



Informatica™

**Ultra Messaging** (Version 6.17)

# C Examples



# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	C Examples Introduction . . . . .	5
1.2	Configuring C Examples . . . . .	6
1.3	Building C Examples . . . . .	6
1.4	Unhandled C Events . . . . .	6
1.5	C Example Support Files . . . . .	7
1.6	Persistence Tutorial C Files . . . . .	7
1.7	C Examples . . . . .	8
1.7.1	Example lbmhfrvc.c . . . . .	8
1.7.2	Example lbmhfrvcq.c . . . . .	8
1.7.3	Example lbmhsrc.c . . . . .	9
1.7.4	Example lbmhsrcv.c . . . . .	9
1.7.5	Example lbmtrvc.c . . . . .	11
1.7.6	Example lbmimsg.c . . . . .	11
1.7.7	Example lbmireq.c . . . . .	12
1.7.8	Example lbmlatping.c . . . . .	12
1.7.9	Example lbmlatpong.c . . . . .	13
1.7.10	Example lbmmon.c . . . . .	13
1.7.11	Example lbmmoncache.c . . . . .	14
1.7.12	Example lbmmon_cmd.c . . . . .	16
1.7.13	Example lbmmondata.c . . . . .	16
1.7.14	Example lbmmonudp.c . . . . .	16
1.7.15	Example lbmmrcv.c . . . . .	18
1.7.16	Example lbmmrcvq.c . . . . .	19
1.7.17	Example lbmmreq.c . . . . .	21
1.7.18	Example lbmmsrc.c . . . . .	21
1.7.19	Example lbmpong.c . . . . .	23
1.7.20	Example lbmprice.c . . . . .	23
1.7.21	Example lbmrcv.c . . . . .	24
1.7.22	Example lbmrcvq.c . . . . .	25
1.7.23	Example lbmrcvxsp.c . . . . .	27

---

1.7.24	Example lbmreq.c	27
1.7.25	Example lbmresp.c	28
1.7.26	Example lbmresping.c	28
1.7.27	Example lbmrespq.c	29
1.7.28	Example lbmspike.c	29
1.7.29	Example lbmsrc.c	30
1.7.30	Example lbmssrc.c	31
1.7.31	Example lbmssrcreq.c	33
1.7.32	Example lbmstrm.c	33
1.7.33	Example lbmtrreq.c	34
1.7.34	Example lbmwrcv.c	34
1.7.35	Example lbmwrcvq.c	36
1.7.36	Example minrcv.c	37
1.7.37	Example minrcv.cpp	37
1.7.38	Example minsrc.c	37
1.7.39	Example srs_cmd.c	38
1.7.40	Example srs_monitor_info_receiver.c	38
1.7.41	Example tnwgdcmd.c	39
1.7.42	Example tnwgdmon.c	39
1.7.43	Example umedcmd.c	40
1.7.44	Example umedmon.c	40
1.7.45	Example ume-example-rcv-2.c	40
1.7.46	Example ume-example-rcv-3.c	40
1.7.47	Example ume-example-rcv.c	41
1.7.48	Example ume-example-src-2.c	41
1.7.49	Example ume-example-src-3.c	41
1.7.50	Example ume-example-src.c	42
1.7.51	Example umercv.c	42
1.7.52	Example umesnaprepo.c	43
1.7.53	Example umesrc.c	43
1.7.54	Example umessrc.c	44
1.7.55	Example umestored_example.c	45
1.7.56	Example umqrcv.c	45
1.7.57	Example umqsrc.c	47
1.8	Example Protocol Files	48
1.8.1	Example dro_mon.proto	48
1.8.2	Example srs_mon.proto	48
1.8.3	Example um_mon_attributes.proto	48
1.8.4	Example um_mon_control.proto	48
1.8.5	Example ump_mon.proto	48

---

---

1.8.6 Example ums\_mon.proto ..... 48



# Chapter 1

## Introduction

This document lists and gives some background information on the C-language example UM programs.

For policies and procedures related to Ultra Messaging Technical Support, see [UM Support](#).

**(C) Copyright 2004,2025 Informatica Inc. All Rights Reserved.**

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

A current list of Informatica trademarks is available on the web at <https://www.informatica.com/trademarks.html>.

Portions of this software and/or documentation are subject to copyright held by third parties. Required third party notices are included with the product.

This software is protected by patents as detailed at <https://www.informatica.com/legal/patents.html>.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, please report them to us in writing at Informatica LLC 2100 Seaport Blvd. Redwood City, CA 94063.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided.

INFORMATICA LLC PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

See **UM Glossary** for Ultra Messaging terminology, abbreviations, and acronyms.

### 1.1 C Examples Introduction

These programs were written to help in troubleshooting, testing, and demonstrating UM coding techniques. See also [Java Example Source Code](#) and [C# Example Source Code](#).

Since the tools are written to be useful as well as instructive, they are more complex than purely-instructive examples would be, with many options to add or subtract functionality. See [UMExamples](#) for purely-instructive examples of a variety of UM use cases.

The example C programs listed here are provided in both source form and in binary executable form.

## 1.2 Configuring C Examples

The example programs universally provide the "-c filename" command-line option. Using that option, the example application calls the `lbm_config()` API. However, note that this API is not recommended for use with XML-format LBM configuration files, largely because you are not able to specify an application name.

To use an XML configuration file with a UM example application, set the environment variables:

- `LBM_XML_CONFIG_APPNAME` - Desired name of application.
- `LBM_XML_CONFIG_FILENAME` - Path name of XML configuration file.

In this way, UM will correctly set the example application's name and will properly load the XML configuration file.

## 1.3 Building C Examples

Most users are not interested in compiling these sources in their current form, but instead use them for "spare parts", extracting fragments of code as appropriate. For users who wish to build the tools, and especially for users who may want to modify the tools, we recommend creating a new directory and copying all of the doc/example directory contents into that new directory.

The documentation below includes example build commands that have been tested on Linux. Note that the environment variable "LBM" should be set to the platform-specific UM directory that contains the "include" and "lib" directories. For example:

```
export LBM=$HOME/UMP_6.17/Linux-glibc-2.17-x86_64
```

For windows builds, use the Unix build line to show the required libraries for each application.

## 1.4 Unhandled C Events

Each of the example programs is written to demonstrate a subset of UM's total available functionality. For example, some programs are written to demonstrate **Streaming** functionality (e.g. `lbmsrc`), while other programs are written to demonstrate **Persistence** functionality (e.g. `umesrc`), while still other programs are written to demonstrate **Queuing** functionality (e.g. `umqsrc`).

UM is generally designed to be event-driven, with events being delivered to the programs through standard callbacks, like source callbacks and receiver callbacks. There are many events which are common across all streaming, persistence, and queuing. Other events are specific to persistence, and still other events are specific to queuing.

This can lead to example programs reporting "unknown" or "unhandled" events. For example, if the "lbmsrc" streaming program is run with a configuration file that enables persistence, UM will deliver events that are specific to persistence to the "lbmsrc" program. But "lbmsrc" is designed for streaming, and does not include code cases for persistence or queuing events. Maybe you should change your configuration to disable persistence, or you should be using the "umesrc" example program.

Similarly, the "umqsrc" program expects queuing functionality, and can report unhandled events if persistence is configured. Or "umesrc" can report unhandled events if queuing is configured.

If you see an unhandled event, it is generally reported as a number. You can see which event this corresponds to by looking up the number in:

- **C Receiver Events** for subscribing programs and
-

- **C Source Events** for publishing programs.

Once you understand the nature of the unhandled event, you can decide how to change your configuration or choose a different program.

## 1.5 C Example Support Files

There are several source files in the example directory that contain useful functions to the main example programs.

**getopt.c** - utility functions to parse command-line options (for Windows).

**verifymsg.c** - utility function to help some programs create verifiable messages.

**monmodopts.h** - common include file used by many of the example programs. It includes option information for monitoring functionality.

**replgetopt.h** - common include file used by many of the example programs. It includes definitions for alternate getopt functions.

**verifymsg.h** - common include file used by many of the example programs. It includes definitions for "verifymsg.c" (which needs to be linked into many programs).

**srs\_monitor\_info\_msg.c** - Module used by the srs\_monitor\_info\_receiver program.

**srs\_monitor\_info\_msg.h** - Definitions for srs\_monitor\_info\_msg.c module.

**srs\_cmd\_msg.c** - Module used by the srs\_cmd program.

**status.c** - Module used by the "lbmlatping" and "lbmlatpong" programs.

**srs\_cmd\_msg.h** - Definitions for srs\_cmd\_msg.c module.

**lbmmondiag.pl** - Reads UDP packets and process statistics. See [lbmmonudp.c](#) and [lbmmondiag.pl](#) in the UM Operations Guide.

## 1.6 Persistence Tutorial C Files

See **Demonstrating Persistence** for information on these files.

**ume-example-src.c** - Initial source application used in the tutorial.

**ume-example-rcv.c** - Initial receiver application used in the tutorial.

**ume-example-src-2.c** - Source application modified to use a UMP persistent store.

**ume-example-rcv-2.c** - Receiver application modified to use a UMP persistent store.

**ume-example-src-3.c** - Modified source application used to demonstrate persistence.

**ume-example-rcv-3.c** - Modified receiver application used to demonstrate persistence.

**ume-example-config.xml** - Elementary persistent store configuration file used for the tutorial.

---

## 1.7 C Examples

### 1.7.1 Example lbmhfrvc.c

Example build:

```
gcc lbmhfrvc.c verifymsg.c -o lbmhfrvc -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmhfrvc.c](#)

Purpose: application that receives messages from a given topic using a single hot-failover receiver.

