informatica

Informatica Ultra Messaging Persistence Edition (Version 6.7.1)

Configuration Guide

Informatica Ultra Messaging Persistence Edition Configuration Guide

Version 6.7.1
August 2014

Part Number: UMP-CFG-67100-0001

Table of Contents

Preface

The *Ultra Messaging Configuration Guide* is written for Ultra Messaging administrators and application developers. It describes Ultra Messaging core configuration options and how to set them. This guide assumes that you are familiar with Ultra Messaging concepts.

Informatica Resources

Informatica My Support Portal

As an Informatica customer, you can access the Informatica My Support Portal at http://mysupport.informatica.com.

The site contains product information, user group information, newsletters, access to the Informatica customer support case management system (ATLAS), the Informatica How-To Library, the Informatica Knowledge Base, Informatica Product Documentation, and access to the Informatica user community.

Informatica Documentation

The Informatica Documentation team takes every effort to create accurate, usable documentation. If you have questions, comments, or ideas about this documentation, contact the Informatica Documentation team through email at infa_documentation@informatica.com. We will use your feedback to improve our documentation. Let us know if we can contact you regarding your comments.

The Documentation team updates documentation as needed. To get the latest documentation for your product, navigate to Product Documentation from http://mysupport.informatica.com.

Informatica Web Site

You can access the Informatica corporate web site at http://www.informatica.com. The site contains information about Informatica, its background, upcoming events, and sales offices. You will also find product and partner information. The services area of the site includes important information about technical support, training and education, and implementation services.

Informatica How-To Library

As an Informatica customer, you can access the Informatica How-To Library at http://mysupport.informatica.com. The How-To Library is a collection of resources to help you learn more about Informatica products and features. It includes articles and interactive demonstrations that provide

solutions to common problems, compare features and behaviors, and guide you through performing specific real-world tasks.

Informatica Knowledge Base

As an Informatica customer, you can access the Informatica Knowledge Base at http://mysupport.informatica.com. Use the Knowledge Base to search for documented solutions to known technical issues about Informatica products. You can also find answers to frequently asked questions, technical white papers, and technical tips. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team through email at KB_Feedback@informatica.com.

Informatica Support YouTube Channel

You can access the Informatica Support YouTube channel at http://www.youtube.com/user/INFASupport. The Informatica Support YouTube channel includes videos about solutions that guide you through performing specific tasks. If you have questions, comments, or ideas about the Informatica Support YouTube channel, contact the Support YouTube team through email at supportvideos@informatica.com or send a tweet to @INFASupport.

Informatica Marketplace

The Informatica Marketplace is a forum where developers and partners can share solutions that augment, extend, or enhance data integration implementations. By leveraging any of the hundreds of solutions available on the Marketplace, you can improve your productivity and speed up time to implementation on your projects. You can access Informatica Marketplace at http://www.informaticamarketplace.com.

Informatica Velocity

You can access Informatica Velocity at http://mysupport.informatica.com. Developed from the real-world experience of hundreds of data management projects, Informatica Velocity represents the collective knowledge of our consultants who have worked with organizations from around the world to plan, develop, deploy, and maintain successful data management solutions. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Global Customer Support

You can contact a Customer Support Center by telephone or through the Online Support.

Online Support requires a user name and password. You can request a user name and password at http://mysupport.informatica.com.

The telephone numbers for Informatica Global Customer Support are available from the Informatica web site at http://www.informatica.com/us/services-and-training/support-services/global-support-centers/.

CHAPTER 1

Configuring Ultra Messaging Options

This chapter includes the following topics:

Overview

For Ultra Messaging applications, you can set a variety of operational options to customize the application's behavior or performance. You assign values to these options in configuration files or by using function calls. You can assign option values to objects upon or after object creation. Within an object, the implemented option values are referred to as attributes.

Ultra Messaging uses intelligent default values for configuration options, enabling applications to run "out of the box." However, expect to customize Ultra Messaging options to optimize your operating environment. You can use different ways to configure option default and customized value assignments.

Assignment Methods

You can use the following ways to set attributes with configuration options:

XML configuration files

customized defaults in an XML-formatted file, used during object creation

plain text configuration files

customized defaults in a text file, used during object creation

attributes objects

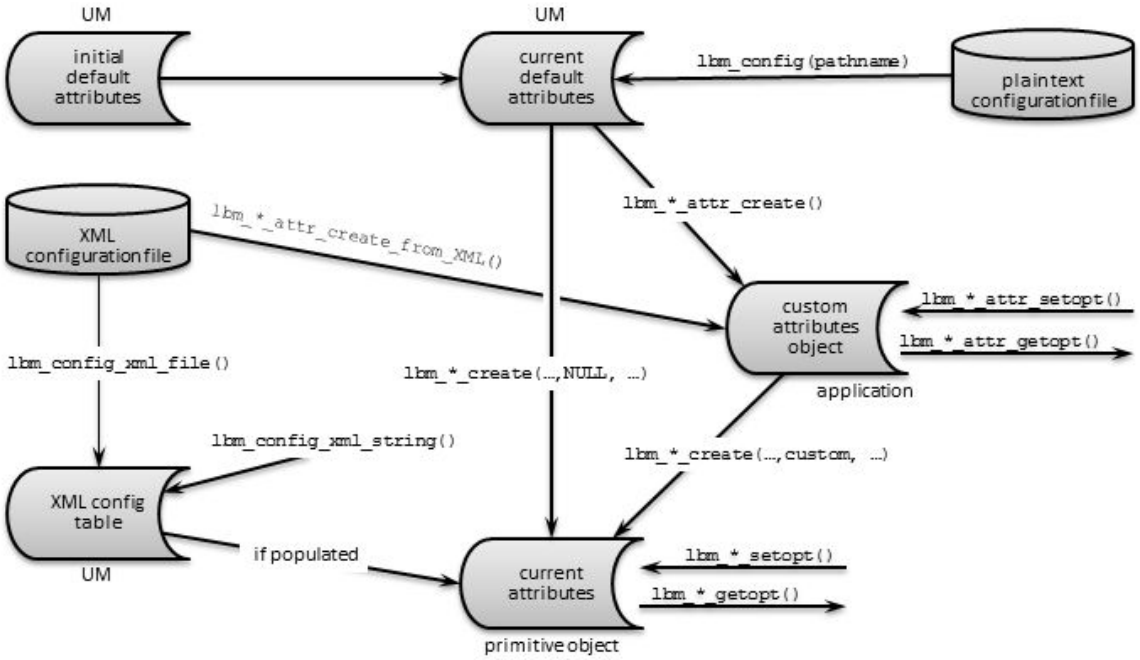
application-specific option values used during object creation

function calls with `lbm[...]setopt()`

used after object creation

The following image shows the different ways Ultra Messaging stores and assigns option values before, during, and after primitive object creation. Primitive objects are sources, receivers, wildcard receivers, event queues, contexts, or HFX objects. The ultimate result is a primitive object with the assigned values residing in current attributes.

Figure 1. Attributes value assignment methods



The *initial default attributes* is the set of factory defaults in Ultra Messaging. Ultra Messaging modifies selected options in the *plain text configuration file*, and then stores these values in *current default attributes*. The *current default attributes* is the starting point for all created primitive objects.

An instantiated primitive object uses values from *current default attributes*, the *XML config table*, and the*custom attributes object*, and then holds the results in *current attributes*.

An *XML configuration file* can pass its setting to an object being created either by directly populating the *XML config table*, or by creating a *custom attributes object*.

Assignment Flow

The above diagram implies, but does not fully explain, the flow of attribute value assignment that UM performs when an application creates a primitive object. This flow is described below, and is important in understanding how and when default values are overridden:

1. If applicable, copy plain text configuration file values to current default attributes.
2. Start creating object.
3. Custom attributes object(s) created/populated (if applicable).
4. If `lbm_*_create()` has a NULL attr, copy *current default attributes* into *current attributes*. Otherwise, copy *custom attributes object* values into *current attributes*.

5. Read applicable options from the *XML config table* into the *current attributes*. Do not overwrite options set with `lbm_config()`, or `lbm_*_attr_setopt()`, which were tagged when modified.
6. Finish object creation.
7. *current attributes* can be changed further (only certain options) via `lbm_*_setopt()`.

Definitions

Before discussing how UM options can be set, some terminology is in order.

- **Option** - A single configuration item that controls some aspect of UM operation. An option typically resides in a configuration file, but can also be assigned a value via a function call. We use options to assign values to an object's attributes.
- **Attribute** - An operational characteristic of an object. An attribute's value is set by an option, hence, there is a one-to-one correspondence between options and attributes. (Note: This use of the term "attribute" is unrelated to, and not to be confused with, "attribute" in XML syntax. In this document, we refer to the latter as "XML attribute".)
- **XML attribute** - See above. In XML syntax, XML attributes are parameters for XML elements.
- **Custom attributes object** A UM object that contains custom attribute values (set by options) for a specific UM object. Separate (and multiple) sets of attributes can exist for each application, though only one can be used when creating a primitive object.
- **Initial default attributes** - The default attributes values built into UM. UM and your applications use these if you have not set any options for the attributes.
- **Primitive object** - Specifically, an object that is a source, receiver, wildcard receiver, event queue, context, or HFX object.
- **Configuration file** - This comes in two types: XML and plain text. Configuration files contain assigned values for options, but the different types are read/copied at different times during the creation of an object.
- **XML config table** - Contains option values that are read from the XML configuration file.
- **Current default attributes** - The attributes values used to create an object in the absence of custom attributes values.
- **Current attributes** - The attribute values for an instantiated UM object that control the current operation of that object.
- **Scope** - The type of object to which an option can apply. Possible scopes are context, source, receiver, wildcard_receiver, event_queue, and hfx.

Which Method Should I Use?

For the four basic assignment methods listed above, following are some scenarios where specific methods are selected.

- To **change a default option value** and apply it to all objects you create, call **lbm_config()** for one or more configuration files. For example, to use LBT-RM rather than TCP for all sources, create a plain text configuration file containing

  ```
  source transport LBTRM
  ```

  and pass its file name to **lbm_config()**.

  Note: The *C API* offers functions **lbm_*_attr_create_default()** to change a current default value back to the initial (factory) default value. No such corresponding method exists for the Java or .NET APIs.

- To **customize specific options before an object is created** for a specific object instance, use a custom attributes object. Also, you can assign XML data to the XML config table directly from your application via **lbm_config_xml_string()**.
- To **create sets of custom values** to be used when creating primitive objects, call **lbm_config_xml_file()** and specify an XML configuration file. This is useful for setting specific default options on a per-topic or per-context basis, which cannot be done with a plain text configuration file. For an example where a sending application uses specific options and values, create an XML configuration file with the application's name (optional) that specifies those options and values. Then pass the XML file name and application name to **lbm_config_xml_file()**.
- To change an option **after an object is created**, modify the current attributes for the object. (Note that many options cannot be changed after an object has been created.)

These methods can be used in combination. Figure 1 on page 2 illustrates the relationships between attributes and the various UM API function calls that affect them.

Configuration Files

There are two types of UM Configuration files:

- “Plain Text Configuration Files” on page 4
- “XML Configuration Files” on page 5

You can read Configuration files either by function call, or automatically upon application launch by specifying a file name in an environment variable. See Attributes value assignment methods and “Assignment Flow” on page 2 for details on how these options replace or override default values.

Plain Text Configuration Files

The plain text configuration file, when invoked, writes option values into UM's current default attributes. These are then read and used in the creation of all objects.

See Chapter 2, “Example Configuration Scenarios” on page 37 for example configuration files.

Reading Plain Text Configuration Files

There are two ways to read a plain text configuration file to set values in current default attributes.

- **API function lbm_config()** - You can call the function multiple times with different file names to set configuration options in phases.

  When you create UM objects (such as a context or receiver), UM sets attributes for that object using the current default attributes. Hence, you must call **lbm_config()** before creating objects (**lbm_*_create()**).
- **Environment variable LBM_DEFAULT_CONFIG_FILE** - reads configuration file when your application is started. You can set this variable to a full pathname or a URL; for example:

  ```
  export LBM_DEFAULT_CONFIG_FILE=/home/lbm/lbtrm.cfg
  ```

  (You can still use the **lbm_config()** function on a different file to make additional changes.)

Plain Text Configuration File Format

A plain text configuration file contains lines that each take the form

scope_keyword option_name option_value

where

scope_keyword - the scope to which the option applies,

option_name - the predefined name for the option, and

option_value - the new value to be assigned to that option.

Allowable values for these parameters are given throughout the rest of this document. Any text following a hash character **#** (also known as a pound sign, number sign, or octothorpe) is interpreted as comment text and is ignored.

For example:

```
# Set transport_tcp_port_low to 4901
context transport_tcp_port_low 4901
# And set transport_tcp_port_high to 4920
context transport_tcp_port_high 4920
```

Note: For plain text configuration files, do not enclose any fields in double quotation marks (").

XML Configuration Files

XML configuration files let you address many different applications and operating requirements, removing the need to programmatically set and reset options for them. A single XML file can contain options for multiple applications. Moreover, for a single application, you can configure multiple named contexts, event queues, etc., with different values for the same options.

See Chapter 2, “Example Configuration Scenarios” on page 37 Chapter 2, “Example Configuration Scenarios” on page 37 for example configuration files.

Reading XML Configuration Files

There are multiple ways to read an XML configuration file to assign values while creating a primitive object.

API function lbm_config_xml_file()

reads an XML configuration file into *XML config table*. Call this before the primitive create function. This does not change the current default attributes. Use a file path, or a URL beginning with `http://` or `ftp://`.

API function lbm_config_xml_string()

populates the XML config table directly from your application. Call this before the primitive create function. This does not change the current default attributes.

API function lbm_*_attr_create_from_XML()

creates a custom attributes object containing the values from an XML configuration file. The values can then be applied to a primitive object being created by calling function **lbm_*_create()** and specifying this custom attributes object in the second parameter.

Environment variable `LBM_XML_CONFIG_FILENAME`

reads the file into the XML config table. These settings are then available to all applications when they start. Use a file path, or a URL beginning with `http://` or `ftp://`.

Environment variable `LBM_XML_CONFIG_APPNAME`

reads options for a specific application from the `LBM_XML_CONFIG_FILENAME` variable's filename. This initiates the specified application's configuration; set this environment variable for every application.

Environment variable `LBM_UMM_INFO`

initiates UMM Daemon to read options for an application and user from the `LBM_XML_CONFIG_FILENAME` variable's filename. Set this variable for every application/user combination, in the following format:

```
export LBM_UMM_INFO=application_name:user_name:password@ip:port
```

Note: Since you can use these API functions and environment variables without the UMM Daemon, you cannot set a username or password.

Using XML Configuration Files With a UM Application

The following procedure describes a general approach to implementing XML configuration files.

1. Create an XML configuration file using an XML editor or text editor. Just for this example, name the file, `UM_CONFIG.XML`.
2. Insert any desired templates in the `<templates>` element to hold configuration option values shared by multiple applications or primitive UM objects (context, source, receiver, wildcard receiver or event queue). You can create and apply multiple templates to applications and primitive UM objects, however, if the same option appears in multiple templates, the option value in the last template overrides the option value in the previous template. See “<templates>” on page 12.
3. Insert an `<application>` element for your UM application in the `<applications>` element and include any relevant templates created in the previous step. Just for this example, name the application, `SENDAPP`. See “<applications>” on page 13.
4. Within the `<Contexts>` element, configure the application's `<Context>` element and context options. And since our example application, SENDAPP is a sending application, also configure its Source options. (If this was a receiving application, you would configure Receiver or Wildcard Receiver options. If your application creates multiple Contexts, enter multiple `<Context>` elements within the Contexts element, inserting the appropriate source, receiver or wildcard receiver options. See “<contexts>” on page 14.
5. Configure the applications Event Queue options. See “<event-queues>” on page 21
6. Save the XML configuration file, `UM_CONFIG.XML`, and load it onto the machine where the application (`SENDAPP`) runs.
7. Set the following environment variables on the machine where `SENDAPP` runs.
   - Set LBM_XML_CONFIG_FILENAME to `UM_CONFIG.XML`.
   - Set LBM_XML_CONFIG_APPNAME to `SENDAPP`.
   - Optionally, you could also use **lbm_config_xml_file(UM_CONFIG.XML,SENDAPP)** in the `SENDAPP` source.
8. Start `SENDAPP`.

XML Configuration File Format

An XML Configuration File follows standard XML conventions. Element declarations or a pointer to a DTD file are not needed, as these are handled by UM.

An XML configuration file generally comprises two primary elements: `templates` and `applications`. Organized and contained within these are option value assignments. Applications containers let you set options for specific applications. To provide more global control over applications, or to simply reduce repetition, you can create templates to hold option settings that are to be used in one or more different applications.

XML configuration files use the high-level structure shown in the following example. This example includes only some container elements, and no options.

```
<um-configuration version="1.0">
  <templates>
    <template name="SENDING">
      <options type="source">
      </options>
      <options type="context">
      </options>
    </template>
   </templates>
  <applications>
    <application name="SENDING-TOPIC1">
      <contexts>
        <context name="SENDING-LBTRM">
          <sources>
            <topic topicname="TOPIC1">
              <options type="source">
              </options>
            </topic>
          </sources>
        </context>
      </contexts>
      <event-queues>
        <event-queue/>
        <event-queue name="EQ-1"/>
      </event-queues>
    </application>
  </applications>
</um-configuration>
```

Merging Multiple XML Configuration Files

For UM XML configuration files and UMP store daemon XML configuration files you can use the XInclude mechanism to merge multiple configuration files.

To include an external file, use the following syntax:

```
<xi:include xmlns:xi=http://www.w3.org/2003/XInclude" href="filename.xml" />
```

Files to be included must be formatted such that all elements are enclosed in a single container element, as shown in the following examples:

Example 1

```
<ume-attributes>
    <option type="store" name="allow-proxy-source" value="1"/>
    <option type="lbm-context" name="resolver_multicast_address" value="239.255.38.0" />
    <option type="lbm-context" name="resolver_multicast_port" value="19999" />
</ume-attributes>
```

Example 2

```
<topics>
  <topic pattern="." type="PCRE">
      <ume-attributes>
         <option type="store" name="repository-type" value="disk"/>
         <option type="store" name="retransmission-request-forwarding" value="0"/>
      </ume-attributes>
  </topic>
</topics>
```

XML Configuration File Elements

Following are descriptions of the XML configuration file elements.

See also "Sample XML Configuration File" on page 25 and "XML Configuration File DTD" on page 27.

<um-configuration>

Description The `<um-configuration>` element is a required container for all UM configuration options residing in the XML configuration file. This is the top-level element.

ParentsNone.

Children

XML Attributes:

XML Attribute	Description	Default Value
`version`	The version of the DTD.	none

Example:

```
<um-configuration version="1.0">
  <applications>
    <application>
      . . .
      . . .
    </application>
  </applications>
</um-configuration>
```

<license>

Description The `<license>` element identifies the UM product license, either as the license key or as a pointer to a license file, as an alternative to setting it in an environment variable.

Parents "<um-configuration>" on page 8

Children. None.

XML Attributes:

XML Attribute	Description	Default Value
`format`	The format for the license element data. `filename` points to the file containing the license key. `string` identifies the data as the license key itself.	`string`
`xml:space`	How whitespace is handled. `default` trims leading and trailing whitespace (e.g., tabs, spaces, linefeeds, etc.), and compresses multiple whitespace characters into a single space character. `preserve` preserves the whitespace exactly as read.	`default`

Example:

```
<um-configuration>
  <license format=filename>
  path/license-file-name
  </license>
  <applications>
    <application>
      . . .
```

<options>

Description The `<options>` element is a container element for individual options. You specify the primitive object in the attribute `type`.

Parents

- "<template>" on page 13
- "<context>" on page 15
- "<topic>" on page 17
- "<wildcard-receiver>" on page 20
- "<event-queue>" on page 22

Children

- "<option>" on page 10
- "<application-data>" on page 24

XML Attributes:

XML Attribute	Description	Default Value
`type`	The type of primitive object, which can be `event-queue`, `context`, `source`, `receiver`, `wildcard-receiver`, or `hfx`).	None

Example:

```
<options type="context">
  <option/>
  . . .
  <application-data/>
</options>
```

<option>

Description The `<option>` element corresponds to any UM configuration option.

Parents

Children "<deny>" on page 12

- "<options>" on page 10
- "<allow>" on page 11

XML Attributes:

XML Attribute	Description	Default Value
`name`	Name of the UM configuration option. See *Reference* for all options.	N/A
`default-value`	The value you are setting for this option.	The default value for the option.
`order`	Permit or restrict particular option values. Valid values are `deny,allow` (deny what you specify, allow everything else) or `allow,deny` (allow what you specify, deny everything else). If using this XML attribute, follow this element with "<allow>" on page 11 or "<deny>" on page 12 elements as needed. See also "Sample XML Configuration File" on page 25.	`deny,allow`

Examples:

To permit any application to choose any transport method except LBT-RU, configure the following in a template included in sending applications.

```
<option default-value="tcp" name="transport" order="deny,allow">
  <deny>LBTRU</deny>
</option>
```

To restrict any application to only the LBT-RM or LBR-RU transport method, configure the following in a template included in sending applications.

```
<option default-value="tcp" name="transport" order="allow,deny">
  <allow>LBTRU</allow>
  <allow>LBTRM</allow>
</option>
```

<allow>

Description Use the `<allow>` element with "<option>" on page 10 to set a condition for that option to permit only a certain subset of possible `default value` values for the option. See also "Using the Order and Rule XML Attributes" on page 26.

Parents "<option>" on page 10

Children. None.

XML Attributes:

XML Attribute	Description	Default Value
`xml:space`	How whitespace is handled. `default` trims leading and trailing whitespace (e.g., tabs, spaces, linefeeds, etc.), and compresses multiple whitespace characters into a single space character. `preserve` preserves the whitespace exactly as read.	`default`

Example:

```
<option default-value="tcp" name="transport" order="allow,deny">
  <allow>LBTRU</allow>
  <allow>LBTRM</allow>
</option>
```

<deny>

Description Use the `<deny>` element with "<option>" on page 10 to set a condition for that option that restricts certain (otherwise) possible `default value` values from being used by the option. See also "Using the Order and Rule XML Attributes" on page 26.

