Configuring a TIBCO Enterprise Message Service™ Fault Tolerant Environment On Amazon Web Services

This document provides the steps for configuring and testing TIBCO Enterprise Message Service Fault Tolerance in a Linux operating environment on AWS utilizing the Elastic File System (EFS)

<table>
<thead>
<tr>
<th>Version .1</th>
<th>Initial Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version .2</td>
<td>Added information on support for TLS Tunneling on Amazon Linux2</td>
</tr>
</tbody>
</table>
Copyright Notice
COPYRIGHT© 2018 TIBCO Software Inc. This document is unpublished and the foregoing notice is affixed to protect TIBCO Software Inc. in the event of inadvertent publication. All rights reserved. No part of this document may be reproduced in any form, including photocopying or transmission electronically to any computer, without prior written consent of TIBCO Software Inc. The information contained in this document is confidential and proprietary to TIBCO Software Inc. and may not be used or disclosed except as expressly authorized in writing by TIBCO Software Inc. Copyright protection includes material generated from our software programs displayed on the screen, such as icons, screen displays, and the like.

Trademarks
All brand and product names are trademarks or registered trademarks of their respective holders and are hereby acknowledged. Technologies described herein are either covered by existing patents or patent applications are in progress.

Content Warranty
The information in this document is subject to change without notice. THIS DOCUMENT IS PROVIDED "AS IS" AND TIBCO MAKES NO WARRANTY, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO ALL WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TIBCO Software Inc. shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Export
This document and related technical data, are subject to U.S. export control laws, including without limitation the U.S. Export Administration Act and its associated regulations, and may be subject to export or import regulations of other countries. You agree not to export or re-export this document in any form in violation of the applicable export or import laws of the United States or any foreign jurisdiction.

For more information, please contact:

TIBCO Software Inc.
3303 Hillview Avenue
Palo Alto, CA 94304
USA
# Table of Contents

Configuring a TIBCO Enterprise Message Service™ Fault Tolerant Environment On Amazon Web Services

1 Overview ........................................................................................................................................... 5
  1.1 Document Purpose ......................................................................................................................... 5
  1.2 Linux Kernel Version Support .......................................................................................................... 5
  1.3 EMS Version Support ..................................................................................................................... 5
  1.4 Assumptions ................................................................................................................................... 5

2 AWS Setup for Linux .......................................................................................................................... 6
  2.1 Creating a new EC2 Instance .......................................................................................................... 6
  2.2 Setting Up the Elastic File System (EFS) ......................................................................................... 8
  2.3 Setting up the Red Hat EC2 Instance for EMS F/T ....................................................................... 9
    2.3.1 Additional Software Installation ............................................................................................. 9
    2.3.2 Setting up the NFS mount for the EFS Storage ..................................................................... 9
    2.3.3 Setting up the EFS/TLS mount for the EFS Storage ............................................................... 11

3 Enterprise Message Service Installation and Configuration .............................................................. 12
  3.1 EMS Installation ............................................................................................................................ 12
  3.2 EMS Configuration .......................................................................................................................... 12
    3.2.1 Stores.conf ............................................................................................................................... 12
    3.2.2 Factories.conf ........................................................................................................................... 13
    3.2.3 Tibemsd.conf ............................................................................................................................ 13
    3.2.4 Starting the EMS Instances ....................................................................................................... 14

4 Testing EMS Fault Tolerance on AWS with EFS ................................................................................ 16
  4.1 EMS Client App Setup .................................................................................................................... 16
  4.2 Performing the EMS Fault Tolerant Test Cases ......................................................................... 17
    4.2.1 EMS Process Failure Test ........................................................................................................ 17
    4.2.2 Network Failure Test on Linux ................................................................................................. 20
    4.2.3 System Failure Test ................................................................................................................ 23
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Amazon Web Services Console</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Configure EC2 Instance Details</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>Security Group Inputs</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>Running AWS EC2 Instances</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>Configure File System Access</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>Newly Created EFS File System</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>Fstab Example</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>Df Example</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>Mount Example</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>Example of /etc/fstab with TLS</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>Stores.conf Example</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>Connection Factory Settings</td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td>Linux EMS Startup</td>
<td>15</td>
</tr>
<tr>
<td>14</td>
<td>Create the Sync queue</td>
<td>17</td>
</tr>
<tr>
<td>15</td>
<td>Running TIBJMSMsgProducerPerf</td>
<td>18</td>
</tr>
<tr>
<td>16</td>
<td>Standby EMS Becoming Active on TIBEMS2</td>
<td>19</td>
</tr>
<tr>
<td>17</td>
<td>Purge the Sync queue from TIBEMSADMIN64</td>
<td>20</td>
</tr>
<tr>
<td>18</td>
<td>Drop_nfs.sh Script</td>
<td>21</td>
</tr>
<tr>
<td>19</td>
<td>Running drop_nfs.sh</td>
<td>22</td>
</tr>
<tr>
<td>20</td>
<td>Disk Write Error on TIBEMS1</td>
<td>22</td>
</tr>
<tr>
<td>21</td>
<td>AWS Console on the EC2 Dashboard</td>
<td>23</td>
</tr>
<tr>
<td>22</td>
<td>Stand-by EMS instance recovering from system failure of the the primary</td>
<td>24</td>
</tr>
</tbody>
</table>
1 Overview