```
Usage: lbmhfrvc [-AEhsvV] [-c filename] [-r msgs] [-U losslev] topic
-A = display messages as ASCII text
-c filename = Use LBM configuration file filename.
              Multiple config files are allowed.
              Example: '-c file1.cfg -c file2.cfg'
-d, --msec-delay=NUM      Implements a number of milliseconds sleep per
                           message received
-E = exit after source ends
-h = help
-r msgs = delete receiver after msgs messages
-s = print statistics along with bandwidth
-S = Exit after source ends, print throughput summary
-v = be verbose about incoming messages (-v -v = be even more verbose)
-V = verify message contents
```

### 1.7.2 Example lbmhfrvcq.c

Example build:

```
gcc lbmhfrvcq.c verifymsg.c -o lbmhfrvcq -I$LBM/include -I$LBM/include/lbm -L$LBM/
lib -lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmhfrvcq.c](#)

Purpose: lbmhfrvc.c: application that receives messages from a given topic using a single hot-failover receiver and an event queue.

```
Usage: lbmhfrvcq [-EhsvV] [-c filename] [-r msgs] [-U losslev] topic
-c filename = Use LBM configuration file filename.
              Multiple config files are allowed.
              Example: '-c file1.cfg -c file2.cfg'
-E = exit after source ends
-h = help
-r msgs = delete receiver after msgs messages
-s = print statistics along with bandwidth
-S = Exit after source ends, print throughput summary
-U losslev = exit after losslev % unrecoverable loss
-v = be verbose about incoming messages (-v -v = be even more verbose)
-V = verify message contents
```

### 1.7.3 Example lbmhfsrc.c

Example build:

```
gcc lbmhfsrc.c verifymsg.c -o lbmhfsrc -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmhfsrc.c](#)

Purpose: application that sends to a given topic using a single hot-failover source.

Usage: lbmhfsrc [options] topic

Available options:

```
-c filename = Use LBM configuration file filename.
                Multiple config files are allowed.
                Example: '-c file1.cfg -c file2.cfg'
-d delay = delay sending for delay seconds after source creation
-h = help
-i init = start at message init instead of 0
-l len = send messages of len bytes
-L linger = linger for linger seconds before closing context
-M msgs = send msgs number of messages
-N NUM = send on channel NUM
-P msec = pause after each send msec milliseconds
-R [UM]DATA/RETR = Set transport type to LBT-R[UM], set data rate limit to
                DATA bits per second, and set retransmit rate limit to
                RETR bits per second. For both limits, the optional
                k, m, and g suffixes may be used. For example,
                '-R 1m/500k' is the same as '-R 1000000/500000'
-s sec = print stats every sec seconds
-t filename = use filename contents as a recording of message sequence numbers
-V = construct verifiable messages
-x bits = Use 32 or 64 bits for hot-failover sequence numbers
```

### 1.7.4 Example lbmhfxrcv.c

Example build:

```
gcc lbmhfxrcv.c verifymsg.c -o lbmhfxrcv -I$LBM/include -I$LBM/include/lbm -L$LBM/
lib -lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmhfxrcv.c](#)

Purpose: application that receives messages from a given topic using a single hot-failover receiver across contexts (HFX).

Usage: lbmhfxrcv [-aACdEfhqsSvV] [-I interface] [-c filename] [-r msgs] [-U losslev] topic

Available options:

```
-a, --arrival-order    deliver messages in the order that they arrive.
-A, --ascii           display messages as ASCII text (-A -A = newlines after each
msg)
-c, --config=FILE     Use LBM configuration file FILE.
                    Multiple config files are allowed.
                    Example: '-c file1.cfg -c file2.cfg'
-C, --context-stats   fetch context rather than receiver stats
-d, --deliver-dups    Enable duplicate delivery
-E, --exit            exit when source stops sending
-h, --help            display this help and exit
```

```

-I, --iface=CIDR      create a context on the interface specified by CIDR
                      Multiple interfaces are allowed.
                      Example: '-I 10.29.1.0/24 -I 10.29.2.0/24'
-q, --eventq          use an LBM event queue
-r, --msgs=NUM        exit after NUM messages
-O, --orderchecks     Enable message order checking
-s, --stats=NUM       print LBM statistics every NUM seconds
--max-sources=NUM     allow up to NUM sources (for statistics gathering purposes)
-S, --stop            exit when source stops sending, and print throughput summary
-U, --losslev=NUM     exit after NUM% unrecoverable loss
-v, --verbose         be verbose about incoming messages (-v -v = be even more
                      verbose)
-V, --verify          verify message contents

```

#### Monitoring options:

```

--monitor-rcv=NUM      monitor receiver every NUM seconds
--monitor-ctx=NUM      monitor context every NUM seconds
--monitor-transport=TRANS
                      use monitor transport module TRANS
                      TRANS may be 'lbm', 'lbmsnmp', or 'udp', default
                      is 'lbm'
--monitor-transport-opts=OPTS
                      use OPTS as transport module options
--monitor-format=FMT   use monitor format module FMT
                      FMT may be 'csv' or 'pb'
--monitor-format-opts=OPTS
                      use OPTS as format module options
--monitor-appid=ID     use ID as application ID string

```

Transport and format options are passed as name=value pairs, separated by a semicolon.

The entire option string should be enclosed in double-quotes.

#### LBM transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE           use LBM configuration file FILE
topic=TOPIC           send statistics on topic TOPIC
                      default is /29west/statistics
allow_debug=VAL       VAL may be 'off' or 'on'
                      defaults to 'off'

```

#### LBMSNMP transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE           use LBM configuration file FILE
topic=TOPIC           send statistics on topic TOPIC
                      default is /29west/statistics

```

#### UDP transport options:

```

address=IP            send statistics to address IP
port=NUM              send to UDP port NUM
                      default is 2933
mcgroup=GRP           send on multicast group GRP
bcaddress=IP          send statistics to broadcast address IP
ttl=NUM               send multicast statistics with TTL NUM
                      default is 16

```

#### CSV format options:

```

separator=CHAR        separate CSV fields with character CHAR
                      defaults to ','
                      Don't use a semicolon!

```

PB format options:

filters=FILE use FILE that contains filter options

### 1.7.5 Example lbmhtrcv.c

Example build:

```
gcc lbmhtrcv.c -o lbmhtrcv -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-lldb -lm -lrsock -lprotobuf-c
```

Source code: [lbmhtrcv.c](#)

Purpose: application that receives from a collection of HyperTopic patterns.

Usage: lbmhtrcv [options] <patterns\_file

Where 'patterns\_file' is a simple text file, supplied as standard input,  
containing one pattern per line.

Available options:

-d msec	delete hypertopic receiver every msec milliseconds
-h, --help	display this help and exit
-p string	set hypertopic prefix to string
-q	use event queue
-s, --statistics	print statistics along with bandwidth
-v, --verbose	be verbose about incoming messages
-x	exit after all receivers deleted

### 1.7.6 Example lbmimsg.c

Example build:

```
gcc lbmimsg.c -o lbmimsg -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-lldb -lm -lrsock -lprotobuf-c
```

Source code: [lbmimsg.c](#)

Purpose: application that sends immediate messages (either unicast or multicast)  
as fast as possible, either to a topic, or send topicless.

Usage: lbmimsg [options] topic

lbmimsg [options] -o

Available options:

-c filename	= Use LBM configuration file filename. Multiple config files are allowed. Example: '-c file1.cfg -c file2.cfg'
-d delay	= delay sending for delay seconds after source creation
-h	= help
-l len	= send messages of len bytes
-L linger	= linger for linger seconds before closing context
-M msgs	= send msgs number of messages
-n num	= Append a number between 1 and num to topic
-o	= send topic-less immediate messages
-P msec	= pause after each send msec milliseconds
-R [UM]DATA/RETR	= Set transport type to LBT-R[UM], set data rate limit to DATA bits per second, and set retransmit rate limit to RETR bits per second. For both limits, the optional

---

k, m, and g suffixes may be used. For example,  
 '-R 1m/500k' is the same as '-R 1000000/500000'  
 -T target = target for unicast immediate messages

### 1.7.7 Example lbmireq.c

Example build:

```
gcc lbmireq.c -o lbmireq -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
  -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmireq.c](#)

Purpose: application that sends immediate message requests (multicast or unicast) to a given topic and waits for responses.

Usage: lbmireq [-hv] [-c filename] [-l len] [-L linger] [-P sec] [-r rate/pct] [-R requests] [-T target] [topic]

-c filename = Use LBM configuration file filename.

Multiple config files are allowed.

Example: '-c file1.cfg -c file2.cfg'

-h = help

-l len = send messages of len bytes

-L linger = linger for linger seconds before closing context

-P sec = pause sec seconds after sending request for responses to arrive

-r [UM]DATA/RETR = Set transport type to LBT-R[UM], set data rate limit to DATA bits per second, and set retransmit rate limit to RETR bits per second. For both limits, the optional k, m, and g suffixes may be used. For example, '-r 1m/500k' is the same as '-r 1000000/500000'

-R requests = number of request messages to send

-T target = send immediate request to target

-v = be verbose (-v -v = be even more verbose)

### 1.7.8 Example lbmlatping.c

Example build:

```
gcc lbmlatping.c stats.c -o lbmlatping -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
  -lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmlatping.c](#)

Purpose: application to measure round-trip latency of SMX. Use with lbmlatpong.

Usage: lbmlatping [-h] [-c filename] [-l len] [-P msec]

-c filename = Use LBM configuration file filename.

Multiple config files are allowed.

Example: '-c file1.cfg -c file2.cfg'

-h = help

-l len = use len length messages

-P usec = pause after each send usec microseconds

(only accurate to milliseconds on windows)

### 1.7.9 Example lbmlatpong.c

Example build:

```
gcc lbmlatpong.c stats.c -o lbmlatpong -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmlatpong.c](#)

Purpose: application to measure round-trip latency of SMX. Use with lbmlatping.

Usage: lbmlatpong [-h] [-c filename]

```
-c filename = Use LBM configuration file filename.
              Multiple config files are allowed.
              Example: '-c file1.cfg -c file2.cfg'

-h = help
```

### 1.7.10 Example lbmmon.c

Example build:

```
gcc lbmmon.c -o lbmmon -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread -llbm
-lm -lrsock -lprotobuf-c
gcc lbmmon_cmd.c -o lbmmon_cmd -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmmon.c](#)

Purpose: example LBM statistics monitoring application.

Usage: lbmmon [options]

Available options:

```
-c, --config=FILE      Use LBM configuration file FILE.
                       Multiple config files are allowed.
                       Example: '-c file1.cfg -c file2.cfg'

-h, --help             display this help and exit

-t, --transport=TRANS  use transport module TRANS
                       TRANS may be 'lbm', 'udp', or 'lbmsnmp', default is
                       'lbm'

--transport-opts=OPTS use OPTS as transport module options
                       See the 'UM Operations Guide' section 'Monitoring
                       Transport Modules'.

-f, --format=FMT       use format module FMT
                       FMT may be 'csv' or 'pb'

--format-opts=OPTS     use OPTS as format module options
                       See the 'UM Operations Guide' section 'Monitoring
                       Format Modules'.
```

Transport and format options are passed as name=value pairs, separated by a semicolon.

The entire option string should be enclosed in double-quotes.