Parents "<option>" on page 10

Children. None.

XML Attributes:

XML Attribute	Description	Default Value
`xml:space`	How whitespace is handled. `default` trims leading and trailing whitespace (e.g., tabs, spaces, linefeeds, etc.), and compresses multiple whitespace characters into a single space character. `preserve` preserves the whitespace exactly as read.	`default`

Example:

```
<option default-value="tcp" name="transport" order="deny,allow">
  <deny>LBTRU</deny>
</option>
```

<templates>

Description The `<templates>` element is a container element for all templates that contain configuration options that can be used in other templates or applications. A template can be very specific, such as configuring options only for LBT-RM sources, or more comprehensive, configuring common options for your applications.

Insert any desired templates in the <templates> element to hold configuration option values shared by multiple applications or primitive objects. You can create and apply multiple templates to applications and primitive UM objects in a comma separated value (CSV) format. However, if the same option appears in multiple templates, the option value in the last or lower-level template overrides the option value in the previous or higher-level template.

Parents "<um-configuration>" on page 8

Children "<template>" on page 13

XML Attributes: None.

Example:

```
<templates>
  <template name="SENDING">
    <options/>
  </template>
</templates>
```

<template>

Description The <template> element is a container for one uniquely named set of options.

Parents "<templates>" on page 12

Children "<options>" on page 10

XML Attributes:

XML Attribute	Description	Default Value
name	Name of the configuration template, which can be referenced elsewhere in this XML configuration file to assign to other configuration elements. Multiple templates can be specified in a comma separated value (CSV) format.	None

Example:

```
<templates>
  <template name="SENDING",name="SENDING-LBTRM">
    <options/>
  </template>
</templates>
```

<applications>

Description The <application> element is a container element for all applications configured in the XML configuration file. UM lets you configure one or more applications.

Parents "<um-configuration>" on page 8

Children "<application>" on page 14

XML Attributes: None.

Example:

```
<applications>
  <application name="SENDING-IXCM-LBTRM" template="SENDING">
    <contexts/>
    <event-queues/>
    <application-data/>
  </application>
</applications>
```

<application>

Description The `application` element contains option values for all object elements within a single, uniquely named, application.

Parents "<applications>" on page 13

Children

- "<application-data>" on page 24
- "<contexts>" on page 14
- "<event-queues>" on page 21
- "<hfxs>" on page 23

XML Attributes:

XML Attribute	Description	Default Value
`name`	Name of the application. Used as an optional parameter for **lbm_config_from_xml()**. If a name is not supplied, this must be the only occurrence of this element in the XML configuration file.	None
`template`	Name of the configuration template to use for the application.	None

Example:

```
<applications>
  <application name="SENDING-IXCM-LBTRM" template="SENDING">
    <application-data/>
    <contexts/>
    <event-queues/>
  </application>
  <application name="SENDING-IXCM-TCP" template="SENDING">
    <application-data/>
    <contexts/>
    <event-queues/>
</applications>
```

<contexts>

Description The `<contexts>` element is a container element for all UM *contexts* configured for an application. UM lets you create one or more contexts for an application.

Parents "<application>" on page 14

Children "<context>" on page 15.

XML Attributes:

XML Attribute	Description	Default Value
`template`	Name of the configuration template to apply to each individual context object configured within this element. Multiple templates can be applied by specifying them in a comma-separated-value manner, i.e., `"SENDING1,SENDING2"`. Can be overridden by a different template configured for an individual context.	None
`order`	Establishes the permission semantic for each individual context configured within this element. Valid values are `deny,allow` (deny what you specify, allow everything else) or `allow,deny` (allow what you specify, deny everything else). Works in conjunction with the "<context>" on page 15 XML attribute, `rule`.	`deny,allow`

Example:

```
<applications>
  <application>
    <contexts template="SENDING" order="deny,allow">
      <context name="SENDING-95" template="SENDING-LBTRM" rule="allow">
        <sources/>
        <receivers/>
        <wildcard-receivers/>
        <options/>
      </context>
    </contexts>
    <event-queues/>
    <application-data/>
  </application>
</applications>
```

<context>

Description The `<context>` element contains option values for a single context, organized into its child elements.

Important: Setting the `name` attribute in this element does not actually name a context. A context name must be established when you create the context. See the `name` description in the table below.

Parents "<contexts>" on page 14

Children

- "<sources>" on page 16
- "<receivers>" on page 18
- "<wildcard-receivers>" on page 19
- "<options>" on page 10

XML Attributes:

XML Attribute	Description	Default Value
`name`	Name of the context. UM only applies an XML configuration using this name to contexts that match this context name. Setting this XML name attribute does not name the context, but provides the ability to map previously created context attributes to this XML element. You can configure an automatic monitoring context by setting `name=29west_statistics_context`.	None
`template`	Name of the configuration template to use for the context object's options.	None
`rule`	Permits or restricts the creation of the context object. If `rule="deny"`, the context object errors upon creation.	`allow`

Example:

```
<applications>
  <application>
    <contexts template="SENDING" order="deny,allow">
      <context name="SENDING-95" template="SENDING-LBTRM" rule="allow">
        <sources/>
        <receivers/>
        <wildcard-receivers/>
        <options/>
      </context>
    </contexts>
  </application>
</applications>
```

<sources>

Description The `<sources>` element is a container for all UM *sources* configured for an application. UM lets you create one or more sources for an application.

Parents "<context>" on page 15

Children "<topic>" on page 17

XML Attributes:

XML Attribute	Description	Default Value
`template`	Name of the configuration template to apply to each individual source object configured within this element. Multiple templates can by applied by specifying them in a comma-separated-value manner, i.e., `"SENDING1,SENDING2"`. Can be overridden by a different template configured for an individual source.	None
`order`	Establishes the permission semantic for each individual source configured within this element. Valid values are `deny,allow` (deny what you specify, allow everything else) or `allow,deny` (allow what you specify, deny everything else). Works in conjunction with the "<topic>" on page 17 XML attribute `rule`.	`deny,allow`

Example:

```
<applications>
  <application>
    <contexts>
      <context>
        <sources template="SENDING" order="deny,allow">
          <topic topicname="ICXM" template="SENDING-LBTRM" rule="allow"/>
        </sources>
        <receivers/>
        <wildcard-receivers/>
        <options/>
      </context>
    </contexts>
    <event-queues/>
    <application-data/>
  </application>
</applications>
```

<topic>

Description The `<topic>` element contains option values for a single source or receiver.

Parents .

- "<hfxs>" on page 23
- "<receivers>" on page 18
- "<sources>" on page 16

Children "<options>" on page 10.

XML Attributes:

XML Attribute	Description	Default Value
`topicname`	The topic string for the topic that the source sends or the receiver accepts. Used as a parameter for **lbm_src_topic_alloc()**, **lbm_rcv_topic_lookup()**, **lbm_src_attr_create_from_xml()**, **lbm_src_attr_set_from_xml()**, **lbm_rcv_attr_create_from_xml()** and **lbm_rcv_attr_set_from_xml()**. Do not use with the `pattern` attribute.	None
`template`	Name of the configuration template to use for this topic's source or receiver options.	None
`rule`	Permits or restricts the creation of the source or receiver object. If `rule="deny"`, the object errors upon creation.	`allow`
`pattern`	Identify the set of options for this topic with a topic string pattern. Any source created with a topic string that matches this pattern receives the configured option values. Do not use with the `topicname` attribute.	None

Example:

```
<applications>
<application name="Sending">
  <contexts order="deny,allow">
    <context rule="allow" template="Sending-LBTRM">
      <sources order="deny,allow">
        <topic rule="allow" topicname="IXCM">
          <options type="source">
            <option default-value="224.12.5.101"
name="transport_lbtrm_multicast_address"/>
          </options>
        </topic>
      </sources>
    </context>
  </contexts>
</application>
</applications>
```

<receivers>

Description The `<receivers>` element is a container element for all UM *receivers* configured for an application. You can create one or more receivers for an application.

Parents "<context>" on page 15

Children “<topic>” on page 17

XML Attributes:

XML Attribute	Description	Default Value
`template`	Name of the configuration template to apply to each individual receiver object configured within this element. You can apply multiple templates by specifying them in a comma-separated-value manner, e.g., `"RECEIVING1,RECEIVING2"`.	None
`order`	Establishes the permission semantic for each individual receiver configured within this element. Valid values are `deny,allow` (deny what you specify, allow everything else) or `allow,deny` (allow what you specify, deny everything else). Works in conjunction with the “<topic>” on page 17 XML attribute `rule`.	`deny,allow`

Example:

```
<applications>
  <application>
    <contexts>
      <context>
        <sources/>
        <receivers template="RECEIVING" order="deny,allow">
          <topic topicname="ICXM" template="RECEIVING" rule="allow"/>
        </receivers>
        <wildcard-receivers/>
        <options/>
      </context>
    <event-queues/>
    <application-data/>
  </application>
</applications>
```

<wildcard-receivers>

Description The <wildcard-receivers> element is a container element for all UM *wildcard receivers* configured for an application. UM lets you create one or more wildcard receivers for an application.

Parents “<context>” on page 15

Children “<wildcard-receiver>” on page 20.

XML Attributes:

XML Attribute	Description	Default Value
`template`	Name of the configuration template to apply to each individual wildcard receiver object configured within this element. Multiple templates can by applied by specifying them in a comma-separated-value manner, i.e., `"RECEIVING1,RECEIVING2"`. Can be overridden by a different template configured for an individual wildcard receiver.	None
`order`	Establishes the permission semantic for each individual wildcard receiver configured within this element. Valid values are `deny,allow` (deny what you specify, allow everything else) or `allow,deny` (allow what you specify, deny everything else). Works in conjunction with the "<wildcard-receiver>" on page 20 XML attribute `rule`.	`deny,allow`

Example:

```
    <applications>
      <application>
        <contexts>
          <context>
            <sources>
            <receivers/>
            <wildcard-receivers template="RECEIVING" order="deny,allow">
              <wildcard-receiver template="RECEIVING-LBTRM" rule="allow" pattern="I*M"
pattern-type="pcre"/>
            </wildcard-receivers>
          <options/>
          </context>
        <event-queues/>
        <application-data/>
      </application>
    </applications>
```

<wildcard-receiver>

Description The `<wildcard-receiver>` element contains option values for a single wildcard receiver.

Parents "<wildcard-receivers>" on page 19.

Children "<options>" on page 10.

XML Attributes:

XML Attribute	Description	Default Value
`template`	Name of the configuration template to use for the wildcard receiver object's options.	None
`rule`	Permits or restricts the creation of the wildcard receiver object. If `rule="deny"`, the object errors upon creation.	`allow`
`pattern`	The wildcard receiver topic string pattern for this wildcard receiver object.	None
`pattern-type`	The type of pattern matching to use for the wildcard receiver object. Valid values are `pcre`, `regex` or `application-callback`.	`pcre`

Example:

```
    <applications>
      <application>
        <contexts>
          <context>
            <wildcard-receivers template="RECEIVING" order="deny,allow">
              <wildcard-receiver template="RECEIVING-LBTRM" rule="allow" pattern="I*M"
pattern-type="pcre"/>
            </wildcard-receivers>
          </context>
      </application>
    </applications>
```

<event-queues>

Description The `<event-queues>` Element is a container element for all UM *event queues* configured for an application. UM lets you create one or more event queues for an application.

Parents "<application>" on page 14.

Children "<event-queue>" on page 22.

XML Attributes:

XML Attribute	Description	Default Value
`template`	Name of the configuration template to apply to each individual event queue object configured within this element. You can apply multiple templates specifying them in a comma-separated-value manner, e.g., `"EVQ-1,EVQ-2"`. A template applied to an individual event queue will override an `<event-queues>`-level template.	None
`order`	Establishes the permission semantic for each individual event queue configured within this element. Valid values are `deny,allow` (deny what you specify, allow everything else) or `allow,deny` (allow what you specify, deny everything else). Works in conjunction with the "<event-queue>" on page 22 XML attribute, `rule`.	`deny,allow`

Example:

```
<applications>
  <application>
    <contexts/>
    <event-queues template="RECEIVING" order="deny,allow">
      <event-queue name="EVQ-1" template="SENDING-LBTRM" rule="allow"/>
    </event-queues>
    <application-data/>
  </application>
</applications>
```

<event-queue>

Description The `<event-queue>` element contains option values for a single event queue.

Parents "<event-queues>" on page 21.

Children "<options>" on page 10.

XML Attributes:

XML Attribute	Description	Default Value
`name`	Name of the event queue. Used as a parameter for **lbm_event_queue_attr_create_from_xml()** and **lbm_event_queue_attr_set_from_xml()**.	None
`template`	Name of the configuration template to use for the event queue object's options.	None
`rule`	Permits or restricts the creation of the event queue object. If `rule="deny"`, the object errors upon creation.	`allow`

Example:

```
<applications>
  <application>
    <contexts/>
    <event-queues template="RECEIVING" order="deny,allow">
      <event-queue name="EVQ-1" template="SENDING-LBTRM" rule="allow"/>
    </event-queues>
    <application-data/>
  </application>
</applications>
```

<hfxs>

Description The `<hfxs>` element is a container for all UM *HFX objects* configured for an application. Within the `<hfxs>` element, options are organized by topic.

Parents "<application>" on page 14

Children "<topic>" on page 17

XML Attributes:

XML Attribute	Description	Default Value
`template`	Name of the configuration template to apply to each individual HFX object configured within this element. Multiple templates can by applied by specifying them in a comma-separated-value manner, i.e., `"SENDING1,SENDING2"`. Can be overridden by a different template configured for an individual HFX object.	None
`order`	Establishes the permission semantic for each individual HFX object configured within this element. Valid values are `deny,allow` (deny what you specify, allow everything else) or `allow,deny` (allow what you specify, deny everything else). Works in conjunction with the "<topic>" on page 17 XML attribute, `rule`.	`deny,allow`

Example:

```
<applications>
  <application>
    <hfxs template="SENDING" order="deny,allow">
      <topic topicname="ICXM" template="SENDING-LBTRM" rule="allow"/>
    </hfxs>
  </application>
</applications>
```

<application-data>

Description The `<application-data>` element is a free-form text comment field that you can use to store application-specific or options-group-specific metadata. When defined at the options level, this content overrides <application-data> elements defined at the application level.

Your application can retrieve this data via the **lbm_*_attr_getopt()** and **lbm_*_attr_str_getopt()** API functions under the option name `application_data`. You can also programmatically set it using the equivalent ***_setopt()** APIs. The `application_data` option is defined for all option scopes.

Also, you can set or retrieve this value at runtime via the ***_getopt()** and ***_setopt()** functions defined for the following types:

- `lbm_context_t`
- `lbm_src_t`
- `lbm_rcv_t`
- `lbm_wildcard_rcv_t`
- `lbm_event_queue_t`
- `lbm_hfx_t`

Parents . .

- “<application>” on page 14
- “<options>” on page 10

Children. None.

XML Attributes:

XML Attribute	Description	Default Value
`xml:space`	How whitespace is handled. `default` trims leading and trailing whitespace (e.g., tabs, spaces, linefeeds, etc.), and compresses multiple whitespace characters into a single space character. `preserve` preserves the whitespace exactly as read.	`default`

Example:

```
<applications>
  <application name="SENDING-IXCM-LBTRM" template="SENDING">
    <application-data>
    SENDING-IXCM-LBTRM options application data string
    <application-data/>
    <contexts/>
      <options type="context">
        <option/>
        . . .
        <application-data>
        context options application data string
        <application-data/>
      </options>
    <event-queues/>
  </application>
</applications>
```

Sample XML Configuration File

A sample XML configuration file appears below and has the following notable aspects.

- Contains object attributes for a UM context and source.
- Application name is `Sending`.
- Uses a template of attributes also called `Sending-LBTRM`.
- The template, `Sending-LBTRM`, uses the `order` attribute for the `fd_management_type` to allow all file descriptor types except DEVPOLL. However the `Sending-LBTRM` application further restricts the file descriptor types to exclude EPOLL in addition to DEVPOLL.

```
<um-configuration version="1.0">
<templates>
<template name="Sending-LBTRM">
  <options type="source">
    <option default-value="0" name="late_join"/>
    <option default-value="500" name="resolver_advertisement_maximum_initial_interval"/>
    <option default-value="5000"
name="resolver_advertisement_minimum_initial_duration"/>
    <option default-value="10" name="resolver_advertisement_minimum_initial_interval"/>
```

```
    <option default-value="60" name="resolver_advertisement_minimum_sustain_duration"/>
    <option default-value="1000" name="resolver_advertisement_sustain_interval"/>
    <option default-value="lbtrm" name="transport"/>
    <option default-value="14400" name="transport_lbtrm_destination_port"/>
    <option default-value="0.0.0.0" name="transport_lbtrm_multicast_address"/>
  </options>
  <options type="context">
    <option default-value="wsaeventselect" name="fd_management_type" order="deny,allow">
      <deny>wincompport</deny>
    </option>
    <option default-value="5000" name="mim_delivery_control_activity_check_interval"/>
    <option default-value="60000" name="mim_delivery_control_activity_timeout"/>
    <option default-value="6000" name="mim_delivery_control_loss_check_interval"/>
    <option default-value="2000000" name="resolver_initial_advertisement_bps"/>
    <option default-value="2000" name="resolver_initial_advertisements_per_second"/>
    <option default-value="2000" name="resolver_initial_queries_per_second"/>
    <option default-value="2000000" name="resolver_initial_query_bps"/>
  </options>
</template>
</templates>
<applications>
<application name="Sending">
  <contexts order="deny,allow">
    <context rule="allow" template="Sending-LBTRM">
      <sources order="deny,allow">
        <topic rule="allow" topicname="IXCM">
          <options type="source">
            <option default-value="1" name="late_join"/>
            <option default-value="lbtrm" name="transport"/>
            <option default-value="14488" name="transport_lbtrm_destination_port"/>
            <option default-value="224.12.5.101"
name="transport_lbtrm_multicast_address"/>
          </options>
        </topic>
      </sources>
      <receivers order="deny,allow"/>
      <wildcard-receivers order="deny,allow"/>
      <options type="context">
        <option default-value="224.9.10.11" name="resolver_multicast_address"/>
        <option default-value="224.9.10.11" name="resolver_multicast_incoming_address"/>
        <option default-value="12965" name="resolver_multicast_incoming_port"/>
        <option default-value="224.9.10.11" name="resolver_multicast_outgoing_address"/>
        <option default-value="12965" name="resolver_multicast_outgoing_port"/>
        <option default-value="12965" name="resolver_multicast_port"/>
        <option default-value="224.9.10.12" name="resolver_multicast_interface"/>
        <option default-value="0" name="resolver_multicast_receiver_socket_buffer"/>
        <option default-value="wsaeventselect" name="fd_management_type"
order="deny,allow">
          <deny>wincompport</deny>
        </option>
      </options>
    </context>
  </contexts>
  <event-queues order="deny,allow">
    <event-queue rule="allow">
      <options type="event-queue">
        <option default-value="lbm" name="monitor_transport"/>
        <option default-value="" name="monitor_appid"/>
      </options>
    </event-queue>
  </event-queues>
</application>
</applications>
</um-configuration>
```

Using the Order and Rule XML Attributes

The `order` and `rule` XML attributes combine to enable you to permit or restrict the creation of primitive UM objects. The container elements such as the “<contexts>” on page 14, “<sources>” on page 16,

“<receivers>” on page 18, etc. have the `order` attribute. The single object elements, such as the “<context>” on page 15, “<topic>” on page 17, etc., have the `rule` attribute. The default for both attributes allows creation of all objects. You can however, exert some administrative control over your applications by allowing the creation of only certain objects.

You can vary the `order` attribute values to suit whether permission or restriction is more prevalent. In the example below, only a single topic needs to be restricted, so we use the default values for the `order` attribute with only a single topic restricted with a `rule="deny"` attribute.

```
<sources order="deny,allow">
  <topic topicname="CDEF" rule="deny"/>
  <!-- all other source topics allowed -->
</sources>
```

In contrast, the following example requires the creation of only a single receiver topic object, so you can change the `order` attribute to `allow,deny`, which restricts the creation of all receiver topic objects except the one allowed.

```
<receivers order="allow,deny">
  <topic topicname="AARM" rule="allow"/>
  <!-- all other receive topics denied -->
</receivers>
```

You can also combine topic names with topic patterns. In the example below, we set the `order` attribute to the default. Topic `ISM` is denied with its `order` attribute. Topics `IRM` and `SRM` satisfy both their own `allow` rules and the pattern `*R* deny` rule. So when you allocate a source topic with **lbm_src_topic_alloc()**, UM accepts the rule that matches the `order` attribute default, which is `allow`.

```
<sources order="deny,allow">
  <topic topicname="ISM" rule="deny"/>
  <topic topicname="IRM" rule="allow"/>
  <topic pattern="*R*" rule="deny"/>
  <topic topicname="SRM" rule="allow"/>
</sources>
```

As a result of the above configuration, UM allows the creation of source topic objects `IRM` and `SRM`, and all other topics, except those that match the pattern `*R*`.