1.1 Document Purpose
The purpose of the document is to provide a guide to install, configure, and run TIBCO Enterprise Message Service™ (EMS) in a fault-tolerant configuration on Amazon Web Service (AWS) utilizing AWS’s Elastic File System (EFS) for shared storage. In addition, the document will provide the steps and expected results for testing EMS F/T on AWS with EFS.

The document will outline:

- Setting up Red Hat Linux EC2 instances on AWS
- Setting up EFS for the shared file system in AWS
- Setting up the NFS4 mount on the Red Hat Linux EC2 instance
- Setting up the Amazon Linux2 EC2 instance to support TLS tunneling when mounting the EFS file system
- Installing and configuring EMS for F/T
- Tuning EMS for AWS/EFS
- Running tests for:
  - EMS process failure
  - Network failure between the EC2 instance running EMS and the shared storage
  - Accidental reboot of the EC2 instance from the AWS Dashboard

1.2 Linux Kernel Version Support
This document covers the installation and configuration of EMS on Red Hat Linux 7.4, which is based on Linux kernel 3.10.0-692.2.2.

However, other Linux kernels can be used. Linux AMI distributions on AWS utilizing kernels between 4.9.0 and 4.9.49 inclusive, have an NFS defect. Linux kernels 4.9.50 and above, should have the NFS defect corrected, and can be used. This includes Amazon Linux (kernel 4.9.51), which was tested with EMS, and is supported.

If not using Red Hat Linux (kernel 3.10.0) or Amazon Linux (kernel 4.9.51) on AWS for EMS, ensure that a supported Linux kernel is used.

Note: Amazon recently released their Amazon Linux2 version of Linux. It is based on the Linux 4.14.51 kernel. This kernel offers many new features, with a support life of five years. This new Linux2 kernel will support TLS tunneling over NFS4 to support EMS data encryption end to end on AWS. Modification to the EMS configuration and to the EFS are required to support this new capability. The modifications necessary are provided in this document, where noted.

1.3 EMS Version Support
EMS version 8.4 or later must be used for EMS server installations running on AWS environments. Any available EMS hot fixes must also be applied.

1.4 Assumptions
The reader of this document is familiar with the following concepts:

- The use of Amazon Web Services, and the AWS Console
- TIBCO EMS installation and configuration
- Red Hat Linux configuration
- NFSv4

Document only provides information for Red Hat Linux, except for setting up TLS tunneling on Amazon Linux. Other Linux kernels should be similar.
The following steps will outline setting up the EC2 instances in the AWS console. Red Hat Linux 7.4 was used for the EC2 instances. Other operating systems or versions are not covered. For the following examples, the EU West (Ireland) zone was used. Note: Ensure that the Elastic File System (EFS), is available in the AWS zone used.

2.1 Creating a new EC2 Instance

- Login into the AWS console.
- To create a new AWS EC2 Red Hat Linux instance, use the following:
  
  - In the AWS console, select Services, EC2, and then Launch Instance to create a new Amazon EC2 instance. Three EC2 instances are needed: two for EMS F/T and one for the client application.
  