LBM transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```
config=FILE          use LBM configuration file FILE
topic=TOPIC          receive statistics on topic TOPIC
                    default is /29west/statistics
```

```
wctopic=PATTERN      receive statistics on wildcard topic PATTERN
                      See https://communities.informatica.com/infakb/faq/5/Pages/
                      80075.aspx
                      for guidelines on using wildcard topics. Also make sure
                      the statistics
                      topic namespace is disjoint from the data topic namespace.
```

LBMSNMP transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```
config=FILE          use LBM configuration file FILE
topic=TOPIC          receive statistics on topic TOPIC
                    default is /29west/statistics
wctopic=PATTERN      receive statistics on wildcard topic PATTERN
                      See https://communities.informatica.com/infakb/faq/5/Pages/
                      80075.aspx
                      for guidelines on using wildcard topics. Also make sure
                      the statistics
                      topic namespace is disjoint from the data topic namespace.
```

UDP transport options:

```
port=NUM             receive on UDP port NUM
                    default is 2933
interface=IP         receive multicast on interface IP
                    default is INADDR_ANY (0.0.0.0)
mcgroup=GRP         receive on multicast group GRP
```

CSV format options:

```
separator=CHAR      separate CSV fields with character CHAR
                    defaults to ','
                    Don't use a semicolon!
passthrough=VAL     VAL may be 'off', 'on' or 'convert'
                    defaults to 'off'
```

PB format options:

```
passthrough=VAL     VAL may be 'off', 'on' or 'convert'
                    defaults to 'off'
```

### 1.7.11 Example lbmmoncache.c

Example build:

```
gcc lbmmoncache.c -o lbmmoncache -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmmoncache.c](#)

Purpose: example LBM statistics monitoring application.

Usage: lbmmoncache [options]

Available options:

```
-c, --config=FILE    Use LBM configuration file FILE.
                    Multiple config files are allowed.
                    Example: '-c file1.cfg -c file2.cfg'
-C, --cache-size=size Set the cache size to 'size' entries
-h, --help           display this help and exit
-t, --transport=TRANS use transport module TRANS
```

```

                                TRANS may be 'lbm', 'udp', or 'lbmsnmp', default is
                                'lbm'
--transport-opts=OPTS use OPTS as transport module options
-f, --format=FMT      use format module FMT
                                FMT may be 'csv'
--format-opts=OPTS   use OPTS as format module options

```

Transport and format options are passed as name=value pairs, separated by a semicolon.

The entire option string should be enclosed in double-quotes.

#### LBM transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE          use LBM configuration file FILE
topic=TOPIC          receive statistics on topic TOPIC
                                default is /29west/statistics
wctopic=PATTERN      receive statistics on wildcard topic PATTERN
                                See https://communities.informatica.com/infakb/faq/5/Pages/80075.aspx
                                for guidelines on using wildcard topics. Also make sure
                                the statistics
                                topic namespace is disjoint from the data topic namespace.

```

#### LBMSNMP transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE          use LBM configuration file FILE
topic=TOPIC          receive statistics on topic TOPIC
                                default is /29west/statistics
wctopic=PATTERN      receive statistics on wildcard topic PATTERN
                                See https://communities.informatica.com/infakb/faq/5/Pages/80075.aspx
                                for guidelines on using wildcard topics. Also make sure
                                the statistics
                                topic namespace is disjoint from the data topic namespace.

```

#### UDP transport options:

```

port=NUM             receive on UDP port NUM
                                default is 2933
interface=IP         receive multicast on interface IP
                                default is INADDR_ANY (0.0.0.0)
mcgroup=GRP          receive on multicast group GRP

```

#### CSV format options:

```

separator=CHAR       separate CSV fields with character CHAR
                                defaults to ','
                                Don't use a semicolon!
passthrough=VAL      VAL may be 'off', 'on' or 'convert'
                                defaults to 'off'

```

#### PB format options:

```

passthrough=VAL      VAL may be 'off', 'on' or 'convert'
                                defaults to 'off'

```

### 1.7.12 Example lbmmon\_cmd.c

Example build:

```
gcc lbmmon_cmd.c -o lbmmon_cmd -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmmon\\_cmd.c](#)

Purpose: send unicast immediate control requests to an LBMMON publisher.

Usage: lbmmon\_cmd -T target\_string -C command [options]

Available options:

```
-c filename = Use LBM configuration file filename.
              Multiple config files are allowed.
              Example: '-c file1.cfg -c file2.cfg'
-C command  = command to send [required]
-d          = dump default filter options to stdout
-D data     = data for command (either -D or -F, not both)
              Example: '-C SET_INTERVAL -D 30'
-F filename = filename for command (either -D or -F, not both)
              Example: '-C SET_FILTER_OPTIONS -F filter.cfg'
-h          = help
-I id       = Application ID of node for command
              Example: '-C SNAP -N UMESTORE -I storeName'
-L linger   = linger for linger seconds before closing context
-N node     = node type for command
              Example: '-C SNAP -N CONTEXT'
-P sec      = pause for sec seconds after sending request to wait for response
-T target   = target string for unicast immediate requests [required]
```

### 1.7.13 Example lbmmondata.c

Example build:

```
gcc lbmmondata.c -o lbmmondata -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmmondata.c](#)

Purpose: example LBM statistics monitoring application.

Usage: lbmmondata [-c filename] [-t topicname]

```
-c filename = Use LBM configuration file filename.
              Multiple config files are allowed.
              Example: '-c file1.cfg -c file2.cfg'
-t topicname = use topic topicname to receive statistics
```

### 1.7.14 Example lbmmonudp.c

Example build:

```
gcc lbmmonudp.c -o lbmmonudp -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-llbm -lm -lrsock -lprotobuf-c
```

---

Source code: [lbmmonudp.c](#)

Purpose: application that receives LBM statistics and forwards as CSV over UDP.

Usage: lbmmonudp [options]

Available options:

```

-3, --force-32bit      force all data values to fit within 32 bits
                        default is to use native data size
                        applies only to 64-bit platforms
-a, --address=IP       send CSV data to unicast address IP
-b, --broadcast=IP     send CSV data to broadcast address IP
-f, --format=FMT       use monitor format module FMT
                        FMT may be 'csv'
      --format-opts=OPTS use OPTS as format module options
-h, --help             display this help and exit
-i, --interface=IP     send multicast via interface IP
-m, --multicast=GRP    send CSV data to multicast group GRP
-p, --port=NUM         send CSV data on UDP port NUM
                        default is port 1234
-t, --transport=TRANS  use monitor transport module TRANS
                        TRANS may be 'lbm' or 'udp', default is 'lbm'
      --transport-opts=OPTS use OPTS as transport module options
-T, --ttl=NUM          send multicast with TTL NUM
                        default is 1

```

Transport and format options are passed as name=value pairs, separated by a semicolon.

The entire option string should be enclosed in double-quotes.

LBM transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE           use LBM configuration file FILE
topic=TOPIC           receive statistics on topic TOPIC
                        default is /29west/statistics
wctopic=PATTERN       receive statistics on wildcard topic PATTERN
                        See https://communities.informatica.com/infakb/faq/5/Pages/80075.aspx
                        for guidelines on using wildcard topics. Also make sure
                        the statistics
                        topic namespace is disjoint from the data topic namespace.

```

LBMSNMP transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE           use LBM configuration file FILE
topic=TOPIC           receive statistics on topic TOPIC
                        default is /29west/statistics
wctopic=PATTERN       receive statistics on wildcard topic PATTERN
                        See https://communities.informatica.com/infakb/faq/5/Pages/80075.aspx
                        for guidelines on using wildcard topics. Also make sure
                        the statistics
                        topic namespace is disjoint from the data topic namespace.

```

UDP transport options:

```

port=NUM              receive on UDP port NUM
                        default is 2933
interface=IP          receive multicast on interface IP
                        default is INADDR_ANY (0.0.0.0)
mcgroup=GRP           receive on multicast group GRP

```

## CSV format options:

```
separator=CHAR          separate CSV fields with character CHAR
                        defaults to ','
                        Don't use a semicolon!
passthrough=VAL        VAL may be 'off', 'on' or 'convert'
                        defaults to 'off'
```

## PB format options:

```
passthrough=VAL        VAL may be 'off', 'on' or 'convert'
                        defaults to 'off'
```

### 1.7.15 Example lbmmrcv.c

## Example build:

```
gcc lbmmrcv.c verifymsg.c -o lbmmrcv -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmmrcv.c](#)

Purpose: application that receives messages from a set of one or more topics.

Usage: lbmmrcv [options]

```
-B, --bufsize=#        Set receive socket buffer size to # (in MB)
-c, --config=FILE      Use LBM configuration file FILE.
                        Multiple config files are allowed.
                        Example: '-c file1.cfg -c file2.cfg'
-C, --contexts=NUM     use NUM lbm_context_t objects
-E, --exit              exit and end upon receiving End-of-Stream notification
-e, --end-flag=FILE    clean up and exit when file FILE is created
-h, --help              display this help and exit
-i, --initial-topic=NUM use NUM as initial topic number
-o, --regid-offset=offset use offset to calculate Registration ID
                        (as source registration ID + offset)
                        offset of 0 forces creation of regid by store
-L, --linger=NUM       linger for NUM seconds after done
-r, --root=STRING      use topic names with root of STRING
-R, --receivers=NUM    create NUM receivers
-s, --statistics        print statistics along with bandwidth
-v, --verbose           be verbose
-V, --verify           verify message contents
```

## Monitoring options:

```
--monitor-rcv=NUM      monitor receiver every NUM seconds
--monitor-ctx=NUM      monitor context every NUM seconds
--monitor-transport=TRANS use monitor transport module TRANS
                        TRANS may be 'lbm', 'lbmsnmp', or 'udp', default
                        is 'lbm'
--monitor-transport-opts=OPTS use OPTS as transport module options
--monitor-format=FMT    use monitor format module FMT
                        FMT may be 'csv' or 'pb'
--monitor-format-opts=OPTS use OPTS as format module options
--monitor-appid=ID     use ID as application ID string
```

Transport and format options are passed as name=value pairs, separated by a semicolon.

The entire option string should be enclosed in double-quotes.

## LBM transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```
config=FILE          use LBM configuration file FILE
topic=TOPIC          send statistics on topic TOPIC
                     default is /29west/statistics
allow_debug=VAL      VAL may be 'off' or 'on'
                     defaults to 'off'
```

LBM SNMP transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```
config=FILE          use LBM configuration file FILE
topic=TOPIC          send statistics on topic TOPIC
                     default is /29west/statistics
```

UDP transport options:

```
address=IP           send statistics to address IP
port=NUM             send to UDP port NUM
                     default is 2933
mcgroup=GRP          send on multicast group GRP
bcaddress=IP         send statistics to broadcast address IP
ttl=NUM              send multicast statistics with TTL NUM
                     default is 16
```

CSV format options:

```
separator=CHAR       separate CSV fields with character CHAR
                     defaults to ','
                     Don't use a semicolon!
```

PB format options:

```
filters=FILE         use FILE that contains filter options
```

### 1.7.16 Example lbmmrcvq.c

Example build:

```
gcc lbmmrcvq.c -o lbmmrcvq -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-lldns -lm -lrsocket -lprotobuf-c
```

Source code: [lbmmrcvq.c](#)

Purpose: application that receives messages from a set of one or more topics using event queues.