XML Configuration File DTD

The XML configuration file DTD is integrated into UM and appears below.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT um-configuration (license | templates | applications)*>
<!ATTLIST um-configuration version CDATA #REQUIRED>

<!ELEMENT license ( #PCDATA )>
<!ATTLIST license format (filename | string) "string">
<!ATTLIST license xml:space (default | preserve) "default">

<!ELEMENT templates (template*)>

<!ELEMENT template (options+)>
<!ATTLIST template name CDATA #REQUIRED>

<!ELEMENT options (option | application-data)*>
<!ATTLIST options type (event-queue | context | source | receiver | wildcard-receiver |
hfx) #IMPLIED>
```

```
<!ELEMENT option (allow | deny)*>
<!ATTLIST option name CDATA #REQUIRED>
<!ATTLIST option default-value CDATA #IMPLIED>
<!ATTLIST option order CDATA #IMPLIED>

<!ELEMENT application-data ( #PCDATA )>
<!ATTLIST application-data xml:space (default | preserve) "default">

<!ELEMENT allow ( #PCDATA )>
<!ATTLIST allow xml:space (default | preserve) "default">

<!ELEMENT deny ( #PCDATA )>
<!ATTLIST deny xml:space (default | preserve) "default">

<!ELEMENT applications (application*)>

<!ELEMENT application (contexts | event-queues | hfxs | application-data)+>
<!ATTLIST application name CDATA #IMPLIED>
<!ATTLIST application template CDATA #IMPLIED>

<!ELEMENT contexts (context*)>
<!ATTLIST contexts template CDATA #IMPLIED>
<!ATTLIST contexts order CDATA #IMPLIED>

<!ELEMENT event-queues (event-queue*)>
<!ATTLIST event-queues template CDATA #IMPLIED>
<!ATTLIST event-queues order CDATA #IMPLIED>

<!ELEMENT hfxs (topic*)>
<!ATTLIST hfxs template CDATA #IMPLIED>
<!ATTLIST hfxs order CDATA #IMPLIED>

<!ELEMENT event-queue (options*)>
<!ATTLIST event-queue name CDATA #IMPLIED>
<!ATTLIST event-queue template CDATA #IMPLIED>
<!ATTLIST event-queue rule (allow | deny) "allow">

<!ELEMENT context (sources | receivers | wildcard-receivers | options)+>
<!ATTLIST context name CDATA #IMPLIED>
<!ATTLIST context template CDATA #IMPLIED>
<!ATTLIST context rule (allow | deny) "allow">

<!ELEMENT sources (topic*)>
<!ATTLIST sources template CDATA #IMPLIED>
<!ATTLIST sources order CDATA #IMPLIED>

<!ELEMENT receivers (topic*)>
<!ATTLIST receivers template CDATA #IMPLIED>
<!ATTLIST receivers order CDATA #IMPLIED>

<!ELEMENT wildcard-receivers (wildcard-receiver*)>
<!ATTLIST wildcard-receivers template CDATA #IMPLIED>
<!ATTLIST wildcard-receivers order CDATA #IMPLIED>

<!ELEMENT topic (options*)>
<!ATTLIST topic template CDATA #IMPLIED>
<!ATTLIST topic rule (allow | deny) "allow">
<!ATTLIST topic pattern CDATA #IMPLIED>
<!ATTLIST topic topicname CDATA #IMPLIED>

<!ELEMENT wildcard-receiver (options*)>
<!ATTLIST wildcard-receiver template CDATA #IMPLIED>
<!ATTLIST wildcard-receiver rule (allow | deny) "allow">
<!ATTLIST wildcard-receiver pattern CDATA #IMPLIED>
<!ATTLIST wildcard-receiver pattern-type (pcre | regex | application-callback) #IMPLIED>
```

Configuration File Restrictions

The only options that you cannot set via configuration file are those that require function pointers as their value. Some examples include **context resolver_source_notification_function** and **wildcard_receiver pattern_callback**. you can set these options via only API functions. See "Options (Callbacks) That Cannot Be Set From a UM Configuration File" on page 239 for a list of these options.

Attributes Objects

Many UM primitive objects have a corresponding *attributes object*, which lets you create *custom attributes*. From here you can set options specific to an object (but different from default option settings) prior to creating that object. The following table lists the UM primitive objects and corresponding attributes objects.

Table 1. UM Objects and Corresponding Attributes Objects

UM object	Corresponding Attributes Object(s)
`lbm_context_t`	`lbm_context_attr_t`
`lbm_topic_t`	`lbm_src_topic_attr_t`, `lbm_rcv_topic_attr_t`
`lbm_wildcard_rcv_t`	`lbm_wildcard_rcv_attr_t`
`lbm_event_queue_t`	`lbm_event_queue_attr_t`
`lbm_hfx_t`	`lbm_hfx_attr_t`

You call API functions to create attributes objects and set, retrieve, or delete their values. These function names are based on the attributes object name and are shown in the following table, using the context object as an example. See the *C API* for all context attribute functions.

Table 2. UM API Functions For Working With `lbm_context_attr_t` Attributes Objects

Action	UM API function
Create	**lbm_context_attr_create()**
Set Option from Binary Value	**lbm_context_attr_setopt()**
Set Option from String Value	**lbm_context_attr_str_setopt()**
Get Option as Binary Value	**lbm_context_attr_getopt()**
Get Option as String Value	**lbm_context_attr_str_getopt()**
Delete	**lbm_context_attr_delete()**
For other object types, replace `context` with `event_queue`, `hfx`, `rcv_topic`, `src_topic`, or `wildcard_rcv`.	

The following sections describe in detail the use of these UM API functions. The functions related to `lbm_context_attr_t` objects are used for the purpose of illustration, but the instructions (if not the specifics) apply to all UM attributes objects.

Creating An Attributes Object

In the following example, the call to **lbm_context_attr_create()** creates the custom attributes object, and initializes each attribute from the current default values. Subsequent calls to **lbm_context_attr_setopt()** or **lbm_context_attr_str_setopt()** modify only the attributes object values.

```
lbm_context_attr_t * attrib;
int rc;
int errnum;
const char * errmsg;

rc = lbm_context_attr_create(&attrib);
if (rc != 0)
{
   errnum = lbm_errnum();
   errmsg = lbm_errmsg();
   fprintf(stderr, "Error %d returned from lbm_context_attr_create(), %s\n",
      errnum, errmsg);
}
```

This example also illustrates the proper way to determine the success or failure of an UM API call. Most UM API calls return **0** to indicate success, and **-1** to indicate failure. To retrieve the specific UM error code for the failure, call **lbm_errnum()**. To retrieve a text string describing the error code, call **lbm_errmsg()**.

Setting an Option from a Binary Value

For an option of type other than "string", call **lbm_context_attr_setopt()** to set its value. (See the C API reference for details on this function.) The final two parameters in the function are a pointer to a variable containing the option value, and a pointer to a variable of type `size_t` that contains the correct length of the option value variable.

UM options are of three general types that:

- accept values in a well-defined range (Examples include **context transport_tcp_port_low** and **context transport_tcp_port_high**. Each requires a value which corresponds to a valid TCP port number.)
- accept values from an enumerated set, (For example, **context operational_mode**. Manifest constants are provided in `lbm.h` for each permitted value. In the case of **context operational_mode**, those constants are **LBM_CTX_ATTR_OP_EMBEDDED** and **LBM_CTX_ATTR_OP_SEQUENTIAL**.)
- act as switches, enabling or disabling a particular feature (For example, **context resolver_cache**. The set of allowed values is limited to **0** (indicating off, no, false, or disabled), and **1**, indicating on, yes, true, or enabled.)

The example code below sets four options. First, we set the operational mode to sequential. Then we set the transport TCP port low and high values to 4901 and 4920, respectively. Finally, we tell UM that our application will not be using multiple sending threads per transport session.

```
lbm_context_attr_t * attrib; /* Must have already been created */
int rc;
unsigned short int optval;
size_t optlen;

/* Set the operational_mode */
optlen = sizeof(optval);
optval = LBM_CTX_ATTR_OP_SEQUENTIAL;
rc = lbm_context_attr_setopt(attrib, "operational_mode", &optval, optlen);
```

```
if (rc != 0)
{
   /* Handle error */
}

/* Set transport_tcp_port_low */
optlen = sizeof(optval);
optval = 4901;
rc = lbm_context_attr_setopt(attrib, "transport_tcp_port_low", &optval, optlen);
if (rc != 0)
{
   /* Handle error */
}

/* Set transport_tcp_port_high */
optlen = sizeof(optval);
optval = 4920;
rc = lbm_context_attr_setopt(attrib, "transport_tcp_port_high", &optval, optlen);
if (rc != 0)
{
   /* Handle error */
}

/* Set transport_session_multiple_sending_threads */
optlen = sizeof(optval);
optval = 0;
rc = lbm_context_attr_setopt(attrib, "transport_session_multiple_sending_threads",
                             &optval, optlen);
if (rc != 0)
{
   /* Handle error */
}
```

Setting an Option from a String Value

Setting an option from a string value effectively does the same thing that setting an option from a binary value does. However, the option value is passed as a null-terminated string, rather than as value and length pointers. UM uses this mechanism to process options in a configuration file. Thus, the format used for option values must match the format you would use in a configuration file.

In the following example, as before, we set the operational mode to sequential. Then we set the transport TCP port low and high values to 4901 and 4920, respectively. Finally, we tell UM that our application will not be using multiple sending threads per transport session.

```
lbm_context_attr_t * attrib; /* Must have already been created */
int rc;

/* Set the operational_mode */
rc = lbm_context_attr_str_setopt(attrib, "operational_mode", "sequential");
if (rc != 0)
{
   /* Handle error */
}

/* Set transport_tcp_port_low */
rc = lbm_context_attr_str_setopt(attrib, "transport_tcp_port_low", "4901");
if (rc != 0)
{
   /* Handle error */
}

/* Set transport_tcp_port_high */
rc = lbm_context_attr_str_setopt(attrib, "transport_tcp_port_high", "4920");
if (rc != 0)
{
   /* Handle error */
}
```

```
/* Set transport_session_multiple_sending_threads */
rc = lbm_context_attr_str_setopt(attrib, "transport_session_multiple_sending_threads",
                                 "0");
if (rc != 0)
{
   /* Handle error */
}
```

Getting an Option as a Binary Value

Getting an option as a binary value is very similar to setting an option from a binary value: it requires knowledge of not only the option name, but its type as well. The final two parameters in the call to **lbm_context_attr_getopt()** are a pointer to a variable to receive the current option value, and a pointer to a variable of type **size_t** which contains the length of the option value variable. This length must be correct for the specified option.

In the example code below, we set the option values for operational mode, the transport TCP port low and high values, and retrieve multiple sending threads.

```
lbm_context_attr_t * attrib; /* Must have already been created */
int rc;
unsigned short int optval;
size_t optlen;

/* Get the operational_mode */
optlen = sizeof(optval);
rc = lbm_context_attr_getopt(attrib, "operational_mode", &optval, &optlen);
if (rc != 0)
{
   /* Handle error */
}
/* optval now contains LBM_CTX_ATTR_OP_EMBEDDED or LBM_CTX_ATTR_OP_SEQUENTIAL */

/* Get transport_tcp_port_low */
optlen = sizeof(optval);
rc = lbm_context_attr_getopt(attrib, "transport_tcp_port_low", &optval, &optlen);
if (rc != 0)
{
   /* Handle error */
}
/* optval now contains the value of transport_tcp_port_low, which should be 4901 */

/* Get transport_tcp_port_high */
optlen = sizeof(optval);
rc = lbm_context_attr_getopt(attrib, "transport_tcp_port_high", &optval, &optlen);
if (rc != 0)
{
   /* Handle error */
}
/* optval now contains the value of transport_tcp_port_high, which should be 4920 */

/* Get transport_session_multiple_sending_threads */
optlen = sizeof(optval);
rc = lbm_context_attr_getopt(attrib, "transport_session_multiple_sending_threads",
                             &optval, &optlen);
if (rc != 0)
{
   /* Handle error */
}
/* optval now contains the value of transport_session_multiple_sending_threads,
   which should be 0. */
```

Getting an Option as a String Value

Getting an option as a string value effectively does the same thing that getting an option as a binary value does. However, the option value is returned as a null-terminated string, just as you would specify the option value in a configuration file. The final two parameters in the call to **lbm_context_attr_str_getopt()** are a pointer to a string variable to receive the current option value, and a pointer to a variable of type **size_t** which contains the maximum size of the option value string variable.

In the example code below, the option values for operational mode, the transport TCP port low and high values, and multiple sending threads are retrieved.

```
lbm_context_attr_t * attrib; /* Must have already been created */
int rc;
char optval_string[256];

/* Get the operational_mode */
optlen = sizeof(optval_string);
rc = lbm_context_attr_str_getopt(attrib, "operational_mode", optval_string, &optlen);
if (rc != 0)
{
   /* Handle error */
}
/* optval_string now contains either "embedded" or "sequential" */

/* Get transport_tcp_port_low */
optlen = sizeof(optval_string);
rc = lbm_context_attr_str_getopt(attrib, "transport_tcp_port_low",
                                 optval_string, &optlen);
if (rc != 0)
{
   /* Handle error */
}
/* optval_string now contains the string value of transport_tcp_port_low,
   which should be "4901" */

/* Get transport_tcp_port_high */
optlen = sizeof(optval_string);
rc = lbm_context_attr_str_getopt(attrib, "transport_tcp_port_high",
                                 optval_string, &optlen);
if (rc != 0)
{
   /* Handle error */
}
/* optval_string now contains the string value of transport_tcp_port_high,
   which should be "4920" */

/* Get transport_session_multiple_sending_threads */
optlen = sizeof(optval_string);
rc = lbm_context_attr_str_getopt(attrib, "transport_session_multiple_sending_threads",
                                 optval_string, &optlen);
if (rc != 0)
{
   /* Handle error */
}
/* optval_string now contains the value of transport_session_multiple_sending_threads,
   which should be "0". */
```

Deleting an Attributes Object

Once the attributes object is no longer needed, it should be deleted.

```
lbm_context_attr_t * attrib; /* Must have already been created */
int rc;

rc = lbm_context_attr_delete(attrib);
if (rc != 0)
```

```
{
   /* Handle error */
}
```

Restrictions

There are no restrictions on setting options via attributes objects. Any option which can be set via a configuration file, can also be set via an attributes object. In addition, attributes objects allow setting certain options (such as function pointers) which cannot be set with a configuration file.

Modifying Current Attributes

A few options within an UM object's current attributes can be set after the object is created. UM API functions supporting such actions operate on the object itself, rather than on an attributes object. In addition to modifying the current attributes, the value of options from the current attributes can be fetched.

The UM objects which support these actions are **lbm_src_t**, **lbm_rcv_t**, **lbm_context_t**, and **lbm_event_queue_t**. For each such object, there are corresponding API functions to set an option from a binary value, set an option from a string value, get an option as a binary value, and get an option as a string value. These function names are based on the object name, suffixed with **_setopt()**, **_str_setopt()**, **_getopt()**, and **_str_getopt()**. As an illustration of this convention, the API functions for working with **lbm_event_queue_t** objects are shown in the following table.

Table 3. UM API Functions For Working With lbm_event_queue_t Objects

Action	UM API function
Set Option from a Binary Value	**lbm_event_queue_setopt()**
Set Option from a String Value	**lbm_event_queue_str_setopt()**

The following sections describe in detail the use of these UM API functions. The functions related to **lbm_event_queue_t** objects are used for the purpose of illustration, but the instructions (if not the specifics) apply to all such UM objects.

Setting An Option from a Binary Value

Setting an option from a binary value requires knowledge of not only the option name, but its type and allowable values as well. The final two parameters in the call to **lbm_event_queue_setopt()** are a pointer to a variable which contains the option value to be set, and a pointer to a variable of type **size_t** which contains the length of the option value variable. This length must be correct for the specified option.

In the example code below, we set the queue size warning to 5000 events.

```
unsigned long int optval;
size_t optlen;
lbm_event_queue_t evq; /* must be previously created */
int rc;

/* Set the queue size warning */
optlen = sizeof(optval);
optval = 5000;
rc = lbm_event_queue_setopt(&evq, "queue_size_warning", &optval, &optlen);
```

```
if (rc != 0)
{
   /* Handle error */
}
```

Setting An Option from a String Value

Setting an option from a string value effectively does the same thing that setting an option from a binary value does. However, the option value is passed as a null-terminated string, rather than as value and length pointers. This is similar to the mechanism used by UM to process options in a configuration file. Thus, the format used for option values must match the format you would use in a configuration file.

As before, we set the queue size warning to 5000 events.

```
lbm_event_queue_t evq; /* must be previously created */
int rc;

/* Set the queue size warning */
rc = lbm_event_queue_setopt(&evq, "queue_size_warning", "5000");
if (rc != 0)
{
   /* Handle error */
}
```

Restrictions

Modifying the current attributes of an object allows only a very limited subset of options to be set or retrieved. Consult subsequent sections of this document to determine if a particular option can be specified.

Retrieving Current Option Values

Most UM objects allow their current attributes' option values to be retrieved during operation. UM API functions supporting such actions operate on the object itself.

The UM objects which support these actions are **lbm_src_t**, **lbm_rcv_t**, **lbm_context_t**, and **lbm_event_queue_t**. For each such object, there are corresponding API functions to get an option as a binary value, and get an option as a string value. These function names are based on the object name, suffixed with **_getopt()**, and **_str_getopt()**. As an illustration of this convention, the API functions for working with **lbm_event_queue_t** objects are shown in the following table.

Table 4. UM API Functions For Retrieving Option Values from lbm_event_queue_t Objects

Action	UM API function
Get Option as a Binary Value	**lbm_event_queue_getopt()**
Get Option as a String Value	**lbm_event_queue_str_getopt()**

The following sections describe in detail the use of these UM API functions. The functions related to **lbm_event_queue_t** objects are used for the purpose of illustration, but the instructions (if not the specifics) apply to all such UM objects.

Getting An Option as a Binary Value

Getting an option as a binary value is very similar to setting an option from a binary value: it requires knowledge of not only the option name, but its type as well. The final two parameters in the call to **lbm_event_queue_getopt()** are a pointer to a variable to receive the current option value, and a pointer to a variable of type **size_t** which contains the length of the option value variable. This length must be correct for the specified option.

In the example code below, the option value for the queue size warning is retrieved.

```
unsigned long int optval;
size_t optlen;
lbm_event_queue_t evq; /* must be previously created */
int rc;

/* Get the queue size warning value */
optlen = sizeof(optval);
rc = lbm_event_queue_getopt(&evq, "queue_size_warning", &optval, &optlen);
if (rc != 0)
{
   /* Handle error */
}
/* optval now contains the value of queue_size_warning, which should be 5000 */
```

Getting An Option as a String Value

Getting an option as a string value effectively does the same thing that getting an option as a binary value does. However, the option value is returned as a null-terminated string, just as you would specify the option value in a configuration file. The final two parameters in the call to **lbm_event_queue_str_getopt()** are a pointer to a string variable to receive the current option value, and a pointer to a variable of type **size_t** which contains the maximum size of the option value string variable.

In the example code below, the option value for the queue size warning is retrieved.

```
char optval_string[256];
size_t optlen;
lbm_event_queue_t evq; /* must be previously created */
int rc;

/* Get the queue size warning value */
optlen = sizeof(optval_string);
rc = lbm_event_queue_str_getopt(&evq, "queue_size_warning", optval_string, &optlen);
if (rc != 0)
{
   /* Handle error */
}
/* optval now contains the value of queue_size_warning, which should be "5000" */
```

CHAPTER 2

Example Configuration Scenarios

This chapter includes the following topics:

Highest Throughput

The following configuration option tunes UMS for the highest possible throughput.

```
#
# LBM can be configured to make efficient use of CPU time, leading
# to the highest-possible throughput (bytes per second or messages
# per second).  This may come at the expense of latency at low
# message rates.  The following line configures LBM to accumulate
# 8KB of messages (or for wait implicit_batching_interval) before sending.
#
source implicit_batching_minimum_length 8192
```

Lowest Latency

This is an example configuration that favors low latency at the expense of higher CPU utilization and potentially lower throughput.

```
#
# Latency can be reduced at the expense of network efficiency and
# system CPU time by adjusting implicit batching parameters.  The
# default parameters hold messages for up to 200 milliseconds or until
# 2048 bytes are waiting to go.  The lowest possible latency is
# obtained by setting the minimum batching length to 1 byte, which
# effectively disables the implicit batching feature.  For example:
#
```

```
context mim_implicit_batching_minimum_length 1
source      implicit_batching_minimum_length 1
#
# Latency can be kept to a minimum with UM by writing receiving
# applications that can accept messages in the order they arrive.
# See https://communities.informatica.com/infakb/faq/5/Pages/80043.aspx and
# http://www.29West.Com/docs/THPM/tcp-latency.html#TCP-RECEIVER-SIDE-LATENCY
# for more information.  Here's how to use arrival-order delivery:
#
receiver ordered_delivery 0
#
# Disable Nagel's algorithm (batching) for TCP responses to eliminate
# queuing latency when sending only single responses.
#
context response_tcp_nodelay 1
#
# If you are running a LAN environment with under 100 machines, you can
# drastically improve your recovery related latencies without significant
# additional network overhead by using the following UM loss
# recovery parameter. See https://communities.informatica.com/infakb/faq/5/Pages/
80070.aspx
# for additional information about this and other recovery parameters.
#
receiver transport_lbtrm_nak_backoff_interval 10
#
# Use of a zero value for the following parameter sends an immediate NAK upon
# loss detection, which can further reduce repair latency. (Immediate NAKs do
# not elicit an NCF by the source.) It is critical you understand the implications
# of this feature and we recommend that you contact http://29west.com/support to
# learn more before enabling it.
#
# receiver transport_lbtrm_nak_initial_backoff_interval 0
#
```

Creating Multicast Sources

This is an example configuration file that changes the default transport to reliable multicast so all sources created send messages over LBT-RM.

```
#
# UM can be configured to create sources using the LBT-RM reliable
# multicast protocol instead of the default TCP.
#
source transport LBT-RM
#
# Stable and reliable operation with multicast requires careful
# setting of rate control limits.  See
# http://www.29west.com/docs/THPM/thpm.html#GROUP-RATE-CONTROL
# for background information.
#
# It's generally best to start with small limits and gradually
# increase them after testing indicates that they can be safely
# sustained on your network.
#
# The following example limits (new) data to 10 Mbps and retransmissions
# to 1 Mbps (10%).  Note that when changing the data rate limit, the
# limit retransmission limit should be changed as well.  A good value
# for most purposes is between 2% and 10% of the data rate limit, with
# a lower limit of 1,000,000.
#
#
context transport_lbtrm_data_rate_limit 10000000
context transport_lbtrm_retransmit_rate_limit 1000000
```

Disabling Aspects of Topic Resolution

If you need to reduce the amount of Topic Resolution traffic on your network, use the following Configuration options and values in a Ultra Messaging Configuration file.

Note: Ultra Messaging does not recommend disabling both advertisements and queries because topics may not resolve at all.

Disabling Topic Advertisements

You can disable topic advertisements in the Initial Phase, Sustaining Phase or both phases of topic resolution.

Disabling Initial Phase Advertisements

Use one or both of the following options to disable topic advertisements in only the Initial Phase.

