine.

A fast publisher is sending messages to multiple subscribers on the same machine. The publisher is sending too fast for one of subscribers (i.e the rate of consumption of the subscriber is slower than the rate of publication of the publisher). The other subscribers are able to keep up with the incoming message rate.

*Expected Results:* The data sent by the publisher exceeds the subscriber term window length. The slow subscriber will disconnect and reconnect to the driver. It should start receiving messages from the live stream. The other subscribers should not be affected.

*NOTES:* This scenario will be re-executed using the various config options documented in the [TestPlanning SpreadSheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) in the ‘SlowRcvTest’ section.

**id0055: KillComponentTest**

*Summary:* The media driver fails while the publisher is actively sending messages to the

subscriber. Start the media driver, the subscriber and the publisher all on the same machine. Allow the publisher to send a few messages before causing a media driver failure using one of the methods described in the [TestPlanning Spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) under the ‘KillComponentTest’ Config options.

*Expected Results:* If media driver is killed via Ctrl-C, verify clean shutdown (files are removed).

Restart driver, verify clients can connect and communication resumes.

If driver is suspended, verify the publisher is no longer sending messages to the subscriber.

Resume the driver, verify communication resumes

*NOTES:* This scenario will be re-executed using the various config options documented in the [TestPlanning SpreadSheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) in the ‘KillComponentTest’ section.

**id0060: KillComponentTest**

*Summary:* The media driver fails while the publisher is actively sending messages to the

subscriber. Enable config option on media driver to retain files in the event of restart/failure. Start the media driver, the subscriber and the publisher all on the same machine. Allow the publisher to send a few messages before causing a media driver failure using one of the methods described in the [TestPlanning Spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) under the ‘KillComponentTest’ Config options.

*Expected Results:* If media driver is killed via Ctrl-C, verify communication is interrupted between the clients. Verify media driver files are NOT removed. Restart driver, verify clients can connect, communication resumes, and driver files are accessible.

If driver is suspended, verify the publisher is no longer sending messages to the subscriber.

Resume the driver, verify communication resumes

*NOTES:* This scenario will be re-executed using the various config options documented in the [TestPlanning SpreadSheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) in the ‘KillComponentTest’ section.

**id0065: KillComponentTest**

*Summary:* The media driver fails while the publisher is actively sending messages to the

subscriber. Start a media driver and subscriber on machine A . Start another media driver and publisher on machine B. Allow the publisher to send a few messages to the subscriber before causing media driver on machine B to fail using one of the methods described in the [TestPlanning Spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) under the ‘KillComponentTest’ Config options.

*Expected Results:*

*NOTES:* This scenario will be re-executed using the various config options documented in the [TestPlanning SpreadSheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) in the ‘KillComponentTest’ section

**id0070: KillComponentTest**

*Summary:* The media driver fails while the subscriber is actively receiving messages from the publisher. Start a media driver and subscriber. Start another media driver and publisher. (All processes are running on the same machine). Allow the publisher to send a few messages to the subscriber before causing one of the media drivers to fail using one of the methods described in the [TestPlanning Spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) under the ‘KillComponentTest’ Config options. ( does this make sense? is this supported? more than one media driver on the same machine?

*Expected Results:*

*NOTES:* This scenario will be re-executed using the various config options documented in the [TestPlanning SpreadSheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) in the ‘KillComponentTest’ section.

**id0075: KillComponentTest**

*Summary:* The publisher fails while it is actively sending messages to the subscriber.

Start a subscriber and publisher. Allow the publisher to send a few messages before causing a publisher failure using one of the methods described in the [TestPlanning Spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) under the ‘KillComponentTest’ Config options.

*Expected Results:*

*NOTES:* This scenario will be re-executed using the various config options documented in the [TestPlanning SpreadSheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) in the ‘KillComponentTest’ section.

**id0080: KillComponentTest**

*Summary:* One of the active publishers is killed while sending to a subscriber.

Publisher 1 is actively sending messages to a subscriber. Start publisher 2 using the same

stream id. Communication between publisher 1 and the subscriber is uninterrupted.

The publishers will be using different session ids, the subscriber will receive messages from both. Terminate publisher 1.

*Expected Results:* The subscriber will continue to receive messages from publisher 2.

*NOTES*: Repeat test scenario using the various config options documented in the [TestPLanning spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) for the ‘KillComponentTest’.