Usage: lbmmrcvq [options]

```
-B, --bufsize=#      Set receive socket buffer size to # (in MB)
-C, --config=FILE    Use LBM configuration file FILE.
                     Multiple config files are allowed.
                     Example: '-c file1.cfg -c file2.cfg'
-C, --contexts=NUM   use NUM lbm_context_t objects
-h, --help           display this help and exit
-i, --initial-topic=NUM use NUM as initial topic number
-L, --linger=NUM     linger for NUM seconds after done
-r, --root=STRING    use topic names with root of STRING
-R, --receivers=NUM  create NUM receivers
-s, --statistics     print statistics along with bandwidth
```



### 1.7.17 Example lbmmreq.c

Example build:

```
gcc lbmmreq.c -o lbmmreq -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-lbbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmmreq.c](#)

Purpose: application that sends request messages to a single topic and processes responses.

Usage: lbmmreq [options] topic

Available options:

```
-c filename = Use LBM configuration file filename.
                Multiple config files are allowed.
                Example: '-c file1.cfg -c file2.cfg topicname'
-d delay = delay sending for delay seconds after source creation
-h = help
-l len = send messages of len bytes
-r rate/pct = send with LBT-RM at rate and retransmission pct%
-R requests = send requests number of requests
-v = be verbose (-v -v = be even more verbose)
```

### 1.7.18 Example lbmmsrc.c

Example build:

```
gcc lbmmsrc.c verifymsg.c -o lbmmsrc -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -lbbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmmsrc.c](#)

Purpose: send messages on multiple topics, optionally by multiple threads.

Topic names generated as a root, a dot, and by an integer.

By default, the first topic created will be '29west.example.multi.0'

Usage: lbmmsrc [options]

Available options:

```
-b, --batch=NUM          send messages in batch sizes of NUM between each pause
-c, --config=FILE       Use LBM configuration file FILE.
                        Multiple config files are allowed.
                        Example: '-c file1.cfg -c file2.cfg'
-d, --delay=NUM         delay sending for delay seconds after source creation
-h, --help              display this help and exit
-i, --initial-topic=NUM use NUM as initial topic number [0]
-j, --late-join=NUM     enable Late Join with specified retention buffer size
                        (in bytes)
-l, --length=NUM        send messages of length NUM bytes
-L, --linger=NUM        linger for NUM seconds after done
-M, --messages=NUM      send maximum of NUM messages
-P, --pause=NUM         pause NUM milliseconds after each send
-r, --root=STRING       use topic names with root of STRING
                        [29west.example.multi]
-R, --rate=[UM]DATA/RETR Set transport type to LBT-R[UM], set data rate limit to
                        DATA bits per second, and set retransmit rate limit to
                        RETR bits per second. For both limits, the optional
                        k, m, and g suffixes may be used. For example,
                        '-R 1m/500k' is the same as '-R 1000000/500000'
-s, --statistics=NUM    print stats every NUM seconds
```

```

-S, --sources=NUM      use NUM sources
-T, --threads=NUM      use NUM threads
-v, --verbose           be verbose
-V, --verifiable_msg  construct verifiable messages

```

Monitoring options:

```

--monitor-src=NUM      monitor source every NUM seconds
--monitor-ctx=NUM      monitor context every NUM seconds
--monitor-transport=TRANS
                        use monitor transport module TRANS
                        TRANS may be 'lbm', 'lbmsnmp', or 'udp', default
                        is 'lbm'
--monitor-transport-opts=OPTS
                        use OPTS as transport module options
--monitor-format=FMT   use monitor format module FMT
                        FMT may be 'csv' or 'pb'
--monitor-format-opts=OPTS
                        use OPTS as format module options
--monitor-appid=ID     use ID as application ID string

```

Transport and format options are passed as name=value pairs, separated by a semicolon.

The entire option string should be enclosed in double-quotes.

LBM transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE           use LBM configuration file FILE
topic=TOPIC           send statistics on topic TOPIC
                        default is /29west/statistics
allow_debug=VAL       VAL may be 'off' or 'on'
                        defaults to 'off'

```

LBMSNMP transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE           use LBM configuration file FILE
topic=TOPIC           send statistics on topic TOPIC
                        default is /29west/statistics

```

UDP transport options:

```

address=IP            send statistics to address IP
port=NUM              send to UDP port NUM
                        default is 2933
mcgroup=GRP          send on multicast group GRP
bcaddress=IP          send statistics to broadcast address IP
ttl=NUM               send multicast statistics with TTL NUM
                        default is 16

```

CSV format options:

```

separator=CHAR        separate CSV fields with character CHAR
                        defaults to ','
                        Don't use a semicolon!

```

PB format options:

```

filters=FILE          use FILE that contains filter options

```

### 1.7.19 Example lbmpong.c

Example build:

```
gcc lbmpong.c -o lbmpong -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-lbbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmpong.c](#)

Purpose: application that measures round trip message latency.

```
Usage: lbmpong [-ChIqRv] [-c filename] [-i msgs] [-l len] [-M msgs] [-P msec] [-r
rate/pct] [-s seed] [-t secs] [-T topic] id
  -c filename = Use LBM configuration file filename.
                Multiple config files are allowed.
                Example: '-c file1.cfg -c file2.cfg'
  -C = collect RTT data
  -h = help
  -i msgs = send and ignore msgs messages to warm up
  -o offset = use offset to calculate Registration ID
                (as source registration ID + offset)
                offset of 0 forces creation of regid by store
  -I = Use MIM
  -l len = use len length messages
  -M msgs = stop after receiving msgs messages
  -P msec = pause after each send msec milliseconds
  -q = use an LBM event queue
  -r [UM]DATA/RETR = Set transport type to LBT-R[UM], set data rate limit to
                DATA bits per second, and set retransmit rate limit to
                RETR bits per second. For both limits, the optional
                k, m, and g suffixes may be used. For example,
                '-r 1m/500k' is the same as '-r 1000000/500000'
  -R = perform RTT measurement per message
  -s seed = init randomization of contents of message payload
  -t secs = run for secs seconds
  -T topic = topic name prefix (appended with '/' and id) [lbmpong]
  -v = be verbose about each message
  id = either 'ping' or 'pong'
```

### 1.7.20 Example lbmprice.c

Example build:

```
gcc lbmprice.c -o lbmprice -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-lbbm -lm -lrsock -lprotobuf-c -llbmsdm -llbmutl
```

Source code: [lbmprice.c](#)

Purpose: simulated price source and receiver for demonstration.

```
Usage: lbmprice -s [-h] [-c filename]
  -c filename = Use LBM configuration file filename.
                Multiple config files are allowed.
                Example: '-c file1.cfg -c file2.cfg'
  -h = help
  -H = act has Hot Failover relay for a price source
  -l pct = induce random receiver loss of pct percent
  -n ms = set receiver NAK generation interval to ms milliseconds
  -s = act as a price source (acts as a receiver by default)
  -t ttl = set resolver (and multicast source) ttl to ttl
```

-v = be verbose

Alternate usage: lbmprice [-h] [-c filename]

-c filename = read config file  
 -h = help  
 -H = use Hot Failover receiver  
 -l pct = induce random receiver loss of pct percent, print max latency  
 -n ms = set receiver NAK generation interval to ms milliseconds  
 -o mode = set ordered delivery mode (1=ordered, 0=arrival order)  
 -t ttl = set resolver (and multicast source) ttl to ttl  
 -v = be verbose

### 1.7.21 Example lbmrcv.c

Example build:

```
gcc lbmrcv.c verifymsg.c -o lbmrcv -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmrcv.c](#)

Purpose: application that receives messages from a given topic.

Usage: lbmrcv [-ACEfhqsSvV] [-c filename] [-r msgs] [-U losslev] topic

Available options:

-A, --ascii	display messages as ASCII text (-A -A = newlines after each msg)
-C, --config=FILE	Use LBM configuration file FILE. Multiple config files are allowed. Example: '-c file1.cfg -c file2.cfg'
-C, --context-stats	fetch context rather than receiver stats
-E, --exit	exit when source stops sending
-f, --failover	use a hot-failover receiver
-h, --help	display this help and exit
-q, --eventq	use an LBM event queue
-r, --msgs=NUM	exit after NUM messages
-O, --orderchecks	Enable message order checking
-N, --channel=NUM	subscribe to channel NUM
-s, --stats=NUM	print LBM statistics every NUM seconds
--max-sources=NUM	allow up to NUM sources (for statistics gathering purposes)
-S, --stop	exit when source stops sending, and print throughput summary
-U, --losslev=NUM	exit after NUM% unrecoverable loss
-v, --verbose	be verbose about incoming messages (-v -v = be even more verbose)
-V, --verify	verify message contents
-G, --datagram=[UM]NUM	Set transport type to LBT-R[UM], set receiver datagram max size to NUM bytes

Monitoring options:

--monitor-rcv=NUM	monitor receiver every NUM seconds
--monitor-ctx=NUM	monitor context every NUM seconds
--monitor-transport=TRANS	use monitor transport module TRANS TRANS may be 'lbm', 'lbmsnmp', or 'udp', default is 'lbm'
--monitor-transport-opts=OPTS	use OPTS as transport module options
--monitor-format=FMT	use monitor format module FMT FMT may be 'csv' or 'pb'
--monitor-format-opts=OPTS	use OPTS as format module options
--monitor-appid=ID	use ID as application ID string

Transport and format options are passed as name=value pairs, separated by a semicolon.

The entire option string should be enclosed in double-quotes.

