Ultra Messaging (Version 6.14)

Troubleshooting Guide

Contents

Chapter 1

Introduction

This document outlines the general procedure for diagnosing UM problems.

For policies and procedures related to Ultra Messaging Technical Support, see UM Support.

NEXT SECTION: Troubleshooting Overview

Chapter 2

Troubleshooting Overview

For a deep technical introduction to Ultra Messaging, see **Fundamental Concepts**.

This guide assumes you have a fully-developed distributed system using UM messaging which has been demonstrate to work within the development environment. Now you are having trouble in a QA or production environment. The goal is to identify and fix problems as quickly as possible.

ASSUMPTIONS:

- Streaming use case (not Persistence or Queuing).
- Multicast Topic Resolution (no lbmrd or SRS).
- LBT-RM transport.
- Kernel-based NIC driver (no kernal-bypass).
- Microsoft Windows.
- Pub/Sub in same TRD (no DRO involved).
- Receivers using main context thread (no XSP involved).
- Receivers using context callbacks (no event queue involved).

The problems covered in this guide are:

- Troubleshooting Deafness - subscriber not getting messages.
- Troubleshooting Loss - subscriber reports loss.
- Troubleshooting Memory Growth - publisher/subscriber growing.

Attention

Before starting, please try to satisfy the Troubleshooting Prerequisites.

NEXT SECTION: Troubleshooting Deafness

Chapter 3

Troubleshooting Deafness

The most common problem is that a subscriber for a UM topic is deafness - not receiving messages from one or more publishers of that topic.

3.1 Application Logs BOS / EOS

If your application logs BOS and EOS events, follow these steps. (If not, go to Try Lbmrcv.)

Troubleshooting steps:

1. Identify the UM topic name. (Requires knowledge of the application.)
2. Log into the subscriber host.
3. Examine the subscriber's log file. Look for BOS and EOS events for the topic in question. The **Source String** will identify the IP of the publisher.
    - Have multiple EOS events: Go to Rolling EOS.
    - Have no BOS or EOS: Go to Check Multicast TR.
    - Have BOS but no EOS. Go to next step.
4. Check the IP address in the BOS **Source String**. Verify that it is that of the desired publisher.
    - If not, then treat this as "no BOS or EOS" and go to Check Multicast TR.
    - If the BOS **Source String** IP address is correct, this suggests that the connection is valid and that the publisher is not sending messages. Check application logic.

3.2 Try Lbmrcv

Troubleshooting steps:

1. Identify the UM topic name. (Requires knowledge of the application.)
2. Log into the subscriber host.
3. Determine the configuration in use by the subscriber.

4. Create a stand-alone configuration file consumable by "lbmrcv". This can be XML or flat file.

   If using an XML file and you need an application name to match to, set the following environment variables:

   - LBM_XML_CONFIG_APPNAME=app_name
   - BM_XML_CONFIG_FILENAME=file_name

   If using a flat file configuration file, set the following environment variable:

   - LBM_DEFAULT_CONFIG_FILE=file_name

   These should be set prior to the next step.

5. Run the "lbmrcv" command (see Troubleshooting Tools):

   ```
   lbmrcv -v -v topic_name
   ```

   where "topic_name" is the topic experiencing deafness.

6. Watch lbmrcv's output for BOS / EOS events. It may take several seconds. The **Source String** will identify the IP of the publisher.

   - Have multiple EOS events: Go to Rolling EOS.
   - Have no BOS or EOS: Go to Check Multicast TR.
   - Have BOS but no EOS. Go to next step.

7. Check the IP address in the BOS **Source String**. Verify that it is that of the desired publisher.

   - If not, then treat this as "no BOS or EOS" and go to Check Multicast TR.
   - If the BOS **Source String** IP address is correct, this suggests that the connection is valid and that the publisher is not sending messages. Check application logic.

Attention

Performing this test can sometimes "fix" the deafness of the subscriber application. If that is the case, then the problem is related to Topic Resolution going **Quiescent**. For example, there may have been a network outage that lasted longer than the Topic Resolution sustaining periods. The **Sustaining Phase** should be extended using **resolver_advertisement_minimum_sustain_duration (source)** and/or **resolver_query_minimum_sustain_duration (receiver)**.