  - Select Red Hat Enterprise Linux, then 7.4 (or most current version).
  - Select the appropriate size instance (vCPUs and Memory) for the environment
  - Click on Next. Configure Instance Details
  - Change number of instance to 3. Use the default VPC, and defaults for the other settings
Figure 2 – Configure EC2 Instance Details

- Click Next: Storage
- Configure the Volume Type and the Size in GB. These will depend on what else is installed other than EMS. If just EMS, 20GB is sufficient.
- Click on Next: Add Tags, and then Next: Configure Security Group
- Either use a new security group, or an existing security group. Ensure the security group used has rules to support SSH, TCP, NFS, and ICMP-IPv4 at the minimum. Must consider Source, Port Range, Protocol, and Type.

Figure 3 – Security Group Inputs

- Click on Review and Launch
2.2 Setting Up the Elastic File System (EFS)

To setup the Elastic File System to mount on your Linux EC2 instance, use the following:

- From the AWS console, click on Services
- Under Storage, click on EFS
- Click on Create file system
  - Use the appropriate VPC. This must be the same VPC used for the EC2 Linux instances
  - Ensure the same security group is used as the EC2 Linux instances, and that the security group has the necessary rules required to support TCP inbound/outbound to/from EFS.
  - Ensure all Availability Zones are selected. Then click Next Step

![Create file system](image)

**Figure 5 – Configure File System Access**

- Under Configure optional settings
  - Choose performance mode; General Purpose or MAX I/O. No measureable performance gains were seen with max i/o over GP.
• Click on Enable encryption, if desired. Note: There will be read/write performance degradation if selected. **Note:** If using TLS, select Enable.
• Click on Next Step
  o Review and Create File System
  o Wait for a “Successful” creation

• Once the new EFS File System is created:
  o Should still be on the File systems page in the AWS console
    • Refresh the page
    • Select the newly created file system. Note the DNS name and that the “Life cycle state” is “Available” for all zones.
    • Verify the correct VPC and Security Group were used
    • Review the Amazon EC2 mount instructions.

[Image of newly created EFS file system]

**Figure 6 – Newly Created EFS File System**

### 2.3 Setting up the Red Hat EC2 Instance for EMS F/T

Use the following to setup two of the newly created Red Hat instances for EMS fault tolerance. All steps must be completed on both of the instances.

#### 2.3.1 Additional Software Installation

• Login to the the EC2 instance using SSH. Use the instructions provided in Connect to Your Instance from the previous section
• Update the EC2 instance to the latest RH version, for Java, the nfs-utils, and unzip. **Note:** This step must be performed on all three EC2 instances.
  o sudo yum update
  o sudo yum install nfs-utils java-devel unzip

  **Note:** If using an Amazon Linux2 EC2 instance, also install amazon-efs-utils. IE: sudo yum install amazon-efs-utils

#### 2.3.2 Setting up the NFS mount for the EFS Storage

The EFS file system is mounted on the Linux EC2 instances using standard NFS4 mount commands.
• Create a new mount point on the Linux EC2 instance.
  o `mkdir /home/ec2-user/efs`

• Mount the new EFS file system with NFS. Note: The options used in the NFS4 mount are required! Failure to use the listed options may result in message loss/corruption, or EMS not performing at an acceptable level:
  o `sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,soft,timeo=100,retrans=2,actimeo=1 <DNS Name for EFS file system>:/home/ec2-user/efs`

  **Note:** The `<DNS Name for the EFS File system>` is the DNS name shown in the file systems page shown previously in the AWS console. It will be similar to `fs-abcd1234.efs.eu-west-1.amazonaws.com`

  **Note:** Though `actimeo` is not required, it is recommended. The `actimeo=1` option provides for better asynchronous persisted message write performance

• The mount can also be added to `/etc/fstab` to make it permanent. The following is an example of the `/etc/fstab` with the NFS4 mount.

```
[user@ip-172-31-21-60 ~]$ more /etc/fstab

# /etc/fstab
# Created by amoonda on Tue Jul 11 15:57:39 2017
# Accessible filesystems, by reference, are maintained under '/dev/disk'
# See man pages fstab(5),findfs(5),mount(8) and/or blkid(8) for more info
# UUID=de95356-5ff2-feed-9e0e-0872570a1866 / ext4	 defaults 0 0
# fs-efw-west-1.amazonaws.com:/home/ec2-user/efs nfs4 nfsvers=4.1,rsize=1048576,wsize=1048576,soft,timeo=100,retrans=2,actimeo=1

[user@ip-172-31-21-60 ~]$`
```