LBM transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```
config=FILE          use LBM configuration file FILE
topic=TOPIC          send statistics on topic TOPIC
                    default is /29west/statistics
allow_debug=VAL     VAL may be 'off' or 'on'
                    defaults to 'off'
```

LBMSNMP transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```
config=FILE          use LBM configuration file FILE
topic=TOPIC          send statistics on topic TOPIC
                    default is /29west/statistics
```

UDP transport options:

```
address=IP           send statistics to address IP
port=NUM             send to UDP port NUM
                    default is 2933
mcgroup=GRP          send on multicast group GRP
bcaddress=IP         send statistics to broadcast address IP
ttl=NUM              send multicast statistics with TTL NUM
                    default is 16
```

CSV format options:

```
separator=CHAR       separate CSV fields with character CHAR
                    defaults to ','
                    Don't use a semicolon!
```

PB format options:

```
filters=FILE         use FILE that contains filter options
```

## 1.7.22 Example lbmrcvq.c

Example build:

```
gcc lbmrcvq.c verifymsg.c -o lbmrcvq -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmrcvq.c](#)

Purpose: application that receives messages from a given topic using an event queue.

Usage: lbmrcvq [options] topic

Available options:

```
-C, --config=FILE    Use LBM configuration file FILE.
-C, --context-stats  fetch context rather than receiver stats
                    Multiple config files are allowed.
                    Example: '-c file1.cfg -c file2.cfg'
-E, --exit           exit after source ends
```

```

-h, --help          display this help and exit
-r NUM             delete receiver after NUM messages
-s, --stats=NUM    print LBM statistics every NUM seconds
-S, --stop         exit after source ends, print throughput summary
-v, --verbose      be verbose about incoming messages (-v -v = be even more
                  verbose)
-V, --verify       verify message contents

```

## Monitoring options:

```

--monitor-rcv=NUM      monitor receiver every NUM seconds
--monitor-ctx=NUM      monitor context every NUM seconds
--monitor-transport=TRANS
                      use monitor transport module TRANS
                      TRANS may be 'lbm', 'lbmsnmp', or 'udp', default
                      is 'lbm'
--monitor-transport-opts=OPTS
                      use OPTS as transport module options
--monitor-format=FMT  use monitor format module FMT
                      FMT may be 'csv' or 'pb'
--monitor-format-opts=OPTS
                      use OPTS as format module options
--monitor-appid=ID     use ID as application ID string
--monitor-evq=NUM      monitor event queue every NUM seconds

```

Transport and format options are passed as name=value pairs, separated by a semicolon.

The entire option string should be enclosed in double-quotes.

## LBM transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE          use LBM configuration file FILE
topic=TOPIC           send statistics on topic TOPIC
                      default is /29west/statistics
allow_debug=VAL      VAL may be 'off' or 'on'
                      defaults to 'off'

```

## LBMSNMP transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE          use LBM configuration file FILE
topic=TOPIC           send statistics on topic TOPIC
                      default is /29west/statistics

```

## UDP transport options:

```

address=IP           send statistics to address IP
port=NUM             send to UDP port NUM
                      default is 2933
mcgroup=GRP          send on multicast group GRP
bcaddress=IP         send statistics to broadcast address IP
ttl=NUM              send multicast statistics with TTL NUM
                      default is 16

```

## CSV format options:

```

separator=CHAR       separate CSV fields with character CHAR
                      defaults to ','
                      Don't use a semicolon!

```

## PB format options:

```

filters=FILE         use FILE that contains filter options

```

### 1.7.23 Example lbmrcvxsp.c

Example build:

```
gcc lbmrcvxsp.c verifymsg.c -o lbmrcvxsp -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmrcvxsp.c](#)

Purpose: application that receives messages from a given topic, mapping transports to various XSPs.

Usage: lbmrcv [-ACdDEhPrRsSv] [-c filename] [-r msgs] topic

Available options:

-A, --ascii	display messages as ASCII text (-A -A = newlines after each msg)
-c, --config=FILE	Use LBM configuration file FILE. Multiple config files are allowed. Example: '-c file1.cfg -c file2.cfg'
-C, --context-stats	fetch context rather than receiver stats
-d, --defer-xsp-deletion	don't delete xsp until shutdown
-D, --default-xsp	use the default XSP for all transports
-E, --exit	exit when source stops sending
-h, --help	display this help and exit
-P, --round-robin-preallocate	preallocate the XSPs - use with -R
-Q, --sequential-xsp	use sequential mode for XSPs
-r, --msgs=NUM	exit after NUM messages
-R, --round-robin=NUM	use a simple round-robin method for assigning transports to NUM XSPs. (this is the DEFAULT for this application, with a NUM of 3)
-s, --stats=NUM	print LBM statistics every NUM seconds
-S, --stop	exit when source stops sending, and print throughput summary
-v, --verbose	be verbose about incoming messages (-v -v = be even more verbose)
-V, --verify	verify message contents

### 1.7.24 Example lbmreq.c

Example build:

```
gcc lbmreq.c -o lbmreq -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmreq.c](#)

Purpose: application that sends requests on a single topic and waits for responses.

Usage: lbmreq [options] topic

Available options:

-c filename	= Use LBM configuration file filename. Multiple config files are allowed. Example: '-c file1.cfg -c file2.cfg'
-d sec	= delay sending for delay seconds after source creation
-h	= help

```

-l len = send messages of len bytes
-L linger = linger for linger seconds before closing context
-P sec = pause sec seconds after sending request for responses to arrive
-r [UM]DATA/RETR = Set transport type to LBT-R[UM], set data rate limit to
                  DATA bits per second, and set retransmit rate limit to
                  RETR bits per second. For both limits, the optional
                  k, m, and g suffixes may be used. For example,
                  '-r 1m/500k' is the same as '-r 1000000/500000'
-R requests = send requests number of requests
-q = Use Event Queue
-v = be verbose (-v -v = be even more verbose)

```

### 1.7.25 Example lbmresp.c

Example build:

```

gcc lbmresp.c -o lbmresp -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-lldb -lm -lrsock -lprotobuf-c

```

Source code: [lbmresp.c](#)

Purpose: application that receives request messages on a single topic and  
and sends responses back

Usage: lbmresp [-Ehsv] [-c filename] [-l len] [-r responses] [-f topic] topic

-c filename = Use LBM configuration file filename.

Multiple config files are allowed.

Example: '-c file1.cfg -c file2.cfg'

-E = end after end-of-stream

-h = help

-l len = use len bytes for the length of each response

-r responses = send responses messages for each request

-s = be silent about incoming messages

-v = be verbose (-v -v = be even more verbose)

-f = forward request to responders listening on given topic

### 1.7.26 Example lbmresping.c

Example build:

```

gcc lbmresping.c -o lbmresping -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -lldb -lm -lrsock -lprotobuf-c

```

Source code: [lbmresping.c](#)

Purpose: Application that tests the operation of lbm topic resolution by creating  
a source and reporting time required for it to resolve and join the  
source.

Usage: lbmresping [-h] [-c filename] [unicast\_resolver\_host]

-c filename = Use LBM configuration file filename.

Multiple config files are allowed.

Example: '-c file1.cfg -c file2.cfg'

-h = help

### 1.7.27 Example lbmrespq.c

Example build:

```
gcc lbmrespq.c -o lbmrespq -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-lbbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmrespq.c](#)

Purpose: application that receives request messages on a single topic and sends responses back, using an event queue.

```
Usage: lbmrespq [-hs] [-c filename] [-r msgs] topic
-c filename = Use LBM configuration file filename.
             Multiple config files are allowed.
             Example: '-c file1.cfg -c file2.cfg'
-h = help
-P msec = pause msec milliseconds before sending response
-r msgs = delete receiver after msgs request messages
-s = be silent about requests/sec rate
-v = be verbose (-v -v = be even more verbose)
```

### 1.7.28 Example lbmspike.c

Example build:

```
gcc lbmspike.c -o lbmspike -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-lbbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmspike.c](#)

Purpose: application that generates & receives message spikes for performance testing.

```
Usage: lbmspike -R [-dhq] [-c filename] [-o ord] [-u bufsiz] [topic]
-c filename = Use LBM configuration file filename.
             Multiple config files are allowed.
             Example: '-c file1.cfg -c file2.cfg'
-d = dump message time stamps to a file
-h = help
-o ord = set receiver ordered delivery to ord
-q = process received messages on an event queue
-R = role is receiver (default role is source)
-u bufsiz = UDP buffer size for LBT-RM
```

```
Alternate usage: lbmspike [-dhLn] [-B bghumms] [-c filename] [-l len] [-M msgs] [-r
rate/pct] [-v recovms] [topic]
-B bghumms = milliseconds between "background hum" messages
-c filename = read config file filename
-d = dump message time stamps to a file
-h = help
-l len = use len length messages
-L = use TCP-LB
-M msgs = stop after receiving msgs messages
-n = use non-blocking writes
-r [UM]DATA/RETR = Set transport type to LBT-R[UM], set data rate limit to
DATA bits per second, and set retransmit rate limit to
RETR bits per second. For both limits, the optional
k, m, and g suffixes may be used. For example,
'-r 1m/500k' is the same as '-r 1000000/500000'
-v recovms = milliseconds after spike to allow for recovery
```

## 1.7.29 Example lbmsrc.c

Example build:

```
gcc lbmsrc.c verifymsg.c -o lbmsrc -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmsrc.c](#)

Purpose: application that sends to a single topic as fast possible.

Usage: lbmsrc [options] topic

Available options:

-c, --config=FILE	Use LBM configuration file FILE. Multiple config files are allowed. Example: '-c file1.cfg -c file2.cfg'
-d, --delay=NUM	delay sending for NUM seconds after source creation
-h, --help	display this help and exit
-j, --late-join=NUM (in bytes)	enable Late Join with specified retention buffer size
-l, --length=NUM	send messages of NUM bytes
-L, --linger=NUM	linger for NUM seconds before closing context
-M, --messages=NUM	send NUM messages
-n, --non-block	use non-blocking I/O
-N, --channel=NUM	send on channel NUM
-P, --pause=NUM	pause NUM milliseconds after each send
-R, --rate=[UM]DATA/RETR	Set transport type to LBT-R[UM], set data rate limit to DATA bits per second, and set retransmit rate limit to RETR bits per second. For both limits, the optional k, m, and g suffixes may be used. For example, '-R 1m/500k' is the same as '-R 1000000/500000'
-s, --statistics=NUM	print statistics every NUM seconds
-v, --verbose	be verbose about each message
-V, --verifiable	construct verifiable messages
-G, --datagram=[UM]NUM	Set transport type to LBT-R[UM], set source datagram max size to NUM bytes

Monitoring options:

--monitor-src=NUM	monitor source every NUM seconds
--monitor-ctx=NUM	monitor context every NUM seconds
--monitor-transport=TRANS	use monitor transport module TRANS TRANS may be 'lbm', 'lbmsnmp', or 'udp', default is 'lbm'
--monitor-transport-opts=OPTS	use OPTS as transport module options
--monitor-format=FMT	use monitor format module FMT FMT may be 'csv' or 'pb'
--monitor-format-opts=OPTS	use OPTS as format module options
--monitor-appid=ID	use ID as application ID string

Transport and format options are passed as name=value pairs, separated by a semicolon.