**id0085: KillComponentTest**

*Summary:* The subscriber fails while it is receiving messages from the publisher. Start a subscriber and publisher. Allow the publisher to send a few messages before causing a subscriber failure using one of the methods described in the [TestPlanning Spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) under the ‘KillComponentTest’ Config options.

*Expected Results:*

*NOTES:* This scenario will be re-executed using the various config options documented in the [TestPlanning SpreadSheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) in the ‘KillComponentTest’ section.

**id0090: KillOneSubTest**

*Summary:* One of the multiple active subscribers fails while receiving messages from a single publisher. Start multiple subscribers (at least 3) and a single publisher. Allow the publisher to send a few messages and verify all subscribers are receiving messages. Cause one of the subscribers to fail using one of the methods described in the [TestPlanning Spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) under the ‘KillOneSubTest’ Config options.

*Expected Results:*

*NOTES:* This scenario will be re-executed using the various config options documented in the [TestPlanning SpreadSheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) in the ‘KillComponentTest’ section.

**id0090: Msrc Test**

*Summary:* Multiple publishers send messages to a single subscriber.

A subscriber is started and sends a subscription using a particular stream id.

Three publishers are started using the same session id, stream id, term id, and term offset.

*Expected Results:* (Will the subscriber receive duplicate messages or will the dups be dropped).

(Publishers all on the same machine?)

*NOTES*: Repeat test scenario using the various config options documented in the [TestPLanning spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) for the ‘MSrcTest’.

**id0100: Msrc Test**

*Summary:* Late publisher (A second publisher joins an active stream)

Publisher 1 is actively sending messages to a subscriber. Start publisher 2 using the same

stream id. Communication between publisher 1 and the subscriber is uninterrupted.

*Expected Results:* The publishers will be using different session ids, the subscriber will receive messages from both publishers.

*NOTES*: Repeat test scenario using the various config options documented in the [TestPLanning spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) for the ‘MsrcTest’.

**id0105: Msrc Test**

*Summary:* Late publisher (A new publisher joins an active session)

Two publishers are actively sending messages to a subscriber. Start a third publisher 2 using the same session id but a different stream id.

*Expected Results:* Verify communication between the initial publishers and the subscriber is not affected. The third publisher is blocked from sending because there is not an active subscriber corresponding to the its stream id

*NOTES*: Repeat test scenario using the various config options documented in the [TestPLanning spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) for the ‘MsrcTest’.

i**d0110: Mrcv Test**

*Summary*: Multiple subscribers receive msgs from a single publisher.

Start at least 3 subscribers . Start a single publisher which will send messages using the stream id of interest to the aforementioned subscribers.

*Expected Results:*  All subscribers will receive all messages sent by the publisher. No faults or errors are reported.

*NOTES:* Repeat test scenario using the various config options documented in the [TestPLanning spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) for the ‘MRcvTest’.

**id0115: MrcvTest**

*Summary:* A subscriber subscribes to an existing stream.

A publisher is actively sending messages to a subscriber. Start a new subscriber using the same stream id.

*Expected Results:* The communication between the initial publisher and subscriber is uninterrupted (i.e. no observable effects like reduced throughput).

The second subscriber will start receiving messages from the live stream.

*NOTES:* Currently this scenario is only supported for multicast

**id0120: Mrcv Test**

*Summary:* Multiple subscribers are started after the publisher.

Start a publisher user a particular stream id. (What happens? Does the publisher send the first frame of messages and then wait for a subscriber to show interest? Are the messages saved in a local buffer?). After a few seconds, start a subscriber using the same stream id.

Next start a second subscriber using the same stream id.

*Expected Results:* What happens will subscriber 1 receive all the messages? Will subscriber 2 only receive messages from the liive stream?

*NOTES:* Currently this scenario is only supported for multicast

**id0125: MsrcMrcvTest**

*Summary:* Multiple publishers send messages to multiple subscribers.

Multiple subscribers(at least 5) are started and send subscriptions using particular stream ids.

Multiple publishers are started and receive subscriptions for the stream ids of interest, the publishers then sends data.

*Expected Results:* The subscribers receive all messages sent by the publishers.

*NOTES*: Repeat test scenario using the various config options documented in the [TestPLanning spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) for the ‘MSrcMRcvTest’.

**id0130: Performance**

*Summary:* Throughput: Using small messages (32-bytes), the message delivery rate is 2 million messages per sec.