**Figure 7 - Fstab example**

Use the `mount` and the `df` commands to verify the EFS file share is mounted.

```
[user@ip-172-31-21-60 ~]$ df

Filesystem  1K-blocks Used Available Use% Mounted on
/dev/nvda2  10473452 2046516  8426936  20%  /
devtmpfs  1914986 0 1914986 0%  /dev
tmpfs  1940204 0 1940204 0%  /dev/shm
tmpfs  1940204 16676 1923538 1%  /run
tmpfs  1940204 0 1940204 0%  /sys/fs/cgroup
fs-efw-west-1.amazonaws.com:/9007199254735668 /home/ec2-user/efs
mpfs  358044 0 358044 0%  /run/user/1000

[user@ip-172-31-21-60 ~]$`
```

**Figure 8 - df example**
2.3.3 Setting up the EFS/TLS mount for the EFS Storage

If Amazon Linux2 EC2 instances are being used, TLS can be used to provide encryption between the EMS Server and the EFS file system. Use the following steps to use TLS. **Note:** If TLS is not required, the EFS file system can be mounted on the Amazon Linux2 EC2 instances using standard NFS4 mount commands as described in the previous section.

- Ensure `amazon-efs-utils` has been installed.
- Create a new mount point on the Amazon Linux2 EC2 instance.
  - `mkdir /home/ec2-user/efs`
- Mount the new EFS file system with EFS. **Note:** The options used in the EFS mount are required! Failure to use the listed options may result in message loss/corruption, or EMS not performing at an acceptable level:
  - `sudo mount -t efs -o tls,_netdev,soft,timeo=100,actimeo=1,defaults <First 10 digits of the DNS Name for EFS file system>:/ /home/ec2-user/efs`
  - **Note:** The `<DNS Name for the EFS File system>` is the DNS name shown in the file systems page shown previously in the AWS console. It will be similar to `fs-abcd1234.efs.eu-west-1.amazonaws.com`. For the EFS/TLS mount, just use `fs-abcd1234`. The mount would be similar to:
    ```bash
    sudo mount -t efs -o tls,_netdev,soft,timeo=100,actimeo=1,defaults fs-abcd1234:/ /home/ec2-user/efs
    ```
- The mount can also be added to `/etc/fstab`. The following is an example of `/etc/fstab` with EFS/TLS:

```bash
# UUID=76e177a9-8195-43cf-84ae-14ea371008b6 / xfs defaults,noatime 1 1
fs-abcd1234:/ /home/ec2-user/efs efs tls,_netdev,soft,actimeo=1,timeo=100,defaults 0 0
```

**Figure 10 - Example of `/etc/fstab` with TLS**
3 Enterprise Message Service Installation and Configuration

This section will outline the installation and configuration of EMS on the EC2 Linux instances.

3.1 EMS Installation

**EMS version 8.4 or later must be used for EMS installations running on AWS environments.** Install EMS on all of the EC2 instances. Nothing specific or custom is required to the base configuration of EMS, so a typical EMS installation can be performed.

Once EMS is installed, use the following to configure EMS for F/T.

- On one of the EC2 instances used for the EMS server:
  - Create the directory on the EFS file system for the shared EMS configuration files and data stores. Ex: /home/ec2-user/efs/tibco/cfgmgmt/ems/data and /home/ec2-user/efs/tibco/cfgmgmt/ems/data/datastore
  - Copy the EMS configuration files (.conf) installed during the EMS installation to the newly created /home/ec2-user/efs/tibco/cfgmgmt/ems/data directory
  - Copy the tibemsd.conf to $TIBCO_HOME/ems/8.4/bin
  - Create the $TIBCO_HOME/ems/8.4/bin/logs directory

- On the second EC2 instance used for the EMS server:
  - Copy the tibemsd.conf to $TIBCO_HOME/ems/8.4/bin
  - Create the $TIBCO_HOME/ems/8.4/bin/logs directory

3.2 EMS Configuration

There are specific configuration changes which must be made to provide better write performance and reliability of EMS F/T on AWS/EFS. This section will discuss these changes. See the EMS User Guide for additional information on setting or the use of, any properties discussed.