3.3 Rolling EOS

A UM receiver can be in a state where it repeatedly reports EOS events (End Of Session) for a topic without corresponding BOS events (Beginning Of Session). These events are delivered to the application via the receiver callback and should be logged (see Application Log Files). See **Receiver BOS and EOS Events** for deeper technical information on BOS/EOS.

The EOS events should occur every 60 seconds by default, but check your value for the configuration option **transport_lbtrm_activity_timeout (receiver)**.

This state indicates that the subscriber is able to get Topic Resolution packets from the publisher, but the data transport packets can't get through.

Troubleshooting steps:

1. Examine the EOS message to determine the multicast group and port for the data transport. This comes from the **Source String**. For example:

```
LBTRM:10.29.3.88:14390:e0679abb:231.13.13.13:14400[1539853954]
```

- 10.29.3.88 - The IP address of the publisher host.
- 231.13.13.13 - The transport group.
- 14400 - The transport port.

2. Go to Verify Multicast Connectivity with **app_group** defined as the transport group and **app_port** defined as the transport port.

3.4 Check Multicast TR

Troubleshooting steps:

1. Log into the subscriber and publisher hosts.
2. Verify that the subscriber and publisher are configured for the same **Topic Resolution group address**.
3. On the publisher's host, list the multicast groups that the host is joined to. Verify that the configured **Topic Resolution group address** is listed and has a reference count of at least 1. If not, restart the publisher.

   On Windows the command is:

   ```
   netsh interface ipv4 show joins
   ```

   On Unix the command is:

   ```
   netstat -gn | egrep -v ":"
   ```

4. Do the same for the subscriber.
5. Verify multicast connectivity for the Topic Resolution group address. Go to Verify Multicast Connectivity with **app_group** defined as the configured value for **resolver_multicast_address (context)** and **app_port** defined as the configured value for **resolver_multicast_port (context)**.
6. If mtools verifies that the TR multicast works, Try Lbmrcv.

3.5 Verify Multicast Connectivity

This checks multicast connectivity between two hosts: A and B. You should already know which multicast group and port you are checking: **app_group** and **app_port**.

This sequence uses "mtools" commands "msend" and "mdump". See Troubleshooting Tools.

Troubleshooting steps:

1. Choose a port that is not the application port: **alt_port**.
2. On host A open two command tool windows.
3. In command tool A1, run:

   ```
   msend app_group alt_port
   ```

4. In command tool A2, run:

   ```
   mdump app_group alt_port
   ```

5. **Result**: the "mdump" command should display the messages sent.
   Note: leave both commands running for now.

6. On host B open two command tool windows.

7. In command tool B1, run:

   ```
   msend app_group alt_port
   ```

8. In command tool B2, run:

   ```
   mdump app_group alt_port
   ```

9. **Result**: the "mdump" commands on both hosts should display the messages sent by both hosts.

If the "mtools" commands can't communicate, then there is a serious problem with either the host OS or the network. (Diagnosing host and network problems is beyond the scope of this guide.)

NEXT SECTION: Troubleshooting Loss

Chapter 4

Troubleshooting Loss

TBD

NEXT SECTION: Troubleshooting Memory Growth

Chapter 5

Troubleshooting Memory Growth

TBD

NEXT SECTION: Troubleshooting Prerequisites

Chapter 6

Troubleshooting Prerequisites

Users can influence the ease and effectiveness of troubleshooting by following some guidelines:

- Operations Alerting
- Application Log Files
- UM Log Alerting
- Configuration Recommendations
- Troubleshooting Tools
- Monitoring Tools

6.1 Operations Alerting

Every application should have a way to alert the operations staff of trouble, both during testing and production. An effective alerting system is heavily dependent on the user's operational environment, and cannot be defined here. However, having an alerting system will save a lot of time when troubleshooting since the root cause of a user-visible problem often doesn't reside with the component that shows the problem.

Having an alerting system can help quickly pin-point the true source of the problem.