The entire option string should be enclosed in double-quotes.

LBM transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

config=FILE	use LBM configuration file FILE
topic=TOPIC	send statistics on topic TOPIC

```

allow_debug=VAL          default is /29west/statistics
                        VAL may be 'off' or 'on'
                        defaults to 'off'

```

#### LBSNMP transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE             use LBM configuration file FILE
topic=TOPIC             send statistics on topic TOPIC
                        default is /29west/statistics

```

#### UDP transport options:

```

address=IP              send statistics to address IP
port=NUM                send to UDP port NUM
                        default is 2933
mcgroup=GRP            send on multicast group GRP
bcaddress=IP           send statistics to broadcast address IP
ttl=NUM                send multicast statistics with TTL NUM
                        default is 16

```

#### CSV format options:

```

separator=CHAR         separate CSV fields with character CHAR
                        defaults to ','
                        Don't use a semicolon!

```

#### PB format options:

```

filters=FILE           use FILE that contains filter options

```

### 1.7.30 Example lbmssrc.c

#### Example build:

```

gcc lbmssrc.c verifymsg.c -o lbmssrc -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c

```

#### Source code: [lbmssrc.c](#)

Purpose: application that uses Smart Source to send to a single topic.

Usage: lbmssrc [options] topic

#### Available options:

```

-a, --available-data-space  print the length of available data space
-b, --user-supplied-buffer  send messages using a user-supplied buffer
-c, --config=FILE          Use LBM configuration file FILE.
                           Multiple config files are allowed.
                           Example: '-c file1.cfg -c file2.cfg'
-d, --delay=NUM            delay sending for NUM seconds after source creation
-h, --help                 display this help and exit
-i, --int-mprop=VAL,KEY    send integer message property value VAL with name KEY
-j, --late-join=NUM        enable Late Join with specified retention buffer count
-l, --length=NUM           send messages of NUM bytes
                           NOTE: set LBM_SMART_SOURCE_CHECK=0xffffffff env
                           variable
                           to check if the maximum message length is
                           exceeded
-L, --linger=NUM           linger for NUM seconds before closing context
-M, --messages=NUM         send NUM messages
-N, --channel=NUM          send on channel NUM

```

```

-S, --perf-stats=NUM,OT    print performance stats every NUM messages sent
                           If optional OT is given, override the default 10 usec
                           Outlier Threshold
-P, --pause=NUM            pause NUM milliseconds after each send
-s, --statistics=NUM       print statistics every NUM seconds
-v, --verbose              be verbose; add per message data
-V, --verifiable           construct verifiable messages

```

## Monitoring options:

```

--monitor-src=NUM          monitor source every NUM seconds
--monitor-ctx=NUM         monitor context every NUM seconds
--monitor-transport=TRANS use monitor transport module TRANS
                           TRANS may be 'lbm', 'lbmsnmp', or 'udp', default
                           is 'lbm'
--monitor-transport-opts=OPTS use OPTS as transport module options
--monitor-format=FMT      use monitor format module FMT
                           FMT may be 'csv' or 'pb'
--monitor-format-opts=OPTS use OPTS as format module options
--monitor-appid=ID        use ID as application ID string

```

Transport and format options are passed as name=value pairs, separated by a semicolon.

The entire option string should be enclosed in double-quotes.

## LBM transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE               use LBM configuration file FILE
topic=TOPIC               send statistics on topic TOPIC
                           default is /29west/statistics
allow_debug=VAL           VAL may be 'off' or 'on'
                           defaults to 'off'

```

## LBMSNMP transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE               use LBM configuration file FILE
topic=TOPIC               send statistics on topic TOPIC
                           default is /29west/statistics

```

## UDP transport options:

```

address=IP                send statistics to address IP
port=NUM                  send to UDP port NUM
                           default is 2933
mcgroup=GRP              send on multicast group GRP
bcaddress=IP             send statistics to broadcast address IP
ttl=NUM                  send multicast statistics with TTL NUM
                           default is 16

```

## CSV format options:

```

separator=CHAR            separate CSV fields with character CHAR
                           defaults to ','
                           Don't use a semicolon!

```

## PB format options:

```

filters=FILE              use FILE that contains filter options

```

### 1.7.31 Example lbmssrcreq.c

Example build:

```
gcc lbmssrcreq.c -o lbmssrcreq -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmssrcreq.c](#)

Purpose: application that sends requests on a single topic and waits for responses.

Usage: lbmssrcreq [options] topic

Available options:

```
-a, --available-data-space  print the length of available data space
-b, --user-supplied-buffer  send messages using a user-supplied buffer
-c filename = Use LBM configuration file filename.
                        Multiple config files are allowed.
                        Example: '-c file1.cfg -c file2.cfg'
-d sec = delay sending for delay seconds after source creation
-h = help
-l len = send messages of len bytes
-L linger = linger for linger seconds before closing context
-P sec = pause sec seconds after sending request for responses to arrive
-r [UM]DATA/RETR = Set transport type to LBT-R[UM], set data rate limit to
                  DATA bits per second, and set retransmit rate limit to
                  RETR bits per second. For both limits, the optional
                  k, m, and g suffixes may be used. For example,
                  '-r 1m/500k' is the same as '-r 1000000/500000'
-R requests = send requests number of requests
-v = be verbose (-v -v = be even more verbose)
```

### 1.7.32 Example lbmstrm.c

Example build:

```
gcc lbmstrm.c verifymsg.c -o lbmstrm -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmstrm.c](#)

Purpose: application that sends messages to one or more topics at a specified rate.

Usage: lbmstrm [options]

Topic names generated as a root, followed by a dot, followed by an integer.

By default, the first topic created will be '29west.example.multi.0'

Available options:

```
-c, --config=FILE          Use LBM configuration file FILE.
                        Multiple config files are allowed.
                        Example: '-c file1.cfg -c file2.cfg'
-h, --help                display this help and exit
-H, --hf                  Use hot failover sources
-i, --initial-topic=NUM  use NUM as initial topic number [0]
-j, --late-join=NUM      enable Late Join with specified retention buffer size
                        (in bytes)
-l, --length=NUM         send messages of length NUM bytes [25]
-L, --linger=NUM         linger for NUM seconds after done [10]
-m, --message-rate=NUM  send at NUM messages per second [10000]
-M, --messages=NUM      send maximum of NUM messages [10000000]
```



```

-D, --deregister      Send Deregistration after receiving 1000 messages
-E, --exit            exit after source ends
-h, --help            display this help and exit
-q                    Use event queue
-r NUM                delete receiver after NUM messages
-N, --channel=NUM     subscribe to channel NUM
-s, --statistics      print statistics along with bandwidth
-v, --verbose         be verbose about incoming messages (-v -v = be even more
                    verbose)

```

## Monitoring options:

```

--monitor-ctx=NUM      monitor context every NUM seconds
--monitor-transport=TRANS
                        use monitor transport module TRANS
                        TRANS may be 'lbm', 'lbmsnmp', or 'udp', default
                        is 'lbm'
--monitor-transport-opts=OPTS
                        use OPTS as transport module options
--monitor-format=FMT   use monitor format module FMT
                        FMT may be 'csv' or 'pb'
--monitor-format-opts=OPTS
                        use OPTS as format module options
--monitor-appid=ID     use ID as application ID string

```

Transport and format options are passed as name=value pairs, separated by a semicolon.

The entire option string should be enclosed in double-quotes.

## LBM transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE           use LBM configuration file FILE
topic=TOPIC            send statistics on topic TOPIC
                        default is /29west/statistics
allow_debug=VAL       VAL may be 'off' or 'on'
                        defaults to 'off'

```

## LBMSNMP transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE           use LBM configuration file FILE
topic=TOPIC            send statistics on topic TOPIC
                        default is /29west/statistics

```

## UDP transport options:

```

address=IP            send statistics to address IP
port=NUM              send to UDP port NUM
                        default is 2933
mcgroup=GRP           send on multicast group GRP
bcaddress=IP          send statistics to broadcast address IP
ttl=NUM               send multicast statistics with TTL NUM
                        default is 16

```

## CSV format options:

```

separator=CHAR        separate CSV fields with character CHAR
                        defaults to ','
                        Don't use a semicolon!

```

## PB format options:

```

filters=FILE          use FILE that contains filter options

```

### 1.7.35 Example lbmwrvcq.c

Example build:

```
gcc lbmwrvcq.c -o lbmwrvcq -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-lbbm -lm -lrsock -lprotobuf-c
```

Source code: [lbmwrvcq.c](#)

Purpose: application that receives messages from a wildcard receiver, using an event queue.

Usage: lbmwrvcq [options] pattern

Available options:

-c, --config=FILE	Use LBM configuration file FILE. Multiple config files are allowed. Example: '-c file1.cfg -c file2.cfg'
-E, --exit	exit after source ends
-h, --help	display this help and exit
-r NUM	delete receiver after NUM messages
-s, --statistics	print statistics along with bandwidth
-v, --verbose	be verbose about incoming messages (-v -v = be even more verbose)

Monitoring options:

--monitor-ctx=NUM	monitor context every NUM seconds
--monitor-transport=TRANS	use monitor transport module TRANS TRANS may be 'lbm', 'lbmsnmp', or 'udp', default is 'lbm'
--monitor-transport-opts=OPTS	use OPTS as transport module options
--monitor-format=FMT	use monitor format module FMT FMT may be 'csv' or 'pb'
--monitor-format-opts=OPTS	use OPTS as format module options
--monitor-appid=ID	use ID as application ID string
--monitor-evq=NUM	monitor event queue every NUM seconds

Transport and format options are passed as name=value pairs, separated by a semicolon.

The entire option string should be enclosed in double-quotes.

LBM transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

config=FILE	use LBM configuration file FILE
topic=TOPIC	send statistics on topic TOPIC default is /29west/statistics
allow_debug=VAL	VAL may be 'off' or 'on' defaults to 'off'

LBMSNMP transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

config=FILE	use LBM configuration file FILE
topic=TOPIC	send statistics on topic TOPIC default is /29west/statistics

UDP transport options:

---

```
address=IP          send statistics to address IP
port=NUM           send to UDP port NUM
                  default is 2933
mcgroup=GRP       send on multicast group GRP
baddress=IP       send statistics to broadcast address IP
ttl=NUM           send multicast statistics with TTL NUM
                  default is 16

CSV format options:
separator=CHAR    separate CSV fields with character CHAR
                  defaults to ','
                  Don't use a semicolon!