```
source resolver_advertisement_minimum_initial_interval 0
source resolver_advertisement_maximum_initial_interval 0
```

Disabling Sustaining Phase Advertisements

Use the following option to disable topic advertisements in only the Sustaining Phase.

```
source resolver_advertisement_sustain_interval 0
```

Disabling Receiver Topic Queries

You can disable the querying of topics by receivers in the Initial Phase, Sustaining Phase or both phases of topic resolution.

Disabling Initial Phase Queries

Use one or both of the following options to disable topic queries in only the Initial Phase.

```
receiver resolver_query_minimum_initial_interval 0
receiver resolver_query_maximum_initial_interval 0
```

Disabling Sustaining Phase Queries

Use one or both of the following options to disable topic queries in only the Sustaining Phase.

```
receiver resolver_query_sustain_interval 0
receiver resolution_number_of_sources_query_threshold 0
```

Disabling Wildcard Topic Queries

Use one or both of the following options to disable topic queries by wildcard receivers.

```
wildcard_receiver resolver_query_minimum_interval 0
wildcard_receiver resolver_query_maximum_interval 0
```

Disabling Store (Context) Name Queries

Use one or both of the following options to disable context name queries by sources.

```
resolver_context_name_query_maximum_interval 0
resolver_context_name_query_minimum_interval 0
```

Disabling All But the Minimum Topic Resolution Traffic

A minimalist approach to topic resolution can take different forms based on you requirements. One approach is to disable all traffic except for queries in the sustaining phase. Add the following settings to your Ultra Messaging configuration file to implement this approach.

```
source resolver_advertisement_minimum_initial_interval 0
source resolver_advertisement_sustain_interval 0
receiver resolver_query_minimum_initial_interval 0
receiver resolution_number_of_sources_query_threshold 1
wildcard_receiver resolver_query_minimum_interval 0
```

Re-establish Pre-4.0 Topic Resolution

Ultra Messaging topic resolution prior to LBM Version 4.0 did not have resolution phases. To implement pre-4.0 topic resolution, include the following configuration option changes in your Ultra Messaging configuration file.

```
# ----- Disable Advertisements in 4.0 Initial Phase
source resolver_advertisement_minimum_initial_interval 0

# ----- Re-establish pre-4.0 Advertisement Behavior
source resolver_advertisement_minimum_sustain_duration 0
context resolver_sustain_advertisement_bps 0

# ----- Disable Queries in 4.0 Initial Phase
receiver resolver_query_minimum_initial_interval 0

# ----- Re-establish pre-4.0 Query Behavior
receiver resolver_query_sustain_interval 100
receiver resolver_query_minimum_sustain_duration 0
context resolver_sustain_query_bps 0
receiver resolution_number_of_sources_query_threshold 1

# ----- Re-establish pre-4.0 Wildcard Query Behavior
wildcard_receiver resolver_query_minimum_interval 0
```

Unicast Resolver

To use the unicast resolver, use a configuration file like the following example:

```
#
# Topic resolution can be configured to use unicast traffic with an
# LBM resolver daemon (lbmrd) instead of the default which uses multicast.
# Be sure to insert the IP address of your lbmrd below.
#
context resolver_unicast_daemon 127.0.0.1:15380
```

Configure Previous Port Defaults

To use the previous default ports (prior to LBM 3.3 and UME 2.0), the following configuration file may be used.

```
context mim_destination_port              4401
context    mim_incoming_destination_port      4401
context    mim_outgoing_destination_port      4401
context    resolver_multicast_incoming_port  2965
context    resolver_multicast_outgoing_port  2965
context    resolver_multicast_port              2965
context    resolver_unicast_destination_port 5380
context    resolver_unicast_port_high        4406
context    resolver_unicast_port_low         4402
source    transport_lbtrm_destination_port  4400
context    transport_lbtrm_source_port_high  4399
context    transport_lbtrm_source_port_low   4390
context    transport_lbtru_port_high         4389
receiver transport_lbtru_port_high        4379
context    transport_lbtru_port_low          4380
receiver transport_lbtru_port_low         4360
context    request_tcp_port_high              4395
context    request_tcp_port_low              4391
context    transport_tcp_port_high              4390
context    transport_tcp_port_low             4371
source    ume_primary_store_port            4567
source    ume_secondary_store_port          4567
source    ume_tertiary_store_port              4567
```

Note: Alternatively, UMS/UMP will use the original port settings with the definition of the "LBM_USE_ORIG_DEFAULT_PORTS" environment variable (value not pertinent).

Configure New Port Defaults

In the unusual case that you must run older versions of Ultra Messaging (less than LBM 3.3 / UME 2.0) on certain machine(s) and need these older version to work with the machines running the current versions of UMS and UMP, you can use the following configuration file for the older versions to synchronize port usage between old and current versions.

```
context mim_destination_port              14401
context    mim_incoming_destination_port     14401
context    mim_outgoing_destination_port     14401
context    resolver_multicast_incoming_port  12965
context    resolver_multicast_outgoing_port  12965
context    resolver_multicast_port           12965
context    resolver_unicast_destination_port 15380
context    resolver_unicast_port_high        14406
context    resolver_unicast_port_low         14402
source    transport_lbtrm_destination_port  14400
context    transport_lbtrm_source_port_high  14399
context    transport_lbtrm_source_port_low   14390
context    transport_lbtru_port_high         14389
receiver transport_lbtru_port_high        14379
context    transport_lbtru_port_low          14380
receiver transport_lbtru_port_low         14360
context    request_tcp_port_high             14395
context    request_tcp_port_low              14391
context    transport_tcp_port_high           14390
context    transport_tcp_port_low            14371
source    ume_primary_store_port            14567
```

```
source    ume_secondary_store_port          14567
source    ume_tertiary_store_port           14567
```

Interrelated Configuration Options

Some Ultra Messaging configuration options are related in ways that might not be immediately apparent. Changing the value for one option without adjusting its related option can cause problems such as NAK storms, tail loss, etc. This section identifies these relationships and recommends a best practice for setting the interrelated options.

The following sections discuss configuration option relationships.

- "Preventing NAK Storms with NAK Interval Options" on page 42
- "Preventing Tail Loss With TSNI and NAK Interval Options" on page 43
- "Preventing IPC Receiver Deafness With Keepalive Options" on page 43
- "Preventing Erroneous LBT-RM/LBT-RU Session Timeouts" on page 44
- "Preventing Errors Due to Bad Multicast Address Ranges" on page 44
- "Preventing Store or Queue Timeouts" on page 45
- "Preventing ULB Timeouts" on page 45
- "Preventing Unicast Resolver Daemon Timeouts" on page 46
- "Preventing Undetected Late Join Loss" on page 46
- "Preventing Undetected Late Join Loss" on page 46

Preventing NAK Storms with NAK Interval Options

The NAK generation interval should be sufficiently longer than the NAK backoff interval so that the source, after receiving the first NAK from a receiver, has time to retransmit the missing datagram and prevent a NAK storm from all receivers. LBTRM, LBTRU, and MIM all use NAK generation and backoff intervals. The NAK behavior for all transports is the same.

Interrelated Options:

- `transport_lbtrm_nak_backoff_interval`
- `transport_lbtrm_nak_generation_interval`
- `transport_lbtru_nak_backoff_interval`
- `transport_lbtru_nak_generation_interval`
- `mim_nak_backoff_interval`
- `mim_nak_generation_interval`

Recommendation:

- Set the NAK generation interval to at least 2x the NAK backoff interval.

For more, see also

- "Transport LBT-RM Reliability Options" on page 105
- "Transport LBT-RU Reliability Options" on page 121
- "Multicast Immediate Messaging Reliability Options" on page 149

Example:

```
#
# +-------------------------------------------------------------------------+
# | To avoid NAK storms, set NAK generation interval to at least 2x the     |
# | NAK backoff interval.                                                   |
# +-------------------------------------------------------------------------+
#
receiver transport_lbtrm_nak_backoff_interval       200
receiver transport_lbtrm_nak_generation_interval  10000
```

Preventing Tail Loss With TSNI and NAK Interval Options

Tail loss refers to the situation when a receiver (subscriber) does not receive the last few messages sent by a source (publisher) before the source exits. A TSNI active threshold that is too small relative to the TSNI and/or NAK generation interval may cause tail loss, especially with ordered delivery.

Interrelated Options:

- `transport_topic_sequence_number_info_active_threshold`
- `transport_topic_sequence_number_info_interval`
- `transport_lbtrm_nak_generation_interval`
- `transport_lbtru_nak_generation_interval`

Recommendation:

- set the TSNI active threshold to at least 4x the topic sequence number info interval (TSNI) plus the NAK generation interval.

For more, see

- “Transport LBT-RM Reliability Options” on page 105
- “Transport LBT-RU Reliability Options” on page 121

Example:

```
#
# +-------------------------------------------------------------------------+
# | To avoid tail loss, set transport_topic_sequence_number_info_active_threshold |
# | to at least the sum of 4x the topic sequence number interval plus the NAK     |
# | generation interval.                                                    |
# | NOTE: resolver_active_threshold is in seconds.                          |
# +-------------------------------------------------------------------------+
#
source   transport_topic_sequence_number_info_interval          2000
receiver transport_lbtrm_nak_generation_interval               10000
source   transport_topic_sequence_number_info_active_threshold    60
```

Preventing IPC Receiver Deafness With Keepalive Options

With an LBT-IPC transport, an activity timeout that is too small relative to the session message interval may cause receiver deafness. If a timeout is too short, the keepalive messages might not be received in time to prevent the receiver from being deleted or disconnecting because the source appears to be gone.

Interrelated Options:

- `transport_lbtipc_activity_timeout`
- `transport_lbtipc_sm_interval`

Recommendations:

- set the activity timeout to at least 2x the session message interval

For more, see “Transport LBT-IPC Operation Options” on page 131.

Example:

```
#
# +---------------------------------------------------------------------------+
# | To avoid receiver deafness:                                               |
# | - set client activity timeout to at least 2x the acknowledgement interval.|
# | - set activity timeout to at least 2x the session message interval.       |
# +---------------------------------------------------------------------------+
#
receiver transport_lbtipc_activity_timeout          60000
source   transport_lbtipc_sm_interval               10000
```

Preventing Erroneous LBT-RM/LBT-RU Session Timeouts

An LBT-RM or LBT-RU receiver-side quiescent timeout may delete a transport session that a source is still active on. This can happen if the timeout is too short relative to the source's interval between session messages (which serve as a session keepalive).

Interrelated Options:

- `transport_lbtrm_activity_timeout`
- `transport_lbtrm_sm_maximum_interval`
- `transport_lbtru_activity_timeout`
- `transport_lbtru_sm_maximum_interval`

Recommendations:

- set the receiver LBT-RM or LBT-RU activity timeout to at least 3x the source session message maximum interval

For more, see “Transport LBT-RM Operation Options” on page 112 or “Transport LBT-RU Operation Options” on page 124.

Example:

```
#
# +---------------------------------------------------------------------------+
# | To avoid erroneous session timeouts, set receiver transport activity      |
# | timeout to at least 3x the source session message maximum interval.       |
# +---------------------------------------------------------------------------+
#
receiver transport_lbtrm_activity_timeout       60000
source   transport_lbtrm_sm_maximum_interval    10000
receiver transport_lbtru_activity_timeout       60000
source   transport_lbtru_sm_maximum_interval    10000
```

Preventing Errors Due to Bad Multicast Address Ranges

Sometimes it is easy to accidentally reverse the low and high values for LBT-RM multicast addresses, which actually creates a very large range. Aside from excluding intended addresses, this can cause error conditions.

Interrelated Options:

- `transport_lbtrm_multicast_address_low`
- `transport_lbtrm_multicast_address_high`

Recommendations:

- ensure that the intended low and high values for LBT-RM multicast addresses are not reversed

For more, see “Transport LBT-RM Network Options” on page 103.

Example:

```
#
# +-------------------------------------------------------------------------------+
# | To avoid incorrect LBT-RM multicast address ranges, ensure that you have not |
# | reversed the low and high values.                                             |
# +-------------------------------------------------------------------------------+
#
context  transport_lbtrm_multicast_address_low  224.10.10.10
context  transport_lbtrm_multicast_address_high 224.10.10.14
```

Preventing Store or Queue Timeouts

Note: These interrelations apply only to the Ultra Messaging Persistence or Ultra Messaging Queuing Edition.

A store or queue may be erroneously declared unresponsive if its activity timeout expires before it has had adequate opportunity to verify it is still active via activity check intervals.

Interrelated Options:

- `ume_store_activity_timeout`
- `ume_store_check_interval`
- `umq_queue_activity_timeout`
- `umq_queue_check_interval`

Recommendations:

- set the store or queue activity timeout to at least 5x the activity check interval

For more, see *Ultra Messaging Persistence Options* and/or (if using UM Queuing Edition), the UM Configuration Guide, 4.30. Ultra Messaging Queuing Options.

Example:

```
#
# +-------------------------------------------------------------------------------+
# | To avoid erroneous store or queue activity timeouts, set the activity        |
# | timeout to at least 5x the activity check interval.                           |
# +-------------------------------------------------------------------------------+
#
source ume_store_activity_timeout              3000
source ume_store_check_interval                 500
context umq_queue_activity_timeout             3000
context umq_queue_check_interval                500
```

Preventing ULB Timeouts

Note: These interrelations apply only to the Ultra Messaging Queuing Edition.

A ULB source or receiver may be erroneously declared unresponsive if its activity timeout expires before it has had adequate opportunities to attempt to re-register via activity check intervals if the source appears to be inactive. It is also possible for sources to attempt to reassign messages that have already been processed.

Interrelated Options:

- `umq_ulb_source_activity_timeout`
- `umq_ulb_source_check_interval`

- umq_ulb_application_set_message_reassignment_timeout
- umq_ulb_application_set_receiver_activity_timeout
- umq_ulb_check_interval

Recommendations:

- set the ULB source activity timeout to at least 5x the ULB source activity check interval
- set the ULB application set message reassignment timeout to at least 5x the ULB check interval
- set the ULB receiver activity timeout to at least 5x the ULB check interval

For more (if using UM Queuing Edition), see the UM Configuration Guide, 4.30. Ultra Messaging Queuing Options.

Example:

```
#
# +-------------------------------------------------------------------------------+
# | To avoid erroneous ULB source, receiver or application set message activity  |
# | timeouts, set the activity timeout to at least 5x the activity check         |
# | interval.                                                                     |
# +-------------------------------------------------------------------------------+
#
receiver umq_ulb_source_activity_timeout                   10000
receiver umq_ulb_source_check_interval                      1000
source umq_ulb_application_set_message_reassignment_timeout 50000
source umq_ulb_application_set_receiver_activity_timeout    10000
source umq_ulb_check_interval                               1000
```

Preventing Unicast Resolver Daemon Timeouts

A unicast resolver daemon may be erroneously declared inactive if its activity timeout expires before it has had adequate opportunity to verify that it is still alive.

Interrelated Options:

- resolver_unicast_activity_timeout
- resolver_unicast_check_interval

Recommendations:

- Set the unicast resolver daemon activity timeout to at least 5x the activity check interval. Or, if activity notification is not desired, set both options to 0.

For more, see “Resolver Operation Options” on page 69 *Resolver Operation Options*.

Example:

```
#
# +-------------------------------------------------------------------------------+
# | To avoid erroneous unicast resolver daemon timeouts, set the activity        |
# | timeout to at least 5x the activity check interval.                          |
# +-------------------------------------------------------------------------------+
#
context  resolver_unicast_activity_timeout      1000
context  resolver_unicast_check_interval         200
```

Preventing Undetected Late Join Loss

If during a Late Join operation, a transport times out while a receiver is requesting retransmission of missing messages, this can cause lost messages to go undetected and likely become unrecoverable.

Interrelated Options:

- `retransmit_request_generation_interval`
- `transport_tcp_activity_timeout`
- `transport_lbtrm_activity_timeout`
- `transport_lbtru_activity_timeout`
- `transport_lbtipc_activity_timeout`

Recommendations:

- set the Late Join retransmit request interval to a value less than its transport's activity timeout value

For more, see "Late Join Options" on page 157 and also the applicable Transport LBT-RU Operation Options section.

Example:

```
#
# +-------------------------------------------------------------------------+
# | To avoid a transport inactivity timeout while requesting Late Join      |
# | retransmissions, set the Late Join retransmit request interval to a value    |
# | less than its transport's activity timeout.                             |
# +-------------------------------------------------------------------------+
#
receiver retransmit_request_generation_interval 10000
receiver transport_lbtrm_activity_timeout       60000
```

Preventing Undetected Loss

It is possible that an unrecoverable loss due to unsatisfied NAKs or a transport activity timeout may go unreported if the delivery controller loss check is disabled or has too long an interval. For UMP stores, the loss check interval must be enabled. Two options (three, if using LBT-RM) are interrelated and must be set according to the guidelines below.

Interrelated Options:

- `delivery_control_loss_check_interval`
- `transport_lbtrm_activity_timeout`
- `transport_lbtrm_nak_generation_interval`
- `transport_lbtru_activity_timeout`

Recommendations:

- For LBT-RM, set the transport activity timeout to value greater than the sum of the delivery control loss check interval and the NAK generation interval. Also, set the NAK generation interval to at least 4x the delivery control loss check interval.
- for LBT-RU, set the transport activity timeout to value greater than the delivery control loss check interval
- for UMP, always enable and set accordingly the delivery control loss check interval when configuring a store

For more, see *Delivery Control Options*.

Example:

```
#
# +-------------------------------------------------------------------------+
# | To avoid undetected or unreported loss, set NAK generation to 4x the delivery|
# | control check interval, and ensure that these two combined are less than the |
# | transport activity timeout                                              |
```

```
# +---------------------------------------------------------------------------+
#
receiver delivery_control_loss_check_interval     2500
receiver transport_lbtrm_activity_timeout        60000
receiver transport_lbtrm_nak_generation_interval 10000
```

CHAPTER 3

Common Tasks

This chapter includes the following topics:

Configuring Multi-Homed Hosts

By default, UM will select the first multicast-capable, non-loopback interface for multicast topic resolution. If you are fortunate, on a multi-homed host, the correct interface will be selected. However, this fortuitous selection should *not* be relied upon. Moving the interface card to a different slot, a change in the operating system kernel, and numerous other factors can lead to a different ordering of interfaces as reported by the operating system. This in turn can lead UM to a select a different interface after the change.

It is strongly recommended that the actual interface be specified. The “ resolver_multicast_interface (context) ” on page 88 option allows you to explicitly specify the multicast interface. Note that this also changes the interface for LBT-RM and multicast immediate messaging.

Other options of interest include

- “ resolver_unicast_interface (context) ” on page 92 when using the unicast resolver
- “ request_tcp_interface (context) ” on page 167 when using the request/response messaging
- “ transport_lbtru_interface (receiver) ” on page 119 and “ transport_tcp_interface (receiver) ” on page 94 for receivers
- “ transport_lbtru_interface (source) ” on page 119 and “ transport_tcp_interface (source) ” on page 95 for sources

Traversing a Firewall

To use UM across a firewall, several port options may need to be changed. The options of interest include:

Multicast resolver: “ resolver_multicast_port (context) ” on page 89.

Unicast resolver:

- “ resolver_unicast_port (context) ” on page 227

- “ resolver_unicast_port_low (context) ” on page 93
- “ resolver_unicast_port_high (context) ” on page 93
- “ resolver_unicast_destination_port (context) ” on page 226

TCP transport:

- “ transport_tcp_port_low (context) ” on page 96 for contexts
- “ transport_tcp_port_high (context) ” on page 96 for contexts
- “ transport_tcp_port (source) ” on page 95 for sources

LBT-RM transport:

- “ transport_lbtrm_source_port_low (context) ” on page 105 for contexts
- “ transport_lbtrm_source_port_high (context) ” on page 105 for contexts
- “ transport_lbtrm_destination_port (source) ” on page 104

LBT-RU transport:

- “ transport_lbtru_port_low (context) ” on page 121 for contexts
- “ transport_lbtru_port_high (context) ” on page 120 for contexts
- “ transport_lbtru_port (source) ” on page 120 for sources
- “ transport_lbtru_port_low (receiver) ” on page 121 for receivers
- “ transport_lbtru_port_high (receiver) ” on page 120 for receivers

Multicast immediate messaging:

- “ mim_destination_port (context) ” on page 147
- “ mim_incoming_destination_port (context) ” on page 148
- “ mim_outgoing_destination_port (context) ” on page 148

Requests:

- “ request_tcp_port (context) ” on page 167
- “ request_tcp_port_low (context) ” on page 168
- “ request_tcp_port_high (context) ” on page 167

In addition, since machines acting as a firewall are often multi-homed as well, consult the section on “Configuring Multi-Homed Hosts” on page 49 for additional considerations.

Running Multiple Applications

If you are running multiple UM applications on the same machine, using the same (or the default) configuration, you may encounter problems due to the way UM allocates and uses ports. The UM Knowledgebase contains an article on Address and Port Usage which explains how to handle this situation.

CHAPTER 4

Reference

This chapter includes the following topics:

Introduction

Case Sensitivity

All Ultra Messaging scope, option, and value strings are case-insensitive. Thus, any of **context**, **CONTEXT**, and **Context** are recognized as specifying the "context" scope.

Specifying Interfaces

The *_interface options require a network interface, usually supplied as a string (from a config file via **lbm_config()** or in source code via ***_attr_str_setopt()**), the syntax used for network interface specifications is:

a.b.c.d/num

where *num* is the number of leading 1 bits in the netmask. If the */num* is omitted, it defaults to 32 (netmask 255.255.255.255), which means that it must be an exact match for the interface's IP address. However, if */num* is supplied, it tells Ultra Messaging to find an interface within that network. This makes it easier to share a configuration file between many (possibly multi-homed) machines on the same network. For example:

```
context resolver_unicast_interface 192.168.0.0/24
```

specifies a netmask of 255.255.255.0 and would match the interface **192.168.0.3** on one host, and **192.168.0.251** on another host. You can also set network interfaces by name. When setting a configuration option's interface by name, you must use quotes, as illustrated below.