6.2 Application Log Files

All applications and test tools that use UM must log significant UM events. These include:

- All receiver event types except DATA. The "unrecoverable loss" event should typically produce an operations alert.
- All failed API calls. These should generally produce an operations alert. (An exception might be a send function "wouldblock" error which the application is written to expect and handle appropriately.)
- All UM logging callbacks. These should be able to generate an operations alert, depending on the content of the UM log. See UM Log Alerting.

The logging should include:

- A timestamp with at least millisecond resolution.
- Thread ID.
- Severity level. Most API failures should be treated as "error" or "severe error".

  Logs delivered via the UM logging callback should use the severity level passed to the logging callback.

  Severity levels of "error" and above should normally generate an alert, although see UM Log Alerting.
- Text of error message.
- For events delivered by the receiver callback, include the topic name and **Source String**.

Application logging to a file should not be done synchronously with the application code, as this can introduce disruptive delays. Messages to be logged should be passed to a separate application logging thread via a queue. Care must be taken in the design of the logging thread and queue so that very rapid log generation does not exhaust available memory.

6.2.1 UM Log Alerting

Informatica has evaluated each log message and has assigned a severity level to each one. This can be used by the application as a first approximation to whether to trigger an operations alert.

However, experience has demonstrated that it is not possible to accurately assign a severity level that applies equally to different use cases. In short: UM cannot know whether a given log deserves immediate attention of operations staff, or if it is just "noise". This must be determined empirically by the user.

One good strategy is to pay less attention to UM's assigned severity level and instead create a list of messages that are known to be "normal" for a given application or application class. A log scanner would then send an operations alert for any log message *not* in that normal list. This log scanner should run in real-time to ensure timely alerting.

For this strategy to be successful, it must be very easy to add new messages to the normal list. I.e. it should not be compiled into the application code. During the early phases of system deployment, new messages will occasionally show up that are informational and do not need attention, and should be added to the normal list. We encourage our users to contact Informatica Support to review your normal list.

Do not use the text of the error message itself for matching as we sometimes change the text to add helpful information. Instead use the initial message ID. For example, "Core-5688-3273".

It may also be appropriate to apply some thresholding to specific messages in terms of their frequency. One or two EOS events might be perfectly normal. But 20 or 30 might indicate a serious problem. Unfortunately, these are very difficult to identify and plan for up front. For example, EOS might be initially declared as normal, and everything will be fine for years. Then the network might develop a multicast routing problem which manifests as Rolling EOS, which indicates a serious problem.

6.3 Configuration Recommendations

TBD lbt-rm loss recovery (NAK backoff)

TBD **UDP-Based Topic Resolution Strategies**

TBD TR durations

6.4 Troubleshooting Tools

The following tools should be available on all test and production machines.

- **msend and mdump** (mtools). See https://community.informatica.com/solutions/informatica_mtools for source and executables.
- **lbmsrc and lbmrcv**. Part of the UM product package in the "bin" directory.
- **Process stack dumps**.

  Most Unix systems have a "pstack" command that shows a per-thread stack back trace snapshot of a running process. This command is minimally invasive.

  For Windows, Microsoft offers Process Monitor at https://docs.microsoft.com/en-us/sysinternals/downl as part of a larger set of process utilities; see https://docs.microsoft.com/en-us/sysinternals/downloads
- **Core dumps**. The operating system should be set up to allow the writing of crash dumps (core files) if a process aborts.

6.5 Monitoring Tools

6.5.1 Monitoring Daemons

TBD alerting.

TBD stats via web vs publish.

TBD store, dro, lbmrd, srs.

TBD daemon log files.

6.5.2 UM Transport Monitoring

TBD Integrate into application alerting system (self monitoring).

TBD loss can be a warning of future unrecoverable loss.

6.5.3 Host Monitoring

There are many different host monitoring packages, both open source and enterprise, which run continuously and allow the user to examine current and historical system statistics.

Informatica recommends a time granularity of at most 5 minutes between samples.

- Socket buffer overflow.

- NIC drops.
- CPU loading, ideally per thread.
- Processes running.

TBD alerting.