3.2.1 Stores.conf

In stores.conf, modify/add the following:

- The file_minimum=xxGB should be added to each synchronous data store. By adding this property, EMS will pre-allocate the space on the shared storage the data store. This will provide a better message write throughput on disk. The minimum should be 1GB. Expect the initial startup of EMS to take longer as it creates and allocates the space for the store file.
- The file_crc=enabled should be added. The enables EMS to check for data integrity of the data store. This is now the default on EMS version 8.4.

The following is an example of stores.conf with the changes.

```plaintext
[sys.failsafe]
type=file
file=sync-msgs.db
mode=sync
file_minimum=2GB
file_crc=enabled

[sync2]
type=file
file=sync2-msgs.db
mode=sync
file_minimum=2GB
```
3.2.2 Factories.conf

The EMS client `reconnect` properties must be set to enable the EMS client to `reconnect` to the EMS server in the event of an EMS server failure in an F/T configuration. The reconnect properties can be defined in a number of ways, including in the Java/C code, TIBCO application’s configuration file, and/or through the connection factory when they are used.

The default values may be too low in AWS to reliably allow the EMS client to reconnect to the EMS server after a failover, especially with network or system failure.

It is recommended to set the `reconnect_attempt_count` to 100, and the `reconnect_attempt_delay` to 5000. With these values, the EMS client will attempt to reconnect 100 times, every 5 seconds.

**Note:** If an Amazon Linux2 EC2 instance, using EFS/TLS is used, it is recommended to increase the `reconnect` values. The `reconnect_attempt_timeout` property should also be added. The values should add up to allow the reconnects to attempt to reconnect for at least 480 seconds.

The following example shows the values for the `FTConnectionFactory` in `factories.conf`.

**Note:** In the following example for the `url`, `<server1>` is `tibems1` and the `<port1>` is 7222, and `<server2>` is `tibems2` and the `<port2>` is 7222. Substitute with the appropriate values for the environment.

```plaintext
[FTConnectionFactory]
  type                  = generic
  url                   = tcp://tibems1:7222,tcp://tibems2:7222
  reconnect_attempt_count = 100
  reconnect_attempt_delay = 5000
```

3.2.3 Tibemsd.conf

The `tibemsd.conf` for both EMS Servers needs to be updated for multiple properties.

These include:

- Location of all configuration files – The location must be on the EFS shared storage device.

```plaintext
# Configuration files.
users                   = "/home/ec2-user/efs/tibco/cfgmgmt/ems/data/users.conf"
groups                  = "/home/ec2-user/efs/tibco/cfgmgmt/ems/data/groups.conf"
topics                  = "/home/ec2-user/efs/tibco/cfgmgmt/ems/data/topics.conf"
queues                  = "/home/ec2-user/efs/tibco/cfgmgmt/ems/data/queues.conf"
acl_list                = "/home/ec2-user/efs/tibco/cfgmgmt/ems/data/acl.conf"
factories               = "/home/ec2-user/efs/tibco/cfgmgmt/ems/data/factories.conf"
routes                  = "/home/ec2-user/efs/tibco/cfgmgmt/ems/data/routes.conf"
bridges                 = "/home/ec2-user/efs/tibco/cfgmgmt/ems/data/bridges.conf"
transports              = "/home/ec2-user/efs/tibco/cfgmgmt/ems/data/transports.conf"
tibrvcm                 = "/home/ec2-user/efs/tibco/cfgmgmt/ems/data/tibrvcm.conf"
durables                = "/home/ec2-user/efs/tibco/cfgmgmt/ems/data/durables.conf"
channels                = "/home/ec2-user/efs/tibco/cfgmgmt/ems/data/channels.conf"
stores                  = "/home/ec2-user/efs/tibco/cfgmgmt/ems/data/stores.conf"
```

**Note:**

- `store: directory to store persistent messages.`
store = "/home/ec2-user/efs/tibco/cfgmgmt/ems/data/datastore"

• Log File location – The location must be on the local disk for the EC2 instance. The following example locates in the ems/8.4/bin/logs directory.

  logfile = "/home/ec2-user/tibco/ems/8.4/bin/logs/tibemsd2-1.log"

• Server and Client Heartbeat and timeout values – These properties determine how long the client/server listen for the heartbeat from the client/server, before disconnecting. These properties must be set longer than what is normally configured for a local F/T environment. The values shown below have been tested, and work well on AWS with NFS4.

  server_heartbeat_client = 10
  server_timeout_client_connection = 120
  client_heartbeat_server = 10
  client_timeout_server_connection = 120

  Note: Note: If an Amazon Linux2 EC2 instance, using EFS/TLS is used, it is highly recommended to increase the two timeout values to a minimum of 480.