```
context resolver_unicast_interface "interfacename"
```

Socket Buffer Sizes

When specifying send or receive socket buffer sizes, keep the following platform-specific information in mind.

- Linux

  The kernel value **net.core.rmem_max** dictates the highest value allowed for a receive socket. The kernel value **net.core.wmem_max** dictates the highest value allowed for a sending socket. Increase these values to increase the amount of buffering allowed.

- Windows

  Windows should allow socket buffer sizes to be set very high if needed without requiring registry changes.

See our whitepaper Topics in High Performance Messaging for background and guidelines on UDP buffer sizing.

Reference Entry Format

This section describes the format of each option reference entry.

Each entry begins with a brief description of the option. Following the description is a series of items that defines permissible usage and describes the values for the option.

- Scope

  Defines the scope to which the option applies.

- Type

  Defines the data type of the option. The type is required for calls to the ***_setopt()** and ***_getopt()** API functions.

- Units

  Defines the units in which the option value is expressed. This item is optional.

- Default value

  For range-valued options, indicates the base default value for the option.

- Byte order

  For options whose value is an IP address or port, defines the byte ordering (Host or Network) expected by the API for ***_setopt()** calls, and returned by the API for ***_getopt()** calls.

- May be set during operation

  If an option may be set during operation, it is so indicated here.

Next, for enumerated-valued options with limited specific choices, a table details the permissible **String Value** (configuration file), **Integer Value** (programmatic attribute setting), and a **Description** of each choice that includes default value designations.

Alternately, for switch-valued options (0 or 1), a table describes the meaning of each of the two possible values. The default value is noted within the description.

Network Compatibility Mode

This section lists the values for the “ network_compatibility_mode (context) ” on page 60 option, that attempts to maintain wire-level backwards compatibility with older releases by blocking the sending of some (though possibly not all) newer message header types. An application using an older UM release typically logs a warning message when receiving an unknown message header type that did not yet exist in that older release. In a mixed UM version environment, Informatica recommends that your applications filter these unknown message header warning log messages. This option should only be used if such filtering is undesired or not possible.

Note that this option does not change any internal behaviors. It merely prevents the sending new message header types which disables any new functionality that relies on the new message header types for both old

and new applications. Other than the warning log message for an application using an earlier release, new message header types do not cause any harm.

String Value	Integer Value	Description
default	LBM_CTX_ATTR_NET_COMPAT_MODE_DEFAULT	Network compatibility mode is disabled. UM sends all new message header types.
LBM_3.6	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_3_6	Block any message headers that only an LBM 3.6 or newer application would understand.
LBM_3.6.1	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_3_6_1	Block any message headers that only an LBM 3.6.1 or newer application would understand.
LBM_3.6.2	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_3_6_2	Block any message headers that only an LBM 3.6.2 or newer application would understand.
LBM_3.6.5	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_3_6_5	Block any message headers that only an LBM 3.6.5 or newer application would understand.
LBM_4.0	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_0	Block any message headers that only an LBM 4.0 or newer application would understand.
LBM_4.0.1	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_0_1	Block any message headers that only an LBM 4.0.1 or newer application would understand.
LBM_4.1	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_1	Block any message headers that only an LBM 4.1 or newer application would understand.
LBM_4.1.1	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_1_1	Block any message headers that only an LBM 4.1.1 or newer application would understand.
LBM_4.1.2	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_1_2	Block any message headers that only an LBM 4.1.1 or newer application would understand.
LBM_4.1.3	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_1_3	Block any message headers that only an LBM 4.1.1 or newer application would understand.
LBM_4.2.1	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_1	Block any message headers that only an LBM 4.2.1 or newer application would understand.
LBM_4.2.2	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_2	Block any message headers that only an LBM 4.2.2 or newer application would understand.

String Value	Integer Value	Description
LBM_4.2.3	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_3	Block any message headers that only an LBM 4.2.3 or newer application would understand.
LBM_4.2.4	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_4	Block any message headers that only an LBM 4.2.4 or newer application would understand.
LBM_4.2.5	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_5	Block any message headers that only an LBM 4.2.5 or newer application would understand.
LBM_4.2.6	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_6	Block any message headers that only an LBM 4.2.6 or newer application would understand.
LBM_4.2.7	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_7	Block any message headers that only an LBM 4.2.7 or newer application would understand.
LBM_4.2.8	LBM_CTX_ATTR_NET_COMPAT_MODE_LBM_4_2_8	Block any message headers that only an LBM 4.2.8 or newer application would understand.
UME_3.0	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_0	Block any message headers that only an UME 3.0 or newer application would understand.
UME_3.0.1	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_0_1	Block any message headers that only an UME 3.0.1 or newer application would understand.
UME_3.0.2	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_0_2	Block any message headers that only an UME 3.0.2 or newer application would understand.
UME_3.1	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_1	Block any message headers that only an UME 3.1 or newer application would understand.
UME_3.1.1	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_1_1	Block any message headers that only an UME 3.1.1 or newer application would understand.
UME_3.1.2	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_1_2	Block any message headers that only an UME 3.1.2 or newer application would understand.
UME_3.1.3	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_1_3	Block any message headers that only an UME 3.1.3 or newer application would understand.
UME_3.2.1	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_1	Block any message headers that only an UME 3.2.1 or newer application would understand.

String Value	Integer Value	Description
UME_3.2.2	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_2	Block any message headers that only an UME 3.2.2 or newer application would understand.
UME_3.2.3	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_3	Block any message headers that only an UME 3.2.3 or newer application would understand.
UME_3.2.4	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_4	Block any message headers that only an UME 3.2.4 or newer application would understand.
UME_3.2.5	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_5	Block any message headers that only an UME 3.2.5 or newer application would understand.
UME_3.2.6	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_6	Block any message headers that only an UME 3.2.6 or newer application would understand.
UME_3.2.7	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_7	Block any message headers that only an UME 3.2.7 or newer application would understand.
UME_3.2.8	LBM_CTX_ATTR_NET_COMPAT_MODE_UME_3_2_8	Block any message headers that only an UME 3.2.8 or newer application would understand.
UMQ_1.0	LBM_CTX_ATTR_NET_COMPAT_MODE_UMQ_1_0	Block any message headers that only an UMQ 1.0 or newer application would understand.
UMQ_1.1	LBM_CTX_ATTR_NET_COMPAT_MODE_UMQ_1_1	Block any message headers that only an UMQ 1.1 or newer application would understand.
UMQ_1.1.1	LBM_CTX_ATTR_NET_COMPAT_MODE_UMQ_1_1_1	Block any message headers that only an UMQ 1.1.1 or newer application would understand.
UMQ_2.0	LBM_CTX_ATTR_NET_COMPAT_MODE_UMQ_2_0	Block any message headers that only an UMQ 2.0 or newer application would understand.
UMQ_2.0.1	LBM_CTX_ATTR_NET_COMPAT_MODE_UMQ_2_0_1	Block any message headers that only an UMQ 2.0.1 or newer application would understand.
UMQ_2.1.1	LBM_CTX_ATTR_NET_COMPAT_MODE_UMQ_2_1_1	Block any message headers that only an UMQ 2.1.1 or newer application would understand.
UMQ_2.1.3	LBM_CTX_ATTR_NET_COMPAT_MODE_UMQ_2_1_3	Block any message headers that only an UMQ 2.1.3 or newer application would understand.

String Value	Integer Value	Description
UMQ_2.1.4	LBM_CTX_ATTR_NET_COMPAT_MODE_UMQ_2_1_4	Block any message headers that only an UMQ 2.1.4 or newer application would understand.
UMQ_2.1.5	LBM_CTX_ATTR_NET_COMPAT_MODE_UMQ_2_1_5	Block any message headers that only an UMQ 2.1.5 or newer application would understand.
UMQ_2.1.6	LBM_CTX_ATTR_NET_COMPAT_MODE_UMQ_2_1_6	Block any message headers that only an UMQ 2.1.6 or newer application would understand.
UMQ_2.1.7	LBM_CTX_ATTR_NET_COMPAT_MODE_UMQ_2_1_7	Block any message headers that only an UMQ 2.1.7 or newer application would understand.
UMQ_2.1.8	LBM_CTX_ATTR_NET_COMPAT_MODE_UMQ_2_1_8	Block any message headers that only an UMQ 2.1.8 or newer application would understand.
UMQ_2.1.9	LBM_CTX_ATTR_NET_COMPAT_MODE_UMQ_2_1_9	Block any message headers that only an UMQ 2.1.9 or newer application would understand.
UMQ_2.1.10	LBM_CTX_ATTR_NET_COMPAT_MODE_UMQ_2_1_10	Block any message headers that only an UMQ 2.1.10 or newer application would understand.
UM_5.0	LBM_CTX_ATTR_NET_COMPAT_MODE_UM_5_0	Block any message headers that only an UM 5.0 or newer application would understand.
UM_5.0.1	LBM_CTX_ATTR_NET_COMPAT_MODE_UM_5_0_1	Block any message headers that only an UM 5.0.1 or newer application would understand.
UM_5.1	LBM_CTX_ATTR_NET_COMPAT_MODE_UM_5_1	Block any message headers that only an UM 5.1 or newer application would understand.
UM_5.1.1	LBM_CTX_ATTR_NET_COMPAT_MODE_UM_5_1_1	Block any message headers that only an UM 5.1.1 or newer application would understand.
UM_5.1.2	LBM_CTX_ATTR_NET_COMPAT_MODE_UM_5_1_2	Block any message headers that only an UM 5.1.2 or newer application would understand.
UM_5.2	LBM_CTX_ATTR_NET_COMPAT_MODE_UM_5_2	Block any message headers that only an UM 5.2 or newer application would understand.

String Value	Integer Value	Description
UM_5.2.1	LBM_CTX_ATTR_NET_COMPAT_MODE_UM_5_2_1	Block any message headers that only an UM 5.2.1 or newer application would understand.
UM_5.2.2	LBM_CTX_ATTR_NET_COMPAT_MODE_UM_5_2_2	Block any message headers that only an UM 5.2.2 or newer application would understand.

Major Options

Options in this group have a major impact on the operation of Ultra Messaging. Most UM application developers will need to be aware of the default values of these options or perhaps override them.

compatibility_include_pre_um_6_0_behavior (context)

Enable Ultra Messaging 6.x applications to inter-operate with pre-6.0 applications. Enabling this option increases overhead data on the wire and slightly changes some operational behaviors of UMP sources.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.7

Value	Description
"1" (Integer value as a string.)	Inter-operate with pre-6.0 applications.
"0" (Integer value as a string.)	Disable Inter-operation with pre-6.0 applcations. **Default for all.**

context_event_function (context)

Callback function (and associated client data pointer) that is called when a context event occurs. This callback may be called inline or from an event queue, if one is given. If called inline, the callback function used should not block or it will block the context thread processing. A value of NULL for the callback turns off the callback being called.

Scope:	context
Type:	**string**

When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMQ 1.0.

context_name (context)

The name of the context, limited to 128 alphanumeric characters, hyphens or underscores.

Scope:	context
Type:	**string**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.3/UMP 3.3/UMQ 2.3.

fd_management_type (context)

Define the mechanism UM uses for socket file descriptor (FD) management. For more information, search on "file descriptors" in the Informatica Knowledge Base.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
poll	**LBM_CTX_ATTR_FDTYPE_POLL**	FD management uses **poll()**. *Unix only*.
select	**LBM_CTX_ATTR_FDTYPE_SELECT**	FD management uses **select()**. **Default for Unix.** *Unix only*.
epoll	**LBM_CTX_ATTR_FDTYPE_EPOLL**	FD management uses **epoll()**. *Linux kernel 2.6 or later only*.
devpoll	**LBM_CTX_ATTR_FDTYPE_DEVPOLL**	FD management uses the **/dev/poll** driver. *Solaris 8 or later only*.
kqueue	**LBM_CTX_ATTR_FDTYPE_KQUEUE**	FD management uses the BSD **kqueue** notification system. *Mac OS X only*.

String value	Integer value	Description
wsaeventselect	**LBM_CTX_ATTR_FDTYPE_WSAEV**	FD management uses **WSAEventSelect()** and **WaitForMultipleObjects()**. Creates a limit of 64 file descriptors. **Default for Windows.** *Windows only*.
wincompport	**LBM_CTX_ATTR_FDTYPE_WINCPORT**	FD management uses Windows completion ports and completion routines. Disables the 64 file descriptor limit set by **WSAEventSelect()**. *Windows XP or later only*.

message_selector (receiver)

Enables UM to pass a message selector string to any receiver. The value must be an expression that conforms to JMS message selector syntax as defined in the Oracle JMS specification. For a UM receiver used with UMP, please see *Native Applications* in the *Ultra Messaging JMS Guide*.

Scope:	receiver
Type:	**string**
Default value:	NULL
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMQ 5.3.

network_compatibility_mode (context)

This option attempts to maintain wire-level backwards compatibility with older releases by blocking the sending of some (though possibly not all) newer message header types. See "Network Compatibility Mode" on page 53 for more information and option values.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2/UME 3.2/UMQ 2.1.

operational_mode (context)

The mode in which UM operates to process events. Refer to *Embedded and Sequential Mode* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
embedded	**LBM_CTX_ATTR_OP_EMBEDDED**	A thread is spawned within UM to handle processing of events (timers and socket events). **Default for all.**
sequential	**LBM_CTX_ATTR_OP_SEQUENTIAL**	The application is responsible for calling **lbm_context_process_events()** to process events. Sequential mode does not support *Multi-Transport Threads*.

ordered_delivery (receiver)

For LBT-RM, LBT-RU, TCP-LB or LBT-IPC transport sessions only. (This option also applies to TCP when using Late Join because the Late Join messages are not part of the TCP message stream.) Indicates whether or not the topic should have its data delivered in order and reassembled. The default value guarantees ordering and reassembly of large messages. Reassembly of large messages is optional. Changing this option from the default value to a value of 0 (zero) results in messages being delivered as soon as they arrive. Value -1 allows arrival order delivery after the reassembly of large messages. See also *Ordered Delivery* in

the *Ultra Messaging Concepts Guide* for more information about large message fragmentation and reassembly.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
"1" (Integer value as a string.)	**1**	UM delivers topic messages to a receiver in-order and reassembles large messages. **Default for all.**
"0" (Integer value as a string.)	**0**	UM delivers topic messages to a receiver as they arrive and may be out of order. Duplicate delivery is possible. UM delivers large messages as individual fragments of less than the maximum datagram size for the transport in use.
"-1" (Integer value as a string.)	**-1**	UM delivers topic messages to a receiver as they arrive and may be out of order. Duplicate delivery is possible. However, UM reassembles large messages. Your application can use the `sequence_number` field of `lbm_msg_t` objects to order or discard messages.

rcv_sync_cache (receiver)

Ultra Messaging Cache only - a valid cache address (such as TCP:192.168.5.11:4567) in the standard form of `TCP:address:port` enables a UM receiver to use UMCache to receive a snapshot of larger, multiple-field messages stored by UMCache. Receiving applications can then become synchronized with the live stream of messages sent on the receiver's topic. `address` is the IP address of the machine where the UMCache runs and `port` is the configured port where the cache request handler listens.

Scope:	receiver
Type:	**umcache_reqlib_request_info_t**
Default value:	NULL
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMS 5.0/UMP 5.0/UMQ 5.0

rcv_sync_cache_timeout (receiver)

Ultra Messaging Cache only - The maximum time period that a UM receiver waits for a snapshot message from the UMCache .

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	2000 (2 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMS 5.0/UMP 5.0/UMQ 5.0

receive_thread_pool_size (context)

For LBT-RM, LBT-RU, or TCP-LB transport sessions only. Defines the maximum number of threads available for transports (excluding the context thread). See *Multi-Transport Threads* in the *Ultra Messaging Concepts Guide*.

Scope:	context
Type:	**int**
Default value:	4
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1/UME 3.1.

receiver_callback_service_time_enabled (context)

Indicates if UM collects receiver callback statistics, which provide the maximum, mean and minimum time in microseconds required to complete wildcard, hot-failover, and regular receiver callbacks.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.5

Value	Description
1	UM collects receiver callback statistics.
0	UM collects receiver callback statistics. **Default for all.**

resolver_source_notification_function (context)

Callback function (and associated client data pointer) that is called when a new source is seen for any topic. This callback is called directly in line and does not use the event queue. Therefore the callback function used should not block or it will block the context thread processing. A value of NULL for the callback turns off the callback being called.

Scope:	context
Type:	**lbm_src_notify_func_t**
Default value:	NULL
When to Set:	Can only be set during object initialization.
Config File:	Cannot be set from an UM configuration file.

source_cost_evaluation_function (context)

Callback function that you can use in the `lbm_src_cost_function_cb()` to evaluate or determine the cost of a message path. The UM Router evaluates the cost of any new topic it detects. The callback supplied with this option can affect the cost of topics to bias the UM Router toward certain message paths. A value of NULL for the callback turns off the callback being called. See also *Applications Can Also Set the Topic Cost* in the *UM Dynamic Routing Guide*.

Scope:	context
Type:	**lbm_src_cost_func_t**
Default value:	NULL
When to Set:	Can only be set during object initialization.
Config File:	Cannot be set from an UM configuration file.
Version:	This option was implemented in UMS 5.0/UMP 5.0/UMQ 5.0

source_event_function (context)

Callback function (and associated client data pointer) that is called when a context source event (such as a multicast immediate mode source wakeup event) occurs. This callback may be called inline or from an event queue, if one is given. If called inline, the callback function used should not block or it will block the context thread processing. A value of NULL for the callback turns off the callback being called.

Scope:	context
Type:	**lbm_context_src_event_func_t**
Default value:	NULL
When to Set:	Can only be set during object initialization.

Config File:	Cannot be set from an UM configuration file.
Version:	This option was implemented in LBM 3.4/UME 2.1.

source_includes_topic_index (context)

Determines whether the topic index is included in the source string generated for messages and new source notifications.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.6/UME 3.0.

Value	Description
1	Indicates the topic index should be included in the source string. **Default for all.**
0	Indicates the topic index should not be included.

transport (source)

The transport type to be used for created sources.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
tcp	**LBM_SRC_TOPIC_ATTR_TRANSPORT_TCP**	TCP over IPv4 **Default for all.**
lbtrm, **lbt-rm**	**LBM_SRC_TOPIC_ATTR_TRANSPORT_LBTRM**	UDP-based reliable multicast with unicast NAKs
lbtru, **lbt-ru**	**LBM_SRC_TOPIC_ATTR_TRANSPORT_LBTRU**	UDP-based reliable unicast with unicast NAKs
lbtipc, **lbt-ipc**	**LBM_SRC_TOPIC_ATTR_TRANSPORT_LBTIPC**	InterProcess Communication between processes on the same host using a shared memory area.

String value	Integer value	Description
lbtsmx, **lbt-smx**	**LBM_SRC_TOPIC_ATTR_TRANSPORT_LBTSMX**	Shared Memory Acceleration. Modified InterProcess Communication transport between processes on the same host using a shared memory area. Restricted to streaming applications.
lbtrdma, **lbt-rdma**	**LBM_SRC_TOPIC_ATTR_TRANSPORT_LBTRDMA**	Voltaire™ InfiniBand Remote Direct Memory Access transport between hosts using a shared memory area.

transport_demux_tablesz (receiver)

Specifies the size of the table used for storing receiver delivery controllers used by UM for message delivery. Must be a power of two (1, 2, 4, 8, 16, etc.). If not a power of two, UM generates a log warning and uses the next highest power of two. For most use cases with low to moderate numbers of topics per transport session, the default suffices. For large numbers of topics and in cases where the lowest latency is desired, set the option to the next highest power of two for the number of topics expected on the transport session.

Scope:	receiver
Type:	**size_t**
Default value:	1
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2/UME 3.2.

transport_session_multiple_sending_threads (context)

Flag that indicates the application intends to use multiple sending threads per transport session.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	Indicates the application does intend to use multiple sending threads per transport session and that UM should make that assumption. **Default for all.**
0	Indicates the application does not intend to use multiple sending threads per transport session and that UM should make that assumption.

transport_source_side_filtering_behavior (source)

The filtering behavior desired when TCP and LBT-RU clients are connected. Any other value besides none requires that the clients send unicast messages to the source. These control messages are sent to the TCP request port of the senders context and processed internally. This option affects the transport session underlying the source rather than the source itself. Refer to *Source Configuration and Transport Sessions* for additional information.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
none	**LBM_SRC_TOPIC_ATTR_SSF_NONE**	The source sends all data to all clients regardless of the topics they are listening to. **Default for all.**
inclusion	**LBM_SRC_TOPIC_ATTR_SSF_INCLUSION**	The source sends only that data to a client that the client specifically requests.

transport_topic_sequence_number_info_active_threshold (source)

Duration in seconds that an inactive source sends contiguous Topic Sequence Number Info (TSNI) messages. (Inactive sources send TSNI messages according to the " transport_topic_sequence_number_info_interval (source) " on page 67.) A value of 0 indicates that sources continue sending TSNIs until data messages resume, with no timeout. See also "Interrelated Configuration Options" on page 42.

Scope:	source
Type:	**lbm_ulong_t**
Units:	seconds
Default value:	60
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

transport_topic_sequence_number_info_interval (source)

The interval between Topic Sequence Number Info (TSNI) messages that a source sends. TSNI messages are enabled on all transports except LBT-SMX, and they carry the topic sequence number of the latest message sent by the source. The interval is also a source inactivity threshold. In other words, a source does not send TSNIs during normal data transmission, but once the source is inactive for as long as this interval, it

starts sending TSNI messages. A value of 0 turns off TSNI messages for the source. See also “Interrelated Configuration Options” on page 42.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	5000 (5 second)
When to Set:	Can only be set during object initialization.

transport_topic_sequence_number_info_request_interval (receiver)

The interval at which the receiver requests a Topic Sequence Number Information Record (TSNI) from the source. Controlling these requests helps reduce receiver start-up traffic on your network.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	1000 (1 second)
When to Set:	Can only be set during object initialization.
Version	This option was implemented in UMP 6.0

transport_topic_sequence_number_info_request_maximum (receiver)

The maximum number of requests the receiver issues for a Topic Sequence Number Information Record (TSNI) from the source. If the receiver has not received an TSNI after this number of requests, it stops requesting.

Scope:	source
Type:	**lbm_ulong_t**
Default value:	60
When to Set:	Can only be set during object initialization.
Version	This option was implemented in UMP 6.0

use_extended_reclaim_notifications (source)

Specifies which reclaim notification your application receives. The expanded notification, LBM_SRC_EVENT_UME_MESSAGE_RECLAIMED_EX, contains a flag, LBM_SRC_EVENT_UME_MESSAGE_RECLAIMED_EX_FLAG_FORCED that UMP sets if the reclamation is a forced reclaim.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2.

Value	Description
1	Indicates your application receives the expanded reclaim notification. **Default for all.**
0	Indicates your application receives the standard reclaim notification that is identical to the expanded notification but without the "Forced" flag.

use_transport_thread (receiver)

For LBT-RM, LBT-RU, or TCP-LB transport sessions only. Determines whether UM uses a thread from the receiver thread pool to process message data or if it uses the context thread, which is the default. See *Multi-Transport Threads* in the *Ultra Messaging Concepts Guide*.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1/UME 3.1.

String value	Integer value	Description
"1" (Integer value as a string.)	**1**	UM uses a thread from the receiver thread pool.
"0" (Integer value as a string.)	**0**	UM uses the context thread to process message data. **Default for all.**

Resolver Operation Options

See *Topic Resolution* in the *UM Concepts Guide* for more information.

The following topic resolution options have been deprecated in LBM Version 4.0.

- `resolver_active_source_interval`
- `resolver_active_threshold`
- `resolver_maximum_advertisements`
- `resolver_maximum_queries`
- `resolver_query_interval`

See "Re-establish Pre-4.0 Topic Resolution" on page 40 for option values that configure the topic resolution used in LBM Version 3.6 and prior versions. You should also comment out or remove from your Ultra Messaging Configuration file the deprecated configuration options shown above.

Minimum Values for Advertisement and Query Intervals

These intervals have the following effective minimal values.

- 10 ms for Initial Phase Advertisements
- 20 ms for Initial Phase Queries
- 30 ms Wildcard Queries
- 100 ms for Sustaining Phase Advertisements and Queries
- 100 ms for Context Name Queries