PB format options:
filters=FILE      use FILE that contains filter options
```

### 1.7.36 Example minrcv.c

Example build:

```
gcc minrcv.c -o minrcv -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread -llbm
-lm -lrsock -lprotobuf-c
```

Source code: [minrcv.c](#)

minrcv.c: minimal application that receives messages from a given topic.

### 1.7.37 Example minrcv.cpp

Example build:

Source code: [minrcv.cpp](#)

minimal C++ application that receives messages from a given topic.

### 1.7.38 Example minsrc.c

Example build:

```
gcc minsrc.c -o minsrc -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread -llbm
-lm -lrsock -lprotobuf-c
```

Source code: [minsrc.c](#)

minsrc.c: minimal application that sends to a given topic.

---

### 1.7.39 Example srs\_cmd.c

Example build:

```
gcc srs_cmd.c srs_cmd_msg.c -o srs_cmd -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [srs\\_cmd.c](#)

Purpose: send unicast immediate command messages to an SRS daemon.

Usage: srs\_cmd [options] [command\_string]

Available options:

```
-c filename = Use LBM configuration file filename.
                Multiple config files are allowed.
                Example: '-c file1.cfg -c file2.cfg'

-h = help
-L linger = linger for linger seconds before closing context
-T target = target for unicast immediate messages (required)
*****
*      help (print this message): h                               *
*      quit (exit application): q                                 *
*      report SRS version: version                               *
* set category publishing interval: srs_stats 0 | 200-N          *
*                                     um_client_stats 0 | 200-N    *
*                                     connection_events 0 | 200-N *
*                                     srs_error_stats 0 | 200-N     *
*                                     um_client_error_stats 0 | 200-N *
*                                     config_opts 0 | 200-N        *
*                                     internal_config_opts 0 | 200-N *
* set all publishing intervals: interval 0 | 200-N              *
*      snapshot category: snap srs_stats | um_client_stats |    *
*                                     connection_events | srs_error_stats |
*                                     um_client_error_stats | config_opts |
*                                     internal_config_opts         *
*      snapshot all categories: snap                             *
*****
```

### 1.7.40 Example srs\_monitor\_info\_receiver.c

Example build:

```
gcc srs_monitor_info_receiver.c -o srs_monitor_info_receiver -I$LBM/include -I$LBM/
include/lbm -L$LBM/lib -lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [srs\\_monitor\\_info\\_receiver.c](#)

Purpose: receive SRS monitor info messages on the specified topic.

Note: this tool is deprecated.

See <https://ultramessaging.github.io/currdoc/doc/ChangeLog/deprecations.html#deprecationsfor615>

Usage: srs\_monitor\_info\_receiver [options] topic

Available options:

```
-c, --config=FILE      Use LBM configuration file filename.
                        Multiple config files are allowed.
                        Example: '-c file1.cfg -c file2.cfg'

-E, --exit             exit upon EOS reception
-h, --help            help
-L, --linger          linger for linger seconds before closing context
```

### 1.7.41 Example `tnwgdcmd.c`

Example build:

```
gcc tnwgdcmd.c -o tnwgdcmd -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-lldb -lm -lrsock -lprotobuf-c
```

Source code: [tnwgdcmd.c](#)

Purpose: application sends unicast immediate command messages to a `tnwgd` publishing daemon.

Usage: `tnwgdcmd -T target_string -c config_file [command_string]`

Available options:

```
-c filename = Use LBM configuration file filename.
              Multiple config files are allowed.
              Example: '-c file1.cfg -c file2.cfg'
```

```
-h = help
```

```
-L linger = linger for linger seconds before closing context
```

```
-T target = target for unicast immediate messages (mandatory)
```

```
*****
* help (print this message): h *
* quit (exit application): q *
* set publishing interval: (0-N = interval in seconds) *
*          ri 0-N          (routing info) *
*          gcfg 0-N        (gateway config) *
*          ["portal name"] pcfg 0-N      (portal config) *
*          ["portal name"] pstat 0-N     (portal stats) *
*          mallinfo 0-N    (malloc info) *
* * * * *
* snapshot all groups (and all portals) : snap *
* snapshot single group: snap (ri|gcfg|pcfg|pstat|mallinfo) *
* snapshot single portal: "portal name" snap pcfg|pstat *
* Print the current version of the monitor: version *
*****
```

### 1.7.42 Example `tnwgdmon.c`

Example build:

```
gcc tnwgdmon.c -o tnwgdmon -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-lldb -lm -lrsock -lprotobuf-c
```

Source code: [tnwgdmon.c](#)

Purpose: application that receives DRO daemon messages on the specified publishing topic.

Usage: `tnwgdmon [-Ehv] [-c filename] publishing_topic`

Available options:

```
-c, --config=FILE    Use LBM configuration file FILE.
                     Multiple config files are allowed.
                     Example: '-c file1.cfg -c file2.cfg'
```

```
-E, --exit           exit when source stops sending
```

```
-h, --help           display this help and exit
```

```
-v, --verbose        be verbose about incoming messages (-v -v = be even more
verbose)
```

### 1.7.43 Example umedcmd.c

Example build:

```
gcc umedcmd.c -o umedcmd -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-lldb -lm -lrsock -lprotobuf-c
```

Source code: [umedcmd.c](#) See **umedcmd Man Page** for usage information.

### 1.7.44 Example umedmon.c

Example build:

```
gcc umedmon.c -o umedmon -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread
-lldb -lm -lrsock -lprotobuf-c
```

Source code: [umedmon.c](#)

Purpose: application that receives umestore daemon messages on the specified publishing topic.

Usage: `umedmon [-Ehv] [-c filename] publishing_topic`

Available options:

<code>-c, --config=FILE</code>	Use LBM configuration file FILE. Multiple config files are allowed. Example: <code>'-c file1.cfg -c file2.cfg'</code>
<code>-E, --exit</code>	exit when source stops sending
<code>-h, --help</code>	display this help and exit
<code>-v, --verbose</code>	be verbose about incoming messages ( <code>-v -v = be even more verbose</code> )

### 1.7.45 Example ume-example-rcv-2.c

Example build:

```
gcc ume-example-rcv-2.c -o ume-example-rcv-2 -I$LBM/include -I$LBM/include/lbm
-L$LBM/lib -lpthread -lldb -lm -lrsock -lprotobuf-c
```

Source code: [ume-example-rcv-2.c](#)

`ume-example-rcv-2.c`: - Persistent example receiver program.  
See Persistence Guide document.

### 1.7.46 Example ume-example-rcv-3.c

Example build:

---

```
gcc ume-example-rcv-3.c -o ume-example-rcv-3 -I$LBM/include -I$LBM/include/lbm
-L$LBM/lib -lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [ume-example-rcv-3.c](#)

ume-example-rcv-3.c: - Persistent example receiver program.  
See Persistence Guide document.

### 1.7.47 Example ume-example-rcv.c

Example build:

```
gcc ume-example-rcv.c -o ume-example-rcv -I$LBM/include -I$LBM/include/lbm -L$LBM/
lib -lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [ume-example-rcv.c](#)

ume-example-rcv.c: - Persistent example receiver program.  
See Persistence Guide document.

### 1.7.48 Example ume-example-src-2.c

Example build:

```
gcc ume-example-src-2.c -o ume-example-src-2 -I$LBM/include -I$LBM/include/lbm
-L$LBM/lib -lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [ume-example-src-2.c](#)

ume-example-src-2.c: - Persistent example source program.  
See Persistence Guide document.

### 1.7.49 Example ume-example-src-3.c

Example build:

```
gcc ume-example-src-3.c -o ume-example-src-3 -I$LBM/include -I$LBM/include/lbm
-L$LBM/lib -lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [ume-example-src-3.c](#)

ume-example-src-3.c: - Persistent example source program.  
See Persistence Guide document.

---

### 1.7.50 Example ume-example-src.c

Example build:

```
gcc ume-example-src.c -o ume-example-src -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [ume-example-src.c](#)

ume-example-src.c: - Persistent example receiver program.  
See Persistence Guide document.

### 1.7.51 Example umercv.c

Example build:

```
gcc umercv.c verifymsg.c -o umercv -I$LBM/include -I$LBM/include/lbm -L$LBM/lib -lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [umercv.c](#)

Purpose: application that receives persisted messages from a given topic.

Usage: umercv [options] topic

Available options:

-A, --ascii	display messages as ASCII text (-A -A for newlines after each msg)
-c, --config=FILE	Use LBM configuration file FILE. Multiple config files are allowed. Example: '-c file1.cfg -c file2.cfg'
-D, --deregister=NUM	Deregister the receiver after receiving NUM messages
-E, --exit	exit after source ends
-e, --explicit-ack=N	send an Explicit ACK every N messages
-h, --help	display this help and exit
--max-sources=num	allow num sources (for statistics gathering purposes)
-i, --regid-offset=offset	use offset to calculate Registration ID (as source registration ID + offset) offset of 0 forces creation of regid by store
-N, --seqnum=X	display recovery sequence number info and set low seqnum to low+X
-r, --msgs=NUM	delete receiver after NUM messages
--session-id=NUM	Use NUM as a Session ID rather than using a Registration ID (regid-offset will be ignored)
-s, --statistics=NUM	print statistics every NUM seconds, along with bandwidth
-S, --stop	exit after source ends, print throughput summary
-u, --uregid=num	set User settable Registration ID to num for context
-v, --verbose	be verbose about incoming messages (-v -v = be even more verbose)
-V, --verify	verify message contents
-x, --no-exit-on-reg-error	don't exit on registration error (default is to exit)

### 1.7.52 Example umesnaprepo.c

Example build:

```
gcc umesnaprepo.c -o umesnaprepo -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c -llbmutl -lumestorelib -lrt
-lsmartheap_smp64
```

Source code: [umesnaprepo.c](#) See [umesnaprepo Man Page](#) for usage information.

### 1.7.53 Example umesrc.c

Example build:

```
gcc umesrc.c verifymsg.c -o umesrc -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [umesrc.c](#)

Purpose: application that sends persisted messages to a given topic at a specified rate.