• Enabling exiting disk error property – New property for EMS version 8.4. This property defines to EMS to exit when there is a disk error reading/writing to a device. This property will help prevent “Dual Active Server” conditions, sometimes seen in networked storage devices.

  always_exit_on_disk_error = enable

• FT properties – Normal properties for the defining the peer EMS server instance, heartbeat between instances, and etc.

• Define a value for destination_backlog_swapout. This will help limit excessive reads to the shared disk. A minimum of 10000 is recommended. If the queues, will persistent a larger number of messages, increase the size.

  destination_backlog_swapout = 10000

3.2.4 Starting the EMS Instances

Once the configuration files are updated, EMS can be started. Ensure the --forceStart parameter is used when starting EMS. Start both instances, taking note of which instance is the active EMS instance. 

Note: Leave the both windows to the EMS server instances open. This will be needed for the testing.
Figure 13 – Linux EMS Startup
4 Testing EMS Fault Tolerance on AWS with EFS

Once EMS has been started on the EC2 Linux instances, the failover testing can be performed. This section will outline several test cases, including EMS Server process failure, machine failure, and network failure. Tests are performed using queues with persistence set. This guarantees that the shared file system will be accessed during the tests.

4.1 EMS Client App Setup

The third EC2 instance is used to run the test application. EMS is shipped with sample Java applications which can be used for the testing. The tibjmsMsgProducerPerf utility should be used for the testing. All sample Java applications are located in $TIBCO_HOME/ems/8.4/samples/java. Use the following to setup the environment:

- Ensure the version Java 1.8 development environment is installed.
- Install EMS version 8.4 on the third EC2 instance following the EMS installation procedures.
- After the installation of EMS is completed:
  - C内陆 $TIBCO_HOME/ems/8.4/samples/java
  - ./setup.sh on Linux
  - javac *.java – This should compile all Java apps in the directory
- Ensure that at least one of the EMS server instances is running (both should be running)
- Use the TIBCO EMS Administration Tool to create the EMS Queue sync utilizing the $sys.failsafe data store. This is required for testing with a synchronous data store:
  - C内陆 $TIBCO_HOME/ems/8.4/bin
  - ./tibemstats64 tcp://<server>:port

Note: In the following examples, <server> is the IP address of tibems1 and the <port> is 7222. Substitute with the appropriate values for the environment.
4.2 Performing the EMS Fault Tolerant Test Cases

Three different tests should be performed:

1. EMS Process failure – Active EMS is stopped
2. Network failure – Network failure between the Active EMS Server machine, and the Azure File Share
3. System failure – Accidental restart of the Azure Red Hat Linux VM running the Active EMS server instance

This section will outline how to run these three tests, and what the expected results should be.

Note: All test cases must be run from the third virtual machine where the Java sample app was compiled.

4.2.1 EMS Process Failure Test
This test verifies that an EMS client continues to function correctly, with no message loss during an EMS server process failover.

Two EMS server instances will be running in a F/T configuration, while messages are being sent. The active EMS instance will be stopped, the stand-by EMS instance should take over, and continue processing messages until the EMS Java completes publishing messages.

Note: In the following examples, <server1> is the IP address of tibems1 and the <port1> is 7222, and <server2> is the IP address of tibems2 and the <port2> is 7222. Tibclient is the EC2 instance running the Java apps. Substitute with the appropriate values for the environment.

4.2.1.1 Running the Process Failure Test
- Three ssh terminal sessions are needed for this test; one for tibems1, one for tibems2, and one for tibclient
• Start EMS on tibems1 and tibems2 in the foreground. EMS on tibems1 should be the active EMS instance.
• From tibclient, start the Java application
  o Cd to STIBCO_HOME/ems/8.4/samples/java
  o ./.setup.sh

```
20000 times took 41.017 seconds, performance is 497 messages/second
```

Figure 15 - Running tibjmsMsgProducerPerf

• Immediately kill/stop the EMS instance on tibems1, with cntrl-C
• The standby EMS instance on tibems2 will become active, and recover all messages. It should be possible to stop and start the EMS instances a few times while the Java test application is running. The number of recovered messages will increase.
Figure 16 - Standby EMS becoming active on tibems2

- After the Java application completes, run `tibemsadmin64 tcp://tibems2:7222` (or `tibems1`, if it is active), to verify that there is a minimum of 20000 messages in the sync queue.
- Restart the EMS instance on `tibems1`, and stop the EMS instance on `tibems2`. EMS on `tibems1` should become active, and recover all 20K messages with no errors.
- Use `tibemsadmin64`, and purge the sync queue in preparation for the next test.
Stop and restart EMS on tibems1 and tibems2 in the foreground. EMS on tibems1 should be the active EMS instance.