These effective minimums exist because the internal timer that schedules advertisements and queries fires at the stated interval, i.e., every 10 ms for Initial Phase Advertisements, every 20 ms for Initial Phase Queries, etc. If you set the option's value below the minimum, after the initial advertisement or query at 0 ms, the resolver schedules the second advertisement or query at the first timer "tick", which is the minimum. Subsequent advertisements or queries can only be issued at the next timer "tick". If you increase this option from the default to a value that is not a multiple of the minimum, the resolver maintains the rate you establish as an average over subsequent "ticks".

As an example, If you set " resolver_advertisement_sustain_interval (source) " on page 73 or " resolver_query_sustain_interval (receiver) " on page 80 at 10 ms, the resolver schedules the second advertisement or query after the initial (0 ms) at the first timer "tick", which is 100 ms. Subsequent advertisements or queries can only be issued at the next timer "tick" (every 100 ms). If you increase either option from the default to 1.25 seconds, for example and not a multiple of 100 ms, the resolver maintains the rate you establish as an average over subsequent "ticks". That is, the second advertisement or query goes out at the 1300 ms "tick". The resolver tracks the tardiness of this advertisement (50 ms) and adjusts the next advertisement or query, which goes out at 2500 ms, giving an average of 1250 ms or 1.25 seconds.

disable_extended_topic_resolution_message_options (context)

This is a topic resolution compatibility option that, when set to 1, lets LBM 4.0 (or later) installations work with LBM 3.5.3 / UME 2.2.4 (or earlier) installations. If you do not have early-version installations in the network, leave this option at 0.

Scope:	context
Type:	**int**

When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0.

Value	Description
1	Enable compatibility with earlier-version installations (and disable some message structure features).
0	Normal current-version compatibility. Strongly recommended. **Default for all.**

resolution_no_source_notification_threshold (receiver)

The threshold for the number of unanswered topic resolution queries before UM delivers a **LBM_MSG_NO_SOURCE_NOTIFICATION** for the topic. The receiver does not stop querying after the delivery of this notification. A value of 0 indicates no notifications will be sent.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	Number of queries
Default value:	0 (do not notify)
When to Set:	May be set during operation.

resolution_number_of_sources_query_threshold (receiver)

The threshold for the number of sources a topic must have before topic resolution queries are not sent. A value of zero results in no topic resolution queries being generated. See also "Disabling Aspects of Topic Resolution " on page 39.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	Number of sources
Default value:	10000000 (10 million)
When to Set:	May be set during operation.

resolver_advertisement_maximum_initial_interval (source)

The longest - and last - interval in the initial phase of topic advertisement. A value of 0 disables the initial phase of advertisement. See also “Disabling Aspects of Topic Resolution ” on page 39.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	500 (0.5 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_advertisement_minimum_initial_duration (source)

The duration of the initial phase of topic advertisement. A value of 0 guarantees that the initial phase of advertisement never completes.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	5000 (5 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_advertisement_minimum_initial_interval (source)

Interval between the first topic advertisement sent upon creation of the source and the second advertisement sent by the source. A value of 0 disables the initial phase of advertisement. See also “Disabling Aspects of Topic Resolution ” on page 39. This option has an effective minimum of 10 ms. See “Minimum Values for Advertisement and Query Intervals” on page 70.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10 (0.01 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_advertisement_minimum_sustain_duration (source)

The duration of the sustaining phase of topic advertisement. A value of 0 guarantees that the sustaining phase of advertising never completes.

Scope:	source
Type:	**lbm_ulong_t**
Units:	seconds
Default value:	60 (1 minute)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_advertisement_send_immediate_response (source)

Allows you to disable the normal immediate response to queries and wildcard queries. Sources normally send topic advertisements (TIR) immediately in response to topic queries (TQR) for a local topic or wildcard queries (WC-TQR) with a pattern that matches a local topic. If you configure sources to delay sending advertisements, UM delays advertisements by the limits defined by the advertisement rate limiter options, `resolver_*_bps` and `resolver_*_per_second`.

Scope:	source
Type:	**lbm_uint_t**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2/UME 3.2/UMQ 2.1

Value	Description
1	Sources immediately send advertisements (TIR) in response to topic queries (TQR) or wildcard queries (WC-TQR). **Default for all.**
0	Sources delay sending advertisements (TIR) in response to topic queries (TQR) or wildcard queries (WC-TQR).

resolver_advertisement_sustain_interval (source)

Interval between sending topic advertisements in the sustaining phase of topic advertisement. A value of 0 disables the sustaining phase of advertisement. See also “Disabling Aspects of Topic Resolution ” on page 39. This option has an effective minimum of 100 ms. See “Minimum Values for Advertisement and Query Intervals” on page 70.

Scope:	source
Type:	**lbm_ulong_t**

Units:	milliseconds
Default value:	1000 (1 second)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_cache (context)

Whether or not to enable the resolver cache to hold topic resolution information. Disabling the resolver cache uses less memory. Note that if you disable the resolver cache, source notification occurs for only topics with UM receivers already created.

To use wildcard receivers, you must enable the resolver cache.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	Topic resolution information will be cached. **Default for all.**
0	Do not cache topic resolution information.

resolver_context_name_activity_timeout (context)

Period of inactivity before a context name is declared unresolved.

The minimum amount of time without any context name resolution traffic that must pass before UM declares a resolved context name unresolved. Context name resolution traffic is defined as the reception of context name advertisements and/or unicast control traffic from the resolved context.

Scope:	source
Type:	**lbm_uint64_t**
Units:	milliseconds
Default value:	60000 (60 seconds)
When to Set:	Can only be set during object initialization.
Version	This option was implemented in UM 6.0.

resolver_context_name_query_duration (context)

Maximum period of time UM sends context name queries.

The maximum duration for which UM sends context name queries for a given context name. UM sends queries until the context name resolves. A value of 0 means queries have no time limit and UM continues to query until the context name resolves.

Scope:	receiver
Type:	**lbm_uint64_t**
Units:	milliseconds
Default value:	0 (Query for as long as unresolved)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.0.

resolver_context_name_query_maximum_interval (context)

The longest interval between sending context name queries. A value of 0 disables context name queries. See also “Disabling Aspects of Topic Resolution ” on page 39. This option has an effective minimum of 100 ms. See “Minimum Values for Advertisement and Query Intervals” on page 70

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	1000 (1 second)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.0.

resolver_context_name_query_minimum_interval (context)

Interval between the first context name query sent upon creation of the UMP source using named stores and the second query sent. A value of 0 disables context name queries. See also “Disabling Aspects of Topic Resolution ” on page 39. This option has an effective minimum of 100 ms. See “Minimum Values for Advertisement and Query Intervals” on page 70.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	100 (0.1 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.0 .

resolver_datagram_max_size (context)

The maximum datagram size that can be generated for topic resolution advertisements and queries. The default value is 8192, the minimum is 500 bytes, and the maximum is 65535.

Scope:	context
Type:	**lbm_uint_t**
Units:	bytes
Default value:	8192
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.6/UME 3.0.

resolver_domain_id_active_propagation_timeout (context)

Indicates how a context learns the ID of its own Topic Resolution Domain (TRD).

Scope:	context
Type:	**int**

When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.7.1.

Value	Description
-1	Learn TRD ID from other contexts in the same TRD, without restriction. This is the method Ultra Messaging has traditionally used. **Default for all.** This method assigns TRD IDs quickly to avoid partial connectivity. However, note that to change a TRD ID, you must reconfigure and restart all UM Routers, if present. Then you must delete all application contexts, and then re-create all application contexts. **Note:** With this option value, newly-created contexts can learn from earlier versions of Ultra Messaging software.
0	Learn TRD ID only from a UM Router directly. Do not learn the TRD ID from other contexts in the same TRD. Consider using this option with a TRD that has many contexts and a possible need to change a TRD ID.
1 to 2,147,483,647	Learn TRD ID from other contexts in the same TRD that have heard the domain ID advertised by a UM Router within this timeout value in milliseconds. Use the following formula: 3 * {*propagation-interval*}/1000 + {*maximum expected duration of UM Router outage*} where *propagation-interval* is an attribute value of the UM Router configuration option <route-info> element, which defaults to 1000. With a timeout value set, a restarted context does not learn obsolete TRD IDs from un-restarted contexts, but instead, learns from UM Routers or restarted contexts. You do not need to bring all contexts to a deleted state simultaneously before you re-create the first context. **Note:** During this timeout period, there is a risk for temporary incomplete connectivity in networks with no UM Routers.

resolver_initial_advertisement_bps (context)

Maximum advertisement rate during the initial phase of topic advertisement. A value of 0 sets no rate limit on advertisements in the initial phase of topic advertisement.

Scope:	context
Type:	**lbm_uint64_t**
Units:	bits per second
Default value:	1000000
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_initial_advertisements_per_second (context)

Maximum number of advertisements sent within a one second period during the initial phase of topic advertisement. A value of 0 sets no rate limit on advertisements in the initial phase of topic advertisement.

Scope:	context
Type:	**lbm_ulong_t**
Units:	advertisements
Default value:	1000
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_initial_queries_per_second (context)

Maximum number of queries sent within a one second period during the initial phase of topic querying. A value of 0 sets no rate limit on queries in the initial phase of topic querying.

Scope:	context
Type:	**lbm_ulong_t**
Units:	advertisements
Default value:	1000
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_initial_query_bps (context)

Maximum query rate during the initial phase of topic querying. A value of 0 sets no rate limit on queries in the initial phase of topic querying.

Scope:	context
Type:	**lbm_uint64_t**
Units:	bits per second
Default value:	1000000
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_query_maximum_initial_interval (receiver)

The longest - and last - interval in the initial phase of topic querying. A value of 0 disables the initial phase of querying. See also “Disabling Aspects of Topic Resolution ” on page 39.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	200 (0.2 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_query_minimum_initial_duration (receiver)

The duration of the initial phase of topic querying. A value of 0 guarantees that the initial phase of querying never completes.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	5000 (5 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_query_minimum_initial_interval (receiver)

Interval between the first topic query sent upon creation of the receiver and the second query sent by the receiver. A value of 0 disables the initial phase of querying. See also “Disabling Aspects of Topic Resolution ” on page 39. This option has an effective minimum of 20 ms. See “Minimum Values for Advertisement and Query Intervals” on page 70.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	20 (0.02 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_query_minimum_sustain_duration (receiver)

The duration of the sustaining phase of topic querying. A value of 0 guarantees that the sustaining phase of querying never completes.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	seconds
Default value:	60 (1 minute)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_query_sustain_interval (receiver)

Interval between sending topic queries in the sustaining phase of topic querying. A value of 0 disables the sustaining phase of querying. See also “Disabling Aspects of Topic Resolution ” on page 39. This option has an effective minimum of 100 ms. See “Minimum Values for Advertisement and Query Intervals” on page 70.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	1000 (1 second)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_receiver_map_tablesz (context)

The size of the hash table used for storing receiver topic information used for topic resolution. This value should be a prime number.

Scope:	context
Type:	**size_t**
Units:	map entries
Default value:	131111
When to Set:	Can only be set during object initialization.

resolver_send_initial_advertisement (source)

Controls whether or not a source sends an advertisement upon creation. Turning off this advertisement speeds source creation and reduces the number of messages on your network through application initialization.

Scope:	source
Type:	**lbm_uint_t**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

Value	Description
1	Source sends a topic advertisement immediately upon creation. **Default for all.**
0	Source does not send an advertisement upon creation. This setting does not affect the topic resolution phases you have configured, which execute as expected. See *Disabling Aspects of Topic Resolution* for information about altering topic resolution phase advertisements.

resolver_source_map_tablesz (context)

The size of the hash table used for storing source topic information used by topic resolution. This value should be a prime number.

Scope:	context
Type:	**size_t**
Units:	map entries
Default value:	131111
When to Set:	Can only be set during object initialization.

resolver_string_hash_function (context)

The hash function to use for hashing topic name strings for source and receiver topics. The application may choose from a list of defined hash functions or it may define its own hash function, as identified by the string value of this option. When setting a hash function, note that:

- If set through a configuration file or a call to **lbm_context_attr_str_setopt()**, only the string values **classic**, **djb2**, **sdbm**, or **murmur2** are valid. (If retrieved by a call to **lbm_context_attr_str_getopt()**, one of these string values is returned.)

- If set through a call to **lbm_context_attr_setopt()**, you must pass a pointer to a hash function. Use this method for hash functions other than the four pre-defined functions.

Scope:	context
Type:	**lbm_str_hash_func_t**
Default value:	NULL
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
classic		A "classic" good string hash function. Works best when topic names have a constant prefix with a changing suffix.
djb2		The Dan Bernstein algorithm from comp.lang.c. Works best when topic names have a changing prefix with a constant suffix.
sdbm		sdbm database library (used in Berkeley DB). A useful alternative to djb2.
murmur2		Good all-around hash function by Austin Appleby. Best for medium to long topic strings. **Default for all.**

resolver_string_hash_function_ex (context)

This option is similar to the " resolver_string_hash_function (context) " on page 81 above, except for the following differences:

- This option can be set via only **lbm_context_attr_setopt()** (not from a configuration file or **lbm_context_attr_str_setopt()**). Hence, this also means you cannot use the string options (**classic**, etc.).
- You can pass a string length to the hash function, allowing it to then possibly run faster by operating on multiple-character strings at a time. Note that if -1 is passed in, you must use a strlen to calculate the length.
- The hash function accepts a clientd pointer, which you can set as needed, and which is passed back in each time the function is called.

This option is the better choice when setting your own custom hash function. Note that both the **resolver_string_hash_function** and **resolver_string_hash_function_ex** options set the same attributes, hence, if you use both (not recommended) one will override the other.

Scope:	context
Type:	**lbm_str_hash_func_ex_t**

Default value:	NULL
When to Set:	Can only be set during object initialization.
Config File:	Cannot be set from an UM configuration file.

resolver_sustain_advertisement_bps (context)

Maximum advertisement rate during the sustaining phase of topic advertisement. A value of 0 sets no rate limit on advertisements in the sustaining phase of topic advertisement.

Scope:	context
Type:	**lbm_uint64_t**
Units:	bits per second
Default value:	1000000
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_sustain_advertisements_per_second (context)

Maximum number of advertisements sent within a one second period during the sustaining phase of topic advertisement. A value of 0 sets no rate limit on advertisements in the sustaining phase of topic advertisement.

Scope:	context
Type:	**lbm_ulong_t**
Units:	advertisements
Default value:	0
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_sustain_queries_per_second (context)

Maximum number of queries sent within a one second period during the sustaining phase of topic querying. A value of 0 sets no rate limit on queries in the sustaining phase of topic querying.

Scope:	context
Type:	**lbm_ulong_t**

Units:	advertisements
Default value:	0
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_sustain_query_bps (context)

Maximum query rate during the sustaining phase of topic querying. A value of 0 sets no rate limit on queries in the sustaining phase of topic querying.

Scope:	context
Type:	**lbm_uint64_t**
Units:	bits per second
Default value:	1000000
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_unicast_activity_timeout (context)

Indicates the maximum time between messages from a unicast resolver daemon before UM declares it inactive and stops sending normal topic resolution traffic via that daemon. UM will still send keepalives to the daemon. A value of 0 will force all resolver daemons to be treated as permanently active.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	1000
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMS 5.0

resolver_unicast_change_interval (context)

Indicates how often UM will change to the next available resolver daemon specified using the. resolver_unicast_daemon configuration option. The actual value used is random, and is selected from the range (1/2*change_interval, 3/2*change_interval). If all resolver daemons have been marked inactive, UM

enters a quick-change mode where it uses a random value from the range (1/4*change_interval, 3/4*change_interval) in order to more quickly locate an active daemon.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	200
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMS 5.0

resolver_unicast_check_interval (context)

Indicates how often a UM checks for resolver activity in order to determine liveness. A value of 0 will disable activity checks. This setting only applies to the unicast resolver.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	200
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMS 5.0

resolver_unicast_force_alive (context)

Indicates whether sources or receivers in this context should send keepalive messages to a configured Unicast Topic Resolver so they can receive topic resolution traffic.

Scope:	context
Type:	**lbm_uint16_t**
When to Set:	Can only be set during object initialization.

Value	Description
1	Contexts send keepalive messages to the Unicast Resolver at the " resolver_unicast_keepalive_interval (context) " on page 86 regardless of whether the context has any sources or receivers that require topic resolution.
0	Contexts do not send keepalive messages to the Unicast Resolver until sources or receivers have been created. Then Contexts send keepalives at the " resolver_unicast_keepalive_interval (context) " on page 86. **Default for all.**

resolver_unicast_ignore_unknown_source (context)

Indicates whether contexts using unicast topic resolution accept topic resolution udp datagrams that originate from any lbmrd or only the specific lbmrd configured for use.

Note: Do not modify this setting without guidance from Informatica.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.7.1.

Value	Description
0	A context using unicast topic resolution accepts traffic from lbmrd resolver daemons not configured for use by the context.
1	Contexts using unicast topic resolution accept topic resolution udp datagrams that originate from only the specific lbmrd configured for use. The context discards topic resolution datagrams from unrecognized origins and logs a message. This prevents applications at the same IP address, but in different topic resolution domains that might share resolver unicast port ranges, from processing unintended topic resolution traffic while lbmrd resolver daemons time out stale client entries. **Default for all.**

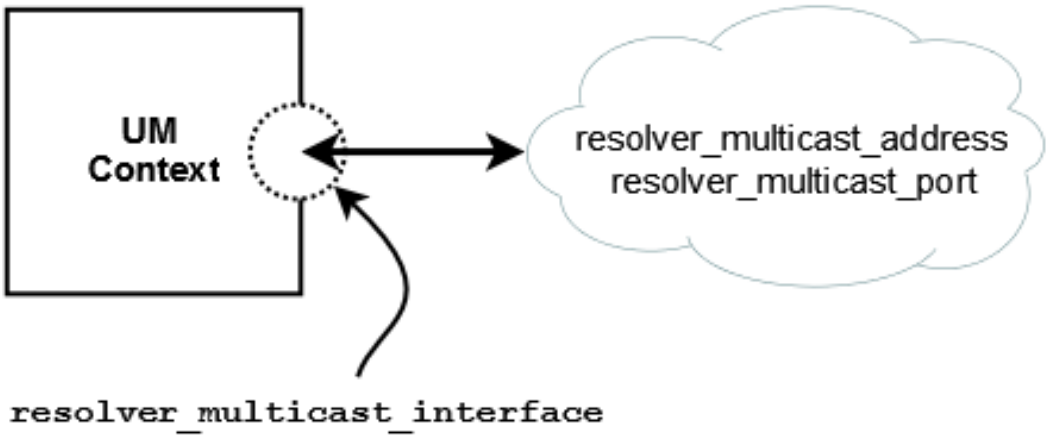
resolver_unicast_keepalive_interval (context)

Indicates how often keepalive messages should be sent to a resolver daemon. Keepalives are only sent if no other traffic has been sent since the last keepalive interval expired.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	500
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMS 5.0

Multicast Resolver Network Options

Multicast resolver network options



See also *Topic Resolution* in the *Ultra Messaging Concepts Guide* for more information.

resolver_multicast_address (context)

Multicast address used for Topic Resolution. This option automatically sets the values for " resolver_multicast_incoming_address (context) " on page 87 and " resolver_multicast_outgoing_address (context) " on page 88 as evidenced by the default values for all three options, which are the same.

Scope:	context
Type:	**struct in_addr**
Default value:	224.9.10.11
When to Set:	Can only be set during object initialization.

resolver_multicast_incoming_address (context)

Incoming multicast address used for finer control of Topic Resolution. For example, if you want the context to listen on a different address/port than the " resolver_multicast_address (context) " on page 87, set this option and " resolver_multicast_incoming_port (context) " on page 88 to different values. This value may be set to 0.0.0.0 (**INADDR_ANY**), to switch off listening to topic resolution messages. This means that queries from receivers or advertisements from sources will not be handled. See also " resolver_multicast_outgoing_address (context) " on page 88.

Scope:	context
Type:	**struct in_addr**

Default value:	224.9.10.11
When to Set:	Can only be set during object initialization.

resolver_multicast_incoming_port (context)

Incoming multicast port used for finer control of Topic Resolution. For example, if you want the context to listen on a different address/port than the “ resolver_multicast_port (context) ” on page 89, set this option and “ resolver_multicast_incoming_address (context) ” on page 87 to different values. See also “ resolver_multicast_outgoing_port (context) ” on page 89.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	12965
Byte order:	Network
When to Set:	Can only be set during object initialization.

resolver_multicast_interface (context)

Specifies which network interface UM sends/receives all multicast traffic (Topic Resolution, LBT-RM, Multicast Immediate Messaging). Can specify full IP address of interface, or just network part (see “Specifying Interfaces” on page 52 for details). Default is set to default multicast interface as determined by UM (the first multicast-capable, non-loopback interface).

Scope:	context
Type:	**lbm_ipv4_address_mask_t**
Default value:	0.0.0.0
When to Set:	Can only be set during object initialization.

resolver_multicast_outgoing_address (context)

Outgoing multicast address used for finer control of Topic Resolution. For example, if you want the context to send on a different address/port than the “ resolver_multicast_address (context) ” on page 87, set this option and “ resolver_multicast_outgoing_port (context) ” on page 89 to different values. See also “ resolver_multicast_incoming_address (context) ” on page 87.

Scope:	context
Type:	**struct in_addr**

Default value:	224.9.10.11
When to Set:	Can only be set during object initialization.

resolver_multicast_outgoing_port (context)

Outgoing multicast port used for finer control of Topic Resolution. For example, if you want the context to send on a different address/port than the " resolver_multicast_port (context) " on page 89, set this option and " resolver_multicast_outgoing_address (context) " on page 88 to different values. See also " resolver_multicast_incoming_port (context) " on page 88.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	12965
Byte order:	Network
When to Set:	Can only be set during object initialization.

resolver_multicast_port (context)

Multicast port used for Topic Resolution. This option automatically sets the values for " resolver_multicast_incoming_port (context) " on page 88 and " resolver_multicast_outgoing_port (context) " on page 89 as evidenced by the default values for all three options, which are the same.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	12965
Byte order:	Network
When to Set:	Can only be set during object initialization.

resolver_multicast_receiver_socket_buffer (context)

Value used to set **SO_RCVBUF** value of the resolver receivers. In some cases the OS will not allow all of this value to be used. A value of 0 instructs UM to use the default OS values. See the section on "Socket Buffer Sizes" on page 52 for platform-dependent information. See also our white paper Topics in High Performance Messaging for background and guidelines on UDP buffer sizing.

Scope:	context
Type:	**lbm_ulong_t**
Units:	bytes

Default value:	0 (use OS default)
When to Set:	Can only be set during object initialization.

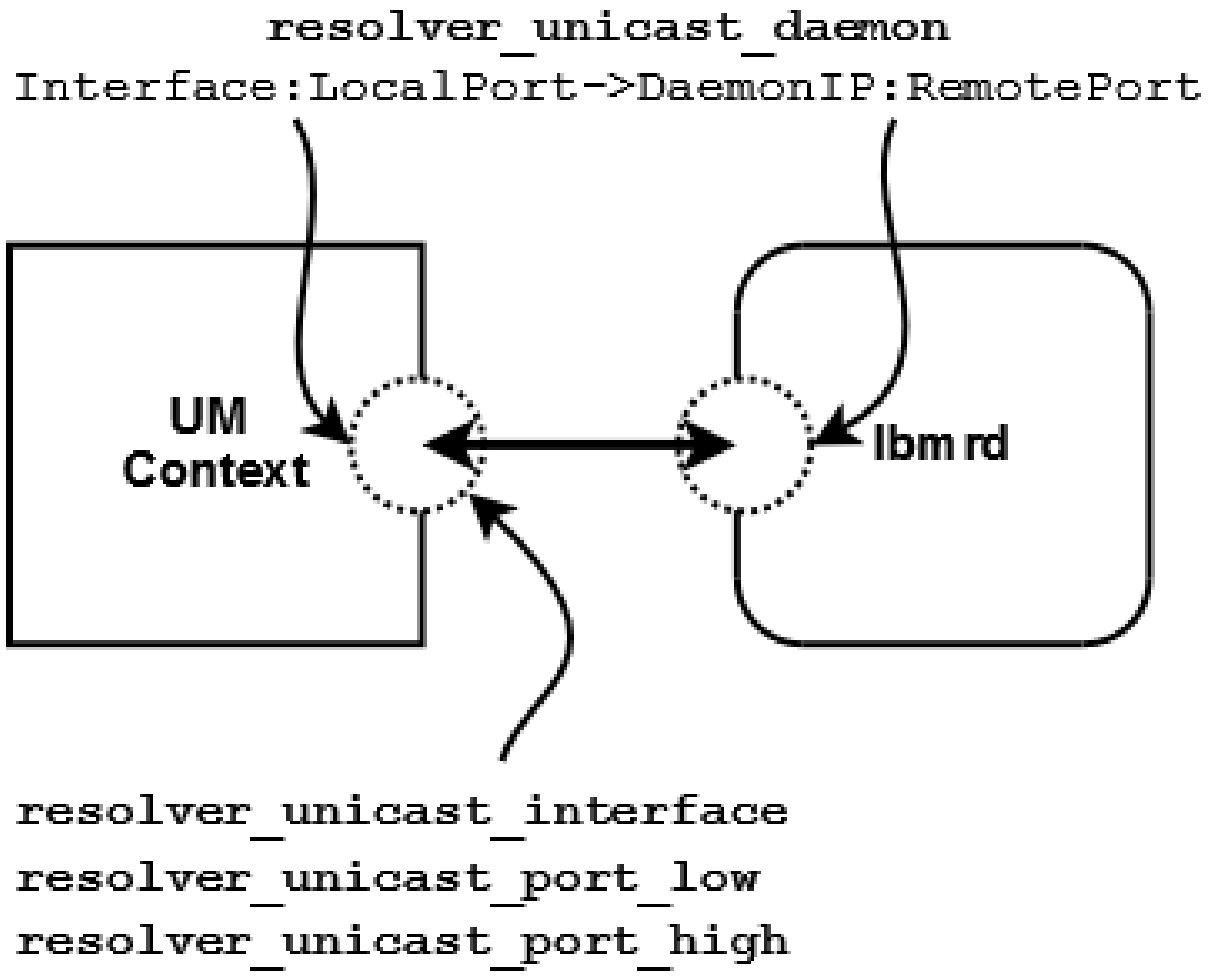
resolver_multicast_ttl (context)

The IP TTL (hop count) to use for a Topic Resolution packet. A value of 1 confines the packet to the local network (but may also cause high CPU usage on some routers). Also controls TTL on LBT-RM packets.

Scope:	context
Type:	**lbm_uint8_t**
Default value:	16
When to Set:	May be set during operation.

Unicast Resolver Network Options

Unicast resolver network options



This diagram shows a single unicast resolver daemon configured with resolver_unicast_daemon.

If using multiple `lbmrd` instances with a single context, you can configure resolver_unicast_interface and resolver_unicast_port_low/high and omit the `Interface:LocalPort` section of resolver_unicast_daemon.

See also *Unicast Topic Resolution* in the *Ultra Messaging Concepts Guide* for more information.

resolver_unicast_daemon (context)

Add a unicast resolver daemon specification to the list of unicast resolver daemons. Unlike most other UM settings, every time this setting is called, it adds another daemon specification to the list and does NOT overwrite previous specifications. Each entry contains the interface, source port, resolver IP, and destination port for a single daemon. For the configuration file as well as string versions of setting this option, the string value is formatted as follows:

```
[Iface[:Src_Port]->]IP:Dest_Port
```

Iface

the interface to use (previously set via “ resolver_unicast_interface (context) ” on page 92)

`Src_Port`

the source port to use (previously “ resolver_unicast_port (context) ” on page 227)

`IP`

the resolver daemon's IP address (previously “ resolver_unicast_address (context) ” on page 226)

`Dest_Port`

the resolver daemon's UDP port (previously “ resolver_unicast_destination_port (context) ” on page 226)

You can omit either the `Src_Port` or both the `Iface` and `Src_Port`, in which case the default “ resolver_unicast_interface (context) ” on page 92 and “ resolver_unicast_port (context) ” on page 227 settings are used. Because each entry adds a new daemon specification and does not overwrite previous values, an entry or string with the IP address of 0.0.0.0 and port of 0 removes all previous daemon specifications. At least one daemon specification means the context does not use multicast topic resolution.

Possible formats of this option are as follows:

```
Interface:LocalPort->DaemonIP:RemotePort
Interface->DaemonIp:RemotePort
DaemonIP:RemotePort
```

You can specify `Interface` in any of the ways described in “Specifying Interfaces” on page 52.

Scope:	context
Type:	**lbm_ucast_resolver_entry_t**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMS 5.0

resolver_unicast_interface (context)

Specifies the network interface over which UM receives unicast Topic Resolution messages. Can specify full IP address of interface, or just network part (see “Specifying Interfaces” on page 52 for details). Default is set to INADDR_ANY, meaning that it will accept unicast Topic Resolution messages on any interface.

Scope:	context
Type:	**lbm_ipv4_address_mask_t**
Default value:	0.0.0.0 (**INADDR_ANY**)
When to Set:	Can only be set during object initialization.

resolver_unicast_port_high (context)

The highest local UDP port in a range of ports used for unicast topic resolution messages. The UM resolution daemon (lbmrd) sends unicast topic resolution messages to the UDP port range defined by this option and " resolver_unicast_port_low (context) " on page 93.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	14406
Byte order:	Host
When to Set:	Can only be set during object initialization.

resolver_unicast_port_low (context)

The lowest local UDP port in a range of ports used for unicast topic resolution messages. The UM resolution daemon (lbmrd) sends unicast topic resolution messages to the UDP port range defined by this option and " resolver_unicast_port_high (context) " on page 93.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	14402
Byte order:	Host
When to Set:	Can only be set during object initialization.

resolver_unicast_receiver_socket_buffer (context)

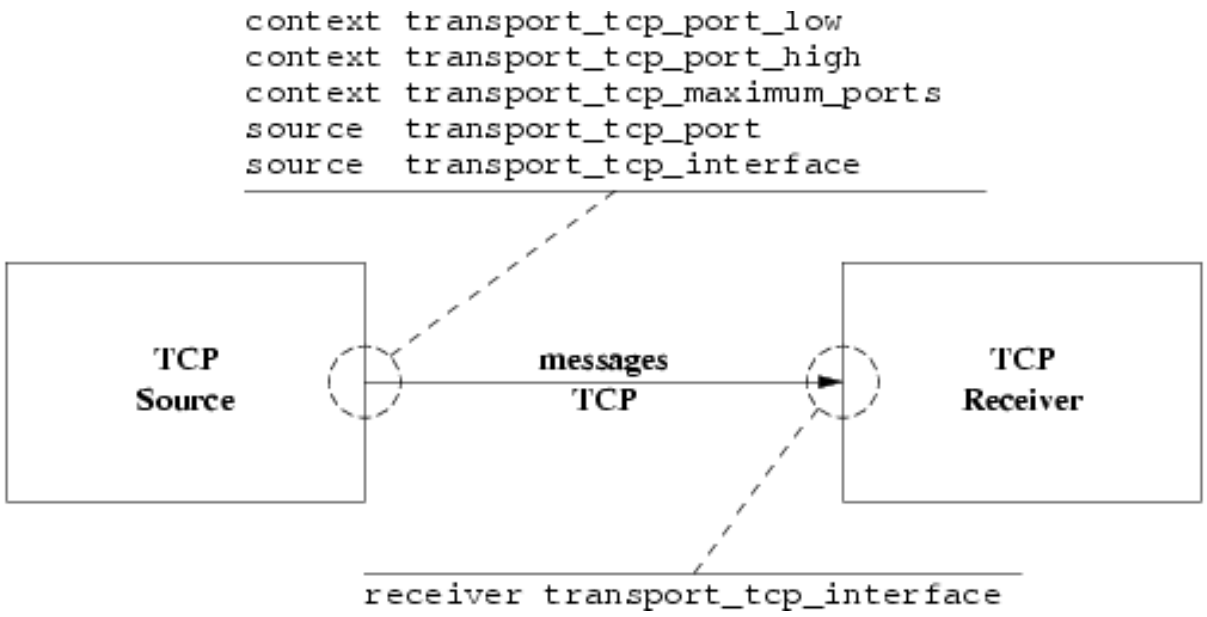
Value used to set **SO_RCVBUF** value of the UDP receivers for unicast topic resolution messages. In some cases the OS will not allow all of this value to be used. A value of 0 instructs UM to use the default OS values. See the section on "Socket Buffer Sizes" on page 52 for platform-dependent information.

Scope:	context
Type:	**lbm_ulong_t**
Units:	bytes
Default value:	0 (use OS default)
When to Set:	Can only be set during object initialization.

Transport TCP Network Options

TCP receivers initiate connections toward TCP sources. Messages flow from sources to receivers.

TCP network options



" transport_tcp_port_low (context) " on page 96
is the lowest port that UMS will allocate for TCP sources in a context; " transport_tcp_port_high (context) " on page 96
is the highest. No more than " transport_tcp_maximum_ports (context) " on page 95
ports will be assigned to TCP sources within a single context.

Creation of a UMS source on a TCP transport will allocate an unused port from the range if less than " transport_tcp_maximum_ports (context) " on page 95 ports have already been allocated. Setting " transport_tcp_maximum_ports (context) " on page 95 to a fraction of the range allows the corresponding multiple number of UMS processes to share a common configuration.

If a particular TCP port is desired by a source, it may be given with " transport_tcp_port (source) " on page 95. If the desired port is already in use, then an unused port will be sought as described above. A value of `0` (the default) expresses no preference and results in the default open port seeking behavior described above.

" transport_tcp_interface (source) " on page 95 may be used on TCP sources to choose particular interface, overriding the default `INADDR_ANY` which accepts connections on all interfaces. Similarly, " transport_tcp_interface (receiver) " on page 94 may be used on receivers to choose a particular interface for outgoing connections.

transport_tcp_interface (receiver)

Specifies the network interface to which UM receivers bind before connecting to sources. You can specify the full IP address of interface, or just the network part (see "Specifying Interfaces" on page 52 for details).

Scope:	receiver
Type:	**lbm_ipv4_address_mask_t**

Default value:	0.0.0.0 (**INADDR_ANY**)
When to Set:	Can only be set during object initialization.

transport_tcp_interface (source)

Specifies the network interface over which UM accepts connection requests (from topic receivers). You can specify the full IP address of interface, or just the network part (see "Specifying Interfaces" on page 52 for details). Be aware that this option is applied to the transport session when the first topic is created on that session. Thus, setting a different interface for a subsequent topic that maps onto the same transport session will have no effect. Default is set to INADDR_ANY, meaning that it will not bind to a specific interface. You can also modify the default by setting the option to 0.0.0.0/0 which produces the same result.

Scope:	source
Type:	**lbm_ipv4_address_mask_t**
Default value:	0.0.0.0 (**INADDR_ANY**)
When to Set:	Can only be set during object initialization.

transport_tcp_maximum_ports (context)

Maximum number of TCP sessions to allocate.

Scope:	context
Type:	**lbm_uint16_t**
Units:	number of ports
Default value:	10
When to Set:	Can only be set during object initialization.

transport_tcp_port (source)

The preferred TCP port number for this Topic. If 0, the context will attempt to find one in the given TCP port range.

Scope:	source
Type:	**lbm_uint16_t**
Default value:	0 (pick open port)
Byte order:	Network
When to Set:	Can only be set during object initialization.

transport_tcp_port_high (context)

High port number to assign to TCP sessions.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	14390
Byte order:	Host
When to Set:	Can only be set during object initialization.

transport_tcp_port_low (context)

Low port number to assign to TCP sessions.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	14371
Byte order:	Host
When to Set:	Can only be set during object initialization.

Transport TCP Operation Options

transport_session_maximum_buffer (source)

Value used to control the maximum amount of data buffered in UM for the transport session used for the topic. For the normal multiple receiver behavior, this value represents the total buffered by all TCP receivers. For the bounded_latency and source_paced multiple receiver behavior, this value represents the individual receiver buffered amount. This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_ulong_t**
Units:	bytes

Default value:	65536
When to Set:	Can only be set during object initialization.

transport_tcp_activity_method (receiver)

For TCP sessions only. The type of timeout method to use for TCP receivers.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.3.8/UME 2.0.6.

String value	Integer value	Description
timer	**LBM_RCV_TOPIC_ATTR_TCP_ACTIVITY_TIMEOUT_TIMER**	Timer method that requires new TCP session data to be sent to determine if the connection is alive. **Default for all.**
SO_KEEPALIVE	**LBM_RCV_TOPIC_ATTR_TCP_ACTIVITY_TIMEOUT_SO_KEEPALIVE**	Set SO_KEEPALIVE on the TCP connection which uses the TCP keepalive support in the operating system to determine if the connection is alive. When you use the SO_KEEPALIVE method, UM uses transport_tcp_activity_timeout value to set the idle and probe times for SO_KEEPALIVE. The idle time is 90% of the timeout value at most. The probe time is 10% with 10 seconds as the minimum.

transport_tcp_activity_timeout (receiver)

For TCP sessions only. The maximum time that a TCP session may be quiescent before it is deleted and an EOS event is delivered for all topics using this transport session. A value greater than zero turns the timer on.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	0
When to Set:	Can only be set during object initialization.

transport_tcp_activity_timeout (source)

For TCP sessions only. This timeout option enables using `SO_KEEPALIVE` to detect when a receiver does not cleanly disconnect or is no longer reachable from the source. When the timeout expires, a DISCONNECT source event is delivered. This option affects only Linux or Windows platforms. Outstanding TCP retransmit timers must expire before this timer starts. With a default Linux or Windows system configuration, TCP retransmit timers may take minutes or even hours to expire. Therefore the total time taken to detect an unreachable receiver may be significantly higher than the value configured for this timeout. A value of zero (0) defers TCP timeout to OS settings.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	0
When to Set:	Can only be set during object initialization.

transport_tcp_coalesce_threshold (source)

UM passes implicitly batched messages to the Operating System **sendmsg()** as a set unless the size of the set exceeds the coalescing threshold at which point the messages are coalesced and passed to the O/S as one copy.

This option accommodates the different number of iovecs supported by different O/Ss. Tuning this option balances the efficiency of less iovecs handled by the OS vs. the expense of an additional copy operation of the messages before sending. The default values are also the maximum allowable values.

Scope:	source
Type:	**int**
Units:	number of individual messages
Default value:	1024 for Linux, Microsoft™ Windows™ , Darwin; 16 for Solaris, AIX, HPUX
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.6/UME 2.3.

transport_tcp_datagram_max_size (context)

The maximum datagram size that can be generated for a TCP transport session. The default value is 65535, the minimum is 500 bytes, and the maximum is 65535.

Scope:	context
Type:	**lbm_uint_t**
Units:	bytes

Default value:	65535
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1/UME 3.1/UMQ 1.1

transport_tcp_exclusiveaddr (source)

Applicable only to Windows. Indicate whether the TCP session should set **SO_EXCLUSIVEADDRUSE** or not before it binds. The default setting in Windows allows multiple binds to the same port. By default, UM will set **SO_EXCLUSIVEADDRUSE** to minimize port sharing. Refer to Microsoft's web site for more information on **SO_EXCLUSIVEADDRUSE**.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	Set **SO_EXCLUSIVEADDRUSE**. **Default for Windows.**
0	Do not set **SO_EXCLUSIVEADDRUSE**.

transport_tcp_listen_backlog (source)

The backlog used in the TCP **listen()** call to set the queue length for incoming connections.

Scope:	source
Type:	**int**
Units:	number of queued connections
Default value:	5
When to Set:	Can only be set during object initialization.

transport_tcp_multiple_receiver_behavior (source)

The flow control behavior desired when multiple TCP clients are receiving for a TCP session. If an application is only allowed to send as fast as all receivers can consume data, markedly slower receivers will lose data (have unrecoverably lost UM messages) if they can not keep up with the other faster receivers for the TCP session. Note that at high rates and with receivers that can consume data at fairly similar rates, all receivers may experience some loss at times. This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and

ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
normal	**LBM_SRC_TOPIC_ATTR_TCP_MULTI_RECV_NORMAL**	The application sends as fast as the slowest receiver consumes data. This slows down all receivers on that TCP session. **Default for all.**
bounded_latency	**LBM_SRC_TOPIC_ATTR_TCP_MULTI_RECV_BOUNDED_LATENCY**	The application sends as fast as the fastest receiver can consume data even if recent data headed for slower receivers must be discarded.
source_paced	**LBM_SRC_TOPIC_ATTR_TCP_MULTI_RECV_SOURCE_PACED**	The application sends as fast as it can even if recent data headed for any or all receivers must be discarded.

transport_tcp_multiple_receiver_send_order (source)

In the case of multiple receivers, this option determines whether datagrams are sent to each receiver in the established order of receivers, or if receivers are selected in random order for each datagram transmission.

Scope:	source
Type:	**lbm_src_topic_attr_t**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
serial	**LBM_SRC_TOPIC_ATTR_TCP_MULTI_RECV_SEND_ORDER_SERIAL**	Select receivers to receive a datagram based on current established order. **Default for all.**
random	**LBM_SRC_TOPIC_ATTR_TCP_MULTI_RECV_SEND_ORDER_RANDOM**	For each datagram sent, select receivers in random order, for the sake of "fairness". Note that this option adds a small amount of CPU overhead.

transport_tcp_nodelay (source)

Whether the TCP sockets used for the transport session should set **TCP_NODELAY** or not. (Setting TCP_NODELAY disables Nagle's algorithm.)

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	TCP transport sockets should set **TCP_NODELAY** (disable Nagle). **Default for all.**
0	TCP transport sockets should not set **TCP_NODELAY** (leave Nagle enabled).

transport_tcp_receiver_socket_buffer (context)

Value used to set **SO_RCVBUF** value of the TCP receivers for topics. In some cases the OS will not allow all of this value to be used. A value of 0 instructs UM to use the default OS values. See the section on "Socket Buffer Sizes" on page 52 for platform-dependent information.

Scope:	context
Type:	**lbm_ulong_t**
Units:	bytes
Default value:	0 (use OS default)
When to Set:	Can only be set during object initialization.

transport_tcp_reuseaddr (source)

Whether the TCP session should set **SO_REUSEADDR** or not before it binds. WARNING: This option is not recommended for Microsoft™ Windows™ users because the **SO_REUSEADDR** socket option in Windows