Usage: umesrc [options] topic

Available options:

-c, --config=FILE	Use LBM configuration file FILE. Multiple config files are allowed. Example: '-c file1.cfg -c file2.cfg'
-d, --delay=NUM	delay sending for NUM seconds after source creation
-D, --deregister	deregister the source after sending messages
-h, --help	display this help and exit
-j, --late-join	turn on UME late join
-f, --flight-size=NUM	allow NUM unstabilized messages in flight (determines message rate)
-l, --length=NUM	send messages of NUM bytes
-L, --linger=NUM	linger for NUM seconds before closing context
-M, --messages=NUM	send NUM messages
-m, --message-rate=NUM	send at NUM messages per second if allowed by the flight size setting
-N, --seqnum-info	display sequence number information from source events
-n, --non-block	use non-blocking I/O
-P, --pause=NUM	pause NUM milliseconds after each send
-R, --rate=[UM]DATA/RETR	Set transport type to LBT-R[UM], set data rate limit to DATA bits per second, and set retransmit rate limit to RETR bits per second. For both limits, the optional k, m, and g suffixes may be used. For example, '-R 1m/500k' is the same as '-R 1000000/500000'
-s, --statistics=NUM	print statistics every NUM seconds
-S, --store=IP	use specified UME store
-t, --storename=NAME	use specified UME store
-v, --verbose	print additional info in verbose form
-V, --verifiable	construct verifiable messages

## 1.7.54 Example umessrc.c

Example build:

```
gcc umessrc.c verifymsg.c -o umessrc -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c
```

Source code: [umessrc.c](#)

Purpose: application that uses Smart Source sends to a given topic. Understands persistence.

Usage: `umessrc [options] topic`

Available options:

<code>-a, --available-data-space</code>	print the length of available data space
<code>-b, --user-supplied-buffer</code>	send messages using a user-supplied buffer
<code>-c, --config=FILE</code>	Use LBM configuration file FILE. Multiple config files are allowed. Example: <code>'-c file1.cfg -c file2.cfg'</code>
<code>-d, --delay=NUM</code>	delay sending for NUM seconds after smart source creation
<code>-D, --deregister</code>	deregister the smart source after sending messages
<code>-h, --help</code>	display this help and exit
<code>-i, --int-mprop=VAL,KEY</code>	send integer message property value VAL with name KEY
<code>-j, --late-join</code>	turn on UME late join
<code>-f, --flight-size=NUM</code>	allow NUM unstabilized messages in flight (determines message rate)
<code>-l, --length=NUM</code>	send messages of NUM bytes
<code>-L, --linger=NUM</code>	linger for NUM seconds before closing context
<code>-M, --messages=NUM</code>	send NUM messages
<code>-m, --message-rate=NUM</code>	send at NUM messages per second if allowed by the flight size setting
<code>-N, --channel=NUM</code>	send on channel NUM
<code>-n, --non-block</code>	use non-blocking I/O
<code>-P, --pause=NUM</code>	pause NUM milliseconds after each send
<code>-Q, --seqnum-info</code>	display sequence number information from smart source events
<code>-s, --statistics=NUM</code>	print statistics every NUM seconds
<code>-S, --store=IP</code>	use specified UME store
<code>-t, --storename=NAME</code>	use specified UME store
<code>-v, --verbose</code>	print additional info in verbose form
<code>-V, --verifiable</code>	construct verifiable messages

Monitoring options:

<code>--monitor-src=NUM</code>	monitor source every NUM seconds
<code>--monitor-ctx=NUM</code>	monitor context every NUM seconds
<code>--monitor-transport=TRANS</code>	use monitor transport module TRANS TRANS may be <code>'lbm'</code> , <code>'lbmsnmp'</code> , or <code>'udp'</code> , default is <code>'lbm'</code>
<code>--monitor-transport-opts=OPTS</code>	use OPTS as transport module options
<code>--monitor-format=FMT</code>	use monitor format module FMT FMT may be <code>'csv'</code> or <code>'pb'</code>
<code>--monitor-format-opts=OPTS</code>	use OPTS as format module options
<code>--monitor-appid=ID</code>	use ID as application ID string

Transport and format options are passed as name=value pairs, separated by a semicolon.

The entire option string should be enclosed in double-quotes.

LBM transport options:

Note that individual LBM options can be specified as `<scope>|<option>=value`, where `<scope>` is one of `context`, `source`, `receiver`, `wildcard_receiver`, or `event_queue` and `<option>` is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE          use LBM configuration file FILE
topic=TOPIC          send statistics on topic TOPIC
                    default is /29west/statistics
allow_debug=VAL      VAL may be 'off' or 'on'
                    defaults to 'off'

```

#### LBMSNMP transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

```

config=FILE          use LBM configuration file FILE
topic=TOPIC          send statistics on topic TOPIC
                    default is /29west/statistics

```

#### UDP transport options:

```

address=IP           send statistics to address IP
port=NUM             send to UDP port NUM
                    default is 2933
mcgroup=GRP         send on multicast group GRP
bcaddress=IP        send statistics to broadcast address IP
ttl=NUM             send multicast statistics with TTL NUM
                    default is 16

```

#### CSV format options:

```

separator=CHAR      separate CSV fields with character CHAR
                    defaults to ','
                    Don't use a semicolon!

```

#### PB format options:

```

filters=FILE        use FILE that contains filter options

```

### 1.7.55 Example umestored\_example.c

#### Example build:

```

gcc umestored_example.c -o umestored_example -I$LBM/include -I$LBM/include/lbm
-L$LBM/lib -lpthread -llbm -lm -lrsock -lprotobuf-c -llbmutl -lumestorelib -lrt
-lsmartheap_smp64

```

Source code: [umestored\\_example.c](#)

umestored\_example.c: application that shows how to call the umestored\_main API to start a UMP store daemon.

### 1.7.56 Example umqrcv.c

#### Example build:

```

gcc umqrcv.c verifymsg.c -o umqrcv -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c

```

Source code: [umqrcv.c](#)

---

Purpose: application that receives brokered queuing messages from a single topic.

Usage: umqrcv [options] topic

Available options:

-A, --ascii	display messages as ASCII text (-A -A for newlines after each msg)
-B, --broker=address	use broker given by address
-C, --config=FILE	use FILE as LBM configuration file
-D, --dereg	deregister upon exit
-d, --delay=NUM	delay receiver creation NUM seconds from context creation
-E, --exit	exit after source ends
-h, --help	display this help and exit
-I, --type-id=ID	set Receiver Type ID to ID
--max-sources=num	allow num sources (for statistics gathering purposes)
-r, --msgs=NUM	delete receiver after NUM messages
-s, --statistics=NUM	print statistics every NUM seconds, along with bandwidth
-S, --stop	exit after source ends, print throughput summary
-X, --index	reserve given index if possible, or leave blank to reserve random index
-v, --verbose	be verbose about incoming messages (-v -v = be even more verbose)
-V, --verify	verify message contents

Monitoring options:

--monitor-rcv=NUM	monitor receiver every NUM seconds
--monitor-ctx=NUM	monitor context every NUM seconds
--monitor-transport=TRANS	use monitor transport module TRANS TRANS may be 'lbm', 'lbmsnmp', or 'udp', default is 'lbm'
--monitor-transport-opts=OPTS	use OPTS as transport module options
--monitor-format=FMT	use monitor format module FMT FMT may be 'csv' or 'pb'
--monitor-format-opts=OPTS	use OPTS as format module options
--monitor-appid=ID	use ID as application ID string

Transport and format options are passed as name=value pairs, separated by a semicolon.

The entire option string should be enclosed in double-quotes.

LBM transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

config=FILE	use LBM configuration file FILE
topic=TOPIC	send statistics on topic TOPIC default is /29west/statistics
allow_debug=VAL	VAL may be 'off' or 'on' defaults to 'off'

LBMSNMP transport options:

Note that individual LBM options can be specified as <scope>|<option>=value, where <scope> is one of context, source, receiver, wildcard\_receiver, or event\_queue <option> is the LBM configuration option name

The vertical bar (pipe symbol) is required when specifying individual LBM options.

config=FILE	use LBM configuration file FILE
topic=TOPIC	send statistics on topic TOPIC default is /29west/statistics

UDP transport options:

address=IP	send statistics to address IP
port=NUM	send to UDP port NUM default is 2933
mcgroup=GRP	send on multicast group GRP

```

bcaddress=IP          send statistics to broadcast address IP
ttl=NUM              send multicast statistics with TTL NUM
                    default is 16

```

CSV format options:

```

separator=CHAR       separate CSV fields with character CHAR
                    defaults to ','
                    Don't use a semicolon!

```

PB format options:

```

filters=FILE         use FILE that contains filter options

```

### 1.7.57 Example umqsrc.c

Example build:

```

gcc umqsrc.c verifymsg.c -o umqsrc -I$LBM/include -I$LBM/include/lbm -L$LBM/lib
-lpthread -llbm -lm -lrsock -lprotobuf-c

```

Source code: [umqsrc.c](#)

Purpose: umqsrc.c: application that sends brokered queuing messages to a single topic at a specified rate.

Usage: umqsrc [options] topic

Available options:

```

-A, --appsets=CFG      use ULB Application Sets given by CFG
-B, --broker=address   use broker given by address
-C, --config=FILE      use LBM configuration file FILE
-d, --delay=NUM        delay sending for NUM seconds after source creation
-h, --help             display this help and exit
-f, --flight-size=NUM  allow NUM unstabilized messages in flight (determines
message rate)
-i, --ids              display Message IDs for sent message
-l, --length=NUM       send messages of NUM bytes
-L, --linger=NUM       linger for NUM seconds before closing context
-M, --messages=NUM     send NUM messages
-m, --message-rate=NUM send at NUM messages per second
-N, --seq-num          display sequence number information
-n, --non-block        use non-blocking I/O
-P, --pause=NUM        pause NUM milliseconds after each send
-R, --rate=[UM]DATA/RETR Set transport type to LBT-R[UM], set data rate limit to
DATA bits per second, and set retransmit rate limit to
RETR bits per second. For both limits, the optional
k, m, and g suffixes may be used. For example,
'-R 1m/500k' is the same as '-R 1000000/500000'
-s, --statistics=NUM  print statistics every NUM seconds
-v, --verbose          print additional info in verbose form
-V, --verifiable       construct verifiable messages
-X, --index            Send messages on specified index for ULB sources
-Y, --broker-index    Send messages on specified named index for broker
sources

```

## 1.8 Example Protocol Files

Google protocol buffer definition files. See **Monitoring Formats**.

### 1.8.1 Example dro\_mon.proto

Source code: [dro\\_mon.proto](#)

### 1.8.2 Example srs\_mon.proto

Source code: [srs\\_mon.proto](#)

### 1.8.3 Example um\_mon\_attributes.proto

Source code: [um\\_mon\\_attributes.proto](#)

### 1.8.4 Example um\_mon\_control.proto

Source code: [um\\_mon\\_control.proto](#)

### 1.8.5 Example ump\_mon.proto

Source code: [ump\\_mon.proto](#)

### 1.8.6 Example ums\_mon.proto

Source code: [ums\\_mon.proto](#)

---