A single publisher sends messages to a single subscriber at a rate of 2 million messages per sec.

*Results:* Verify no faults or errors are reported during execution of test.

*NOTES:* This scenario will be using both unicast and multicast, as well as with the applications on different machines.

**id0135: Performance**

*Summary:* Message delivery latency: Measure the message delivery latency for 2 million messages per sec. Using a single publisher to send messages to a single subscriber at a rate of 2 million messages per sec, measure the message delivery latency.

*Expected Results:* What is the target? (Expected result for apps on the same machine and across machines)?

*NOTES:* This scenario will be executed using both unicast and multicast.

**id0140: Stats**

*Summary:* Scenarios are executed which will increment the following stats:

Bytes sent

Bytes Received

NAKs sent

Retransmits sent

*Expected Results:* Verify the aforementioned stats are incremented appropriately.

*NOTE:* *Manual Test Execution Procedure for NAKs out and RX sent stats*

Tools: MediaDriverTool, StatsDriver, PublisherTool, and SubscriberTool from the aeron.tools directory.

The MediaDriverTool, and StatsDriver always ran on Linux.

1) Start Subscriber on Linux and Publisher on machine B and observed bytes in value in StatsDriver incrementing.

2) Start Publisher on Linux and Subscriber on machine B and observed bytes out value increasing.

3) Restart the MediaDriverTool with -Daeron.debug.data.loss.rate=0.05 for 5% data loss rate, and restart the StatsDriver. Then start a Subscriber on Linux and a Publisher on machine B

Results:

Observe NAKs Out stat increasing, and RX also increasing.

**id0145: Stress**

*Summary:* Media driver failure occurs while multiple publishers are sending messages to multiple subscribers.

While 50 publishers are actively sending messages to 50 subscribers at a rate of 30,000 messages per sec per pub/sub pair, cause a media driver failure using one of the methods described in the [TestPlanning Spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) under the ‘KillComponentTest’ Config options.

*Expected Results:* During the driver failure, communication between the publishers and subscribers will cease. The clients should not fault or exit. Allow the driver to recover. Verify the the publishers and subscribers can reconnect to the driver and communication can resume.

**id0150: Stress**

*Summary:* Publisher failure occurs while multiple publishers are sending messages to multiple subscribers.

While 50 publishers are actively sending messages to 50 subscribers at a rate of 30,000 messages per sec per pub/sub pair, cause one of the publishers to fail using one of the methods described in the [TestPlanning Spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) under the ‘KillComponentTest’ Config options.

*Expected Results:* During the publisher failure, communication between the other publishers and subscribers should NOT be affected. Allow the publisher to recover. During the recover of the publisher under test, communication should NOT be affected. Verify the publisher can reconnect to the driver and successfully send messages to a subscriber.

**id0155: Stress**

*Summary:* Subscriber failure occurs while multiple publishers are sending messages to multiple subscribers.

While 50 publishers are actively sending messages to 50 subscribers at a rate of 30,000 messages per sec per pub/sub pair, cause one of the subscribers to fail using one of the methods described in the [TestPlanning Spreadsheet](https://docs.google.com/a/kaazing.com/spreadsheets/d/15ZztzZnF9kQgTUB8fsx7AuxmBhmR4GJ49XdhqfZuLCM/edit#gid=0) under the ‘KillComponentTest’ Config options.

*Expected Results:* During the publisher failure, communication between the other publishers and subscribers should NOT be affected. Allow the subscriber to recover. During the recover of the subscriber under test, communication should NOT be affected. Verify the publisher can reconnect to the driver and successfully send messages to a subscriber.

**id0145: Thwacker test**

*Summary:* Dynamic Object creation and deletion during execution. The test app will run for a given amount of time and during this period it will create and delete Publications and Subscriptions as fast as possible while messages are being sent. This is to focus on exposing race conditions or deadlocks in the Aeron code.

*Expected Results:* The test application should run to completion for approximately the same time it was set to run for. (i.e. it shouldn’t run for 5 minutes if you set it to run for 30 seconds. If it takes 32 seconds, we would deem it as a pass.) There shouldn’t be any exceptions or crashes and there should be some messages actually delivered (verified via logged message counts).

*NOTES:* This scenario will be executed using unicast and multicast, bursty and continuous, fan-in and one-to-one.

### Resources

The Aeron Wiki: <https://github.com/real-logic/Aeron/wiki>