4.2.1.2 Expected Results

The Java test application should complete, with a slight pause during failover but should resume sending messages once the failover is complete. No messages should be lost. There may be more than 20K messages, but never should there be less than 20K. Depending on the number of messages that must be recovered, the fail-over should be very short, possibly less than 1 second.

4.2.2 Network Failure Test on Linux

This test verifies that an EMS client continues to function correctly, with no message loss during a network failure between the active EMS server instance, and the EFS shared file system.

Two EMS server instances will be running in a F/T configuration, while messages are being sent. The TCP port will be blocked between then active EMS instance and the AWS/EFS file system via iptables. The active EMS instance should get a write error, and exit, allowing the stand-by EMS instance to gain the locks on the EMS data stores, and take over. The EMS Java application should continue processing messages until it completes.

Note: In the following Linux examples, <server1> is IP address of tibems1 and the <port1> is 7222, and <server2> is the IP address of tibems2 and the <port2> is 7222. Tibclient is the EC2 instance running the Java apps. Substitute with the appropriate values for the environment.

4.2.2.1 Running the Network Failure Test

- Four ssh terminal sessions are needed for this test; two for tibems1, one for tibems2, and one for tibclient
- A script will be needed to block the NFS port (2049) on tibems1 while the Java app is publishing messages. The following figure shows the drop_nfs.sh script. Cut and past the following to create the script. The script must be created in the second ssh terminal window on tibems1.
Figure 18 - drop_nfs.sh script

- From tibclient, start the Java application
  - Cd to STIBCO_HOME/ems/8.4/samples/java
  - ./setup.sh
- From the second ssh terminal window on tibems1, run sudo drop_nfs.sh
• The active EMS instance on *tibems1* should terminate with a disk write error:

```
[ec2-user@ip-172-31-21-60 ~]$ ./drop_nfs.sh
Saving existing IP table definitions
Wed Oct 4 17:35:42 UTC 2017
Dropping NFS 2049 port
Sleeping for 2 minutes
Wed Oct 4 17:37:42 UTC 2017
Restoring original IP table definitions
Done.
[ec2-user@ip-172-31-21-60 ~]$  
```

*Figure 19 - Running drop_nfs.sh*

```
TIBCO Enterprise Message Service.
Copyright 2003-2017 by TIBCO Software Inc.
All rights reserved.
Version 8.4.0 V14 7/20/2017
2017-10-04 17:35:16.838 Process started from './tibemsd64'.
2017-10-04 17:35:16.838 Process Id: 1576
2017-10-04 17:35:16.838 Hostname: ip-172-31-21-60.eu-west-1.compute.internal
2017-10-04 17:35:16.838 Hostname IP address: [fe80::fe:abff:fede:024]  
2017-10-04 17:35:16.838 Hostname IP address: 172.31.21.60
2017-10-04 17:35:16.838 Reading configuration from 'tibemsdl.conf'.
2017-10-04 17:35:16.840 Logging into file '/home/ec2-user/tibco/ems/8.4/bin/logs/tibemsdl.log'
2017-10-04 17:35:16.841 Server name: 'EMS-SERVER-EF2'.
2017-10-04 17:35:16.841 Storage Location: '/home/ec2-user/efs/tibco/cfgmgmt/ems/data/datastore'.
2017-10-04 17:35:16.841 Routing is disabled.
2017-10-04 17:35:16.841 Authorization is disabled.
2017-10-04 17:35:16.841 The server will attempt to trace warnings about destinations that are growing unbounded above 5687091 bytes or 50000 messages.
2017-10-04 17:35:16.841 Set server properties 'large_destination_memory' and 'large_destination_count' respectively to alter these thresholds.
2017-10-04 17:35:16.962 Continuing as active server.
2017-10-04 17:35:16.962 Accepting connections on tcp://ip-172-31-21-60.eu-west-1.compute.internal/0.0.0.0:7222.
2017-10-04 17:35:16.961 Recovering state, please wait.
2017-10-04 17:35:17.468 Server is active.
2017-10-04 17:35:25.364 Standby server 'EMS-SERVER-EF2@ip-172-31-26-245.eu-west-1.compute.internal' has connected.
```

*Figure 20 - Disk Write Error on tibems1*
- The standby EMS instance on tibems2 should determine EMS on tibems1 is no longer producing a heartbeat, will attempt to become active. Depending on the amount of data, this should take ~90 seconds to occur. There can be other warnings, depending on how long it takes for tibems2 to obtain the locks.
- After the Java application completes, run tibemsadmin64 tcp://tibems2:7222 (or tibems1, if it is active), to verify that there is a minimum of 20000 messages in the sync queue.
- Restart the EMS instance on tibems1, and stop the EMS instance on tibems2. EMS on tibems1 should become active, and recover all 20K messages with no errors.
- While still in tibemsadmin64, purge the sync queue in preparation for the next test.
- Stop and restart EMS on tibems1 and tibems2 in the foreground. EMS on tibems1 should be the active EMS instance.

### 4.2.2.2 Expected Results

The Java test application should complete, pausing during the failover, but should resume sending messages once the failover is complete. No messages should be lost. There can be more than 20K messages, depending on the number of connections/threads, but never should there be less than 20K. Depending on the number of messages that must be recovered, the fail-over take ~90 seconds. It has been observed with the AWS/EFS file system that with the network failure test, the EMS recovery of the messages will take longer than the other tests.

### 4.2.3 System Failure Test

This test verifies that an EMS client continues to function correctly, with no message loss during a system failure on the EC2 instance running the active EMS server instance. This is not a normal occurrence. However, it is possible to accidentally restart the virtual machine from the AWS console.

Two EMS server instances will be running in a F/T configuration, while messages are being sent. From the AWS console, the EC2 instance where the active EMS instance is running will be restarted. The stand-by EMS instance should be able to gain the locks on the EMS data stores, and take over. The EMS Java application should continue processing messages until it completes.

**Note:** In the following examples, <server1> is the IP address of tibems1 and the <port1> is 7222, and <server2> is the IP address of tibems2 and the <port2> is 7222. Tibclient is the EC2 instance running the Java apps. Substitute with the appropriate values for the environment.

### 4.2.3.1 Running the System Failure Test

- Three ssh terminal sessions are needed for this test; one one each of the virtual machine.
- The AWS console must also be available, and be on the EC2 dashboard page.

- From tibclient, start the Java application
  - Cd to STIBCO_HOME/ems/8.4/samples/java
  - ./setup.sh

![Figure 21 - AWS Console on the EC2 Dashboard](image)
4.2.3.2 Expected Results

The Java test application should complete, pausing during failover, but should resume sending messages once the failover is complete. No messages should be lost. There can be more than 20K messages, depending on the number of messages to be recovered.

From the EC2 dashboard, click on the `tibems1` instance, click on the `actions, instance state`, and finally click on `reboot`. This will reboot the `tibems1` EC2 instance.

The ssh terminal window to the `tibems1` EC2 Linux instance should immediately terminate, and the stand-by EMS instance should recover all messages, and become active within a few seconds, depending on the number of messages to be recovered.

After the Java application completes, run `tibemsadmin64 tcp://tibems2:7222` (or to the active EMS instance), to verify that there is a minimum of 20000 messages in the sync queue.

Restart the EMS instance on the restarted EC2 instance, and stop the currently active EMS instance on the second EC2 instance. EMS should become active on the restarted instance, and recover all 20K messages with no errors.

Use `tibemsadmin64` to verify, then `purge` the sync queue.

Stop EMS on both Linux EC2 instances.

This concludes the tests, so all processes, terminal windows, and EC2 instances can be stopped.
connections/threads, but there should never be less than 20K messages. Depending on the number of messages that must be recovered, the fail-over should only take a few seconds. It has been observed with the AWS reboot of the EC2 instance than the EMS recovery is virtually no longer than a EMS process failure.

**Note:** If an Amazon Linux2 EC2 instance, using EFS/TLS is used, the EMS recovery time may take as long as seven minutes, in some situations to complete. Testing this scenario multiple times is recommended.