allows a socket to forcibly bind to a port in use by another socket. Multiple sockets using the same port results in indeterminate behavior.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	Set **SO_REUSEADDR**.
0	Do not set **SO_REUSEADDR**. **Default for all.**

transport_tcp_sender_socket_buffer (source)

Value used to set the **SO_SNDBUF** value of the TCP session. In some cases the OS will not allow all of this value to be used. A value of 0 instructs UM to use the OS defaults. See the section on "Socket Buffer Sizes" on page 52 for platform-dependent information.

Scope:	source
Type:	**lbm_ulong_t**
Units:	bytes
Default value:	0 (use OS default)
When to Set:	Can only be set during object initialization.

transport_tcp_use_session_id (source)

Enable a session ID for the TCP Transport.

Indicates if the application desires the UM TCP Transport to use a Session ID. Older versions of UM may not understand session IDs with TCP and may not be able to receive TCP transport sessions that include session IDs.

Scope:	source
Type:	**int**

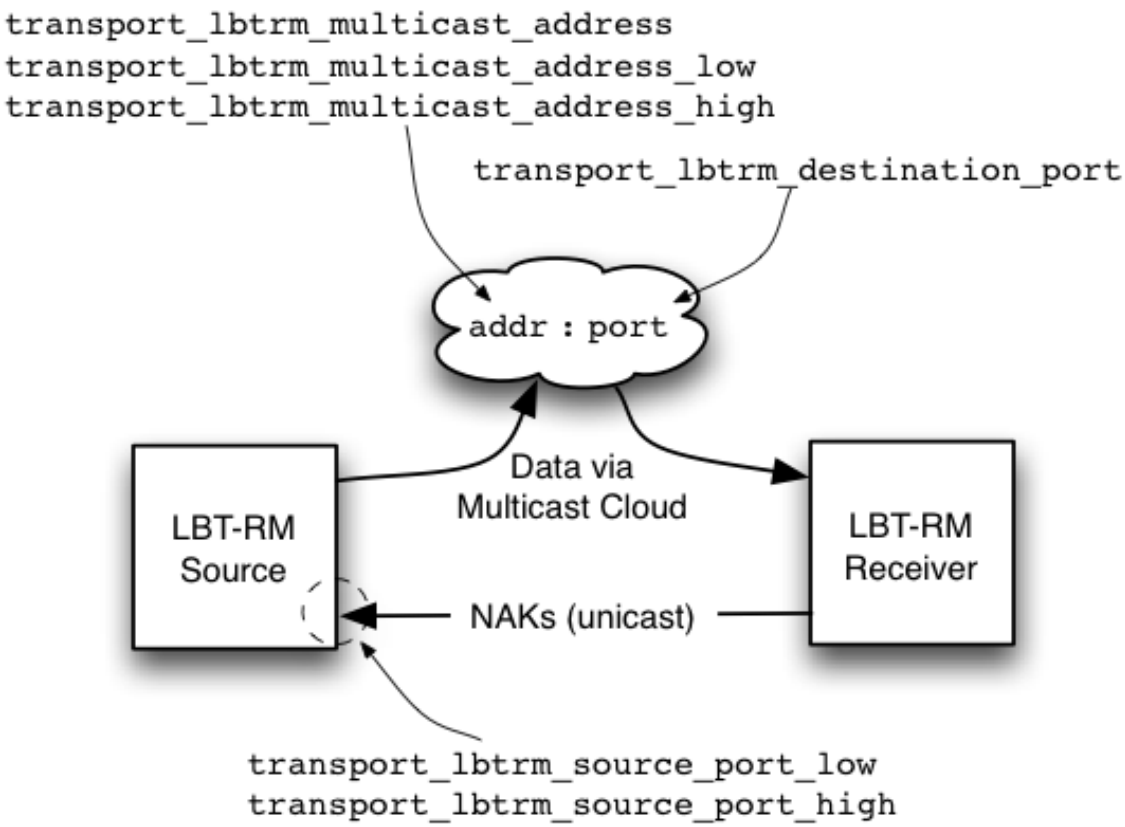
When to Set:	Can only be set during object initialization.
Version	This option was implemented in UM 6.0.

Value	Description
1	Indicates the application desires TCP to use a session ID. **Default for all.**
0	Indicates the application does not desire TCP to use a session ID.

Transport LBT-RM Network Options

The following illustration shows where the various options are applied. Note that for a multi-homed LBT-RM source, the interface LBT-RM multicast resolver interface specified with " resolver_multicast_interface (context) " on page 88 will be used as the source for LBT-RM.

LBT-RM network options



transport_lbtrm_destination_port (source)

The UDP destination port used for this Topic when the transport is LBT-RM.

Scope:	source
Type:	**lbm_uint16_t**
Default value:	14400
Byte order:	Network
When to Set:	Can only be set during object initialization.

transport_lbtrm_multicast_address (source)

The preferred multicast address for this Topic when the transport is LBT-RM. If 0.0.0.0 (**INADDR_ANY**), the context will attempt to find one in the given multicast address range.

Scope:	source
Type:	**struct in_addr**
Default value:	0.0.0.0 (**INADDR_ANY**)
When to Set:	Can only be set during object initialization.

transport_lbtrm_multicast_address_high (context)

Multicast address used as the highest value to assign to LBT-RM sessions.

Scope:	context
Type:	**struct in_addr**
Default value:	224.10.10.14
When to Set:	Can only be set during object initialization.

transport_lbtrm_multicast_address_low (context)

Multicast address used as the lowest value to assign to LBT-RM sessions.

Scope:	context
Type:	**struct in_addr**
Default value:	224.10.10.10
When to Set:	Can only be set during object initialization.

transport_lbtrm_source_port_high (context)

Highest port number value used for LBT-RM source sessions' unicast NAK processing. UM sends NAKs to this port number for processing and retransmission generation. Each LBT-RM session must use a unique port value. Note that this does not control the UDP source port on the outbound LBT-RM stream.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	14399
Byte order:	Host
When to Set:	Can only be set during object initialization.

transport_lbtrm_source_port_low (context)

Lowest port number value used for LBT-RM source sessions' unicast NAK processing. UM sends NAKs to this port number for processing and retransmission generation. Each LBT-RM session must use a unique port value. Note that this does not control the UDP source port on the outbound LBT-RM stream.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	14390
Byte order:	Host
When to Set:	Can only be set during object initialization.

Transport LBT-RM Reliability Options

In addition to LBT-RM reliability options, this section discusses the following topics.

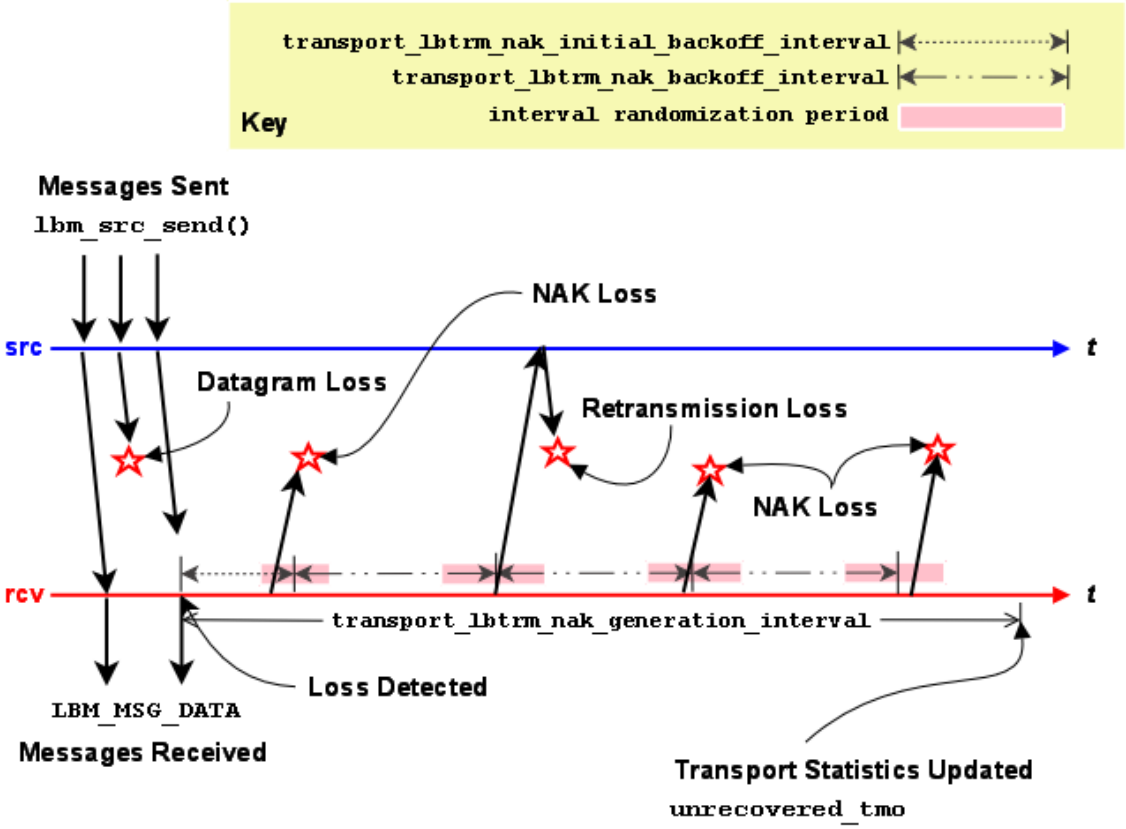
- "LBT-RM Datagram Loss Resulting in Unrecovered Message Loss" on page 105
- "LBT-RM Source Ignoring NAKs for Efficiency" on page 106
- "LBT-RM Receiver Suppressing NAK Generation" on page 107

LBT-RM Datagram Loss Resulting in Unrecovered Message Loss

The key options that control the effort that an LBT-RM receiver will make to recover from datagram loss are " transport_lbtrm_nak_backoff_interval (receiver) " on page 108 and " transport_lbtrm_nak_generation_interval (receiver) " on page 109. Timers for both start when loss is detected. The following timeline illustrates a case where a receiver is notified of unrecoverable message loss following repeated datagram loss.

Note: The actual length of the interval randomization periods are between 1/2 and 3/2 of the configured interval value. In the diagram below (Scenario Timeline: LBT-RM Datagram Loss Resulting in Unrecovered Message Losson page 106), these periods appear shorter to simplify the diagram.

Scenario Timeline: LBT-RM Datagram Loss Resulting in Unrecovered Message Loss



Set " transport_lbtrm_nak_backoff_interval (receiver) " on page 108 to the NAK service time that could be reasonably expected from the receiver's location in the network plus some cushion for network congestion. Set " transport_lbtrm_nak_generation_interval (receiver) " on page 109 to the latency budget established for the transport layer. See our whitepaper Topics in High Performance Messaging for background on latency budgets. See also Reducing Loss Recovery Latencies

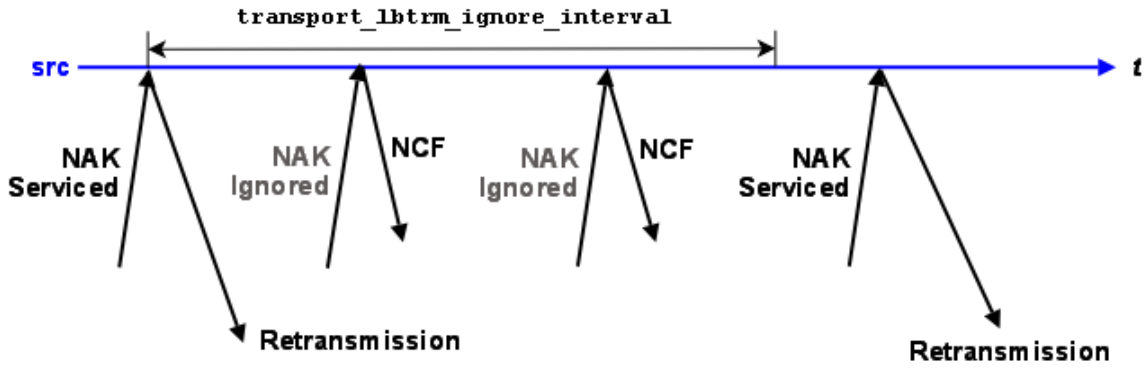
Note: The diagram, Scenario Timeline: LBT-RM Datagram Loss Resulting in Unrecovered Message Losson page 106, depicts loss occurring over a LBT-RM transport session. Many topics may be sent across a given transport session. For information about how topic level loss is reported, see "Delivery Control Options" on page 172.

LBT-RM Source Ignoring NAKs for Efficiency

Bandwidth efficiency of an LBT-RM source may be improved by avoiding useless retransmissions. Consider the case of an LBT-RM source that has received a NAK for a datagram that it has just retransmitted. It's likely that the NAK and the retransmission "crossed in the mail." Hence it's likely that the receiver generating the NAK will have already received the retransmission just sent. If so, there's no need for the source to send

another retransmission so the NAK can be safely ignored. Consider the timeline illustrated in Scenario Timeline: LBT-RM Source Ignoring NAKs for Efficiencyon page 107.

Scenario Timeline: LBT-RM Source Ignoring NAKs for Efficiency



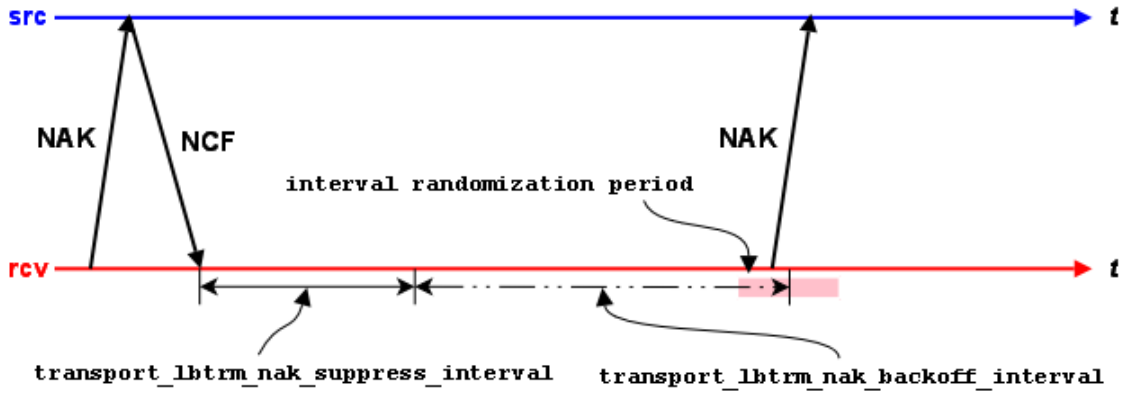
This shows NAKs for a given datagram being ignored for " transport_lbtrm_ignore_interval (source) " on page 108 following the retransmission of that datagram. (The successive NAKs received by the source in Scenario Timeline: LBT-RM Source Ignoring NAKs for Efficiencyon page 107 indicate that more than one receiver is subscribed to the source's topic.) NAKs will be serviced as normal following the passage of the interval. When ignoring a NAK, the source sends a NCF (NAK ConFirmation) instead of a retransmission, which starts a NAK suppression interval at the receiver. (See Scenario Timeline: An LBT-RM Receiver Suppressing NAK Generationon page 108.)

LBT-RM Receiver Suppressing NAK Generation

LBT-RM sources want receivers to be notified that their NAKs have been heard. Prompt notification via a retransmission or NCF can suppress useless NAK generation. There are a variety of circumstances where the source does not send a retransmission in response to a receiver's NAK. For example, as shown in Scenario Timeline: LBT-RM Source Ignoring NAKs for Efficiencyon page 107, NAKs received during the ignore interval do not generate retransmissions. Another example would be if previous retransmissions have

used up all the retransmission bandwidth for the current rate limiter interval. See Scenario Timeline: An LBT-RM Receiver Suppressing NAK Generationon page 108 for a depiction of how a receiver responds to a NCF.

Scenario Timeline: An LBT-RM Receiver Suppressing NAK Generation



Following the receipt of an NCF, a receiver suppresses all NAK generation until " transport_lbtrm_nak_suppress_interval (receiver) " on page 110 passes. NAK generation resumes with the usual " transport_lbtrm_nak_backoff_interval (receiver) " on page 108 if repair was not received during the suppression interval.

Note: The actual length of the interval randomization period is between 1/2 and 3/2 of the configured interval value. In Scenario Timeline: An LBT-RM Receiver Suppressing NAK Generationon page 108, this period appears shorter to simplify the diagram.

transport_lbtrm_ignore_interval (source)

The interval to ignore NAKs after a retransmission is sent. This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	500 (0.5 seconds)
When to Set:	Can only be set during object initialization.

transport_lbtrm_nak_backoff_interval (receiver)

For LBT-RM sessions only. The maximum interval between transmissions of a single NAK. The actual time the receiver will wait to NAK again is random. The algorithm used to determine the time range is (1/2 * backoff_interval - 3/2 * backoff_interval). This can result in a wait interval longer than the specified value. This option affects the transport session underlying the receiver rather than the receiver itself. The transport session uses the value from the first receiver created on the session and ignores subsequent receivers.

Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* and "Interrelated Configuration Options" on page 42 for additional information.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	200 (0.2 seconds)
When to Set:	Can only be set during object initialization.

transport_lbtrm_nak_generation_interval (receiver)

For LBT-RM sessions only. The maximum time that a piece of data may be outstanding before the data is unrecoverably lost. Although the minimum valid value is 5 milliseconds, larger values are advisable. This option affects the transport session underlying the receiver rather than the receiver itself. The transport session uses the value from the first receiver created on the session and ignores subsequent receivers. Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* and "Interrelated Configuration Options" on page 42 for additional information.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10000 (10 seconds)
When to Set:	Can only be set during object initialization.

transport_lbtrm_nak_initial_backoff_interval (receiver)

For LBT-RM sessions only. The interval between loss detection and transmission of the first NAK. The actual time the receiver will wait to NAK is random. The algorithm used to determine the time range is (1/2 * initial_backoff_interval - 3/2 * initial_backoff_interval). This can result in a wait interval longer than the specified value. A value of 0 indicates that the receiver should immediately send a NAK. Users should be fully aware of the implications of this before using a value of 0.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	50 (0.05 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.4/UME 2.1.

transport_lbtrm_nak_suppress_interval (receiver)

For LBT-RM sessions only. The maximum interval to suppress sending NAKs after an NCF or a NAK from another receiver. This option affects the transport session underlying the receiver rather than the receiver itself. The transport session uses the value from the first receiver created on the session and ignores subsequent receivers. Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	1000 (1 second)
When to Set:	Can only be set during object initialization.

transport_lbtrm_receiver_socket_buffer (context)

Value used to set **SO_RCVBUF** value of the LBT-RM receiver multicast socket. In some cases the OS will not allow all of this value to be used. See the section on "Socket Buffer Sizes" on page 52 for platform-dependent information. See also our white paper Topics in High Performance Messaging for background and guidelines on UDP buffer sizing.

Scope:	context
Type:	**lbm_ulong_t**
Units:	bytes
Default value:	524288 (512KB)
When to Set:	Can only be set during object initialization.

transport_lbtrm_send_naks (receiver)

For LBT-RM sessions only. This flag indicates whether LBT-RM should send negative acknowledgements (NAKs) for missing packets or not. This option affects the transport session underlying the receiver rather than the receiver itself. The transport session uses the value from the first receiver created on the session

and ignores subsequent receivers. Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	NAKs are sent for missing packets to request retransmission. **Default for all.**
0	Do not send NAKs for missing packets.

transport_lbtrm_source_socket_buffer (context)

Value used to set **SO_SNDBUF** value of the LBT-RM send multicast socket. In some cases the OS will not allow all of this value to be used. See the section on "Socket Buffer Sizes" on page 52 for platform-dependent information. A value of 0 instructs UM to use the OS default.

Scope:	context
Type:	**lbm_ulong_t**
Units:	bytes
Default value:	0 (use OS default or 131072, whichever is larger)
When to Set:	Can only be set during object initialization.

transport_lbtrm_transmission_window_limit (source)

Caps the total amount of memory that a transmission window uses, which includes data and overhead. For example, if the " transport_lbtrm_transmission_window_size (source) " on page 112 (source) is 24 MB (default) and the source sends (with flush flag set) 1.2 million messages with a 20-byte payload and 230-byte header, the actual amount of memory used can approximate 300 MB. The default value of 0 (zero) disables the transmission window size limit.

Scope:	source
Type:	**size_t**
Units:	bytes
Default value:	0 (zero)
When to Set:	Can only be set during object initialization.

transport_lbtrm_transmission_window_size (source)

The maximum amount of buffered payload data, excluding UM headers, that the LBT-RM source is allowed to retain for retransmissions. The minimum valid value is 65,536 bytes. This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Fore more information, see the *Ultra Messaging Concepts Guide*, *Source Configuration and Transport Sessions*.

Scope:	source
Type:	**size_t**
Units:	bytes
Default value:	25165824 (24 MB)
When to Set:	Can only be set during object initialization.

Transport LBT-RM Operation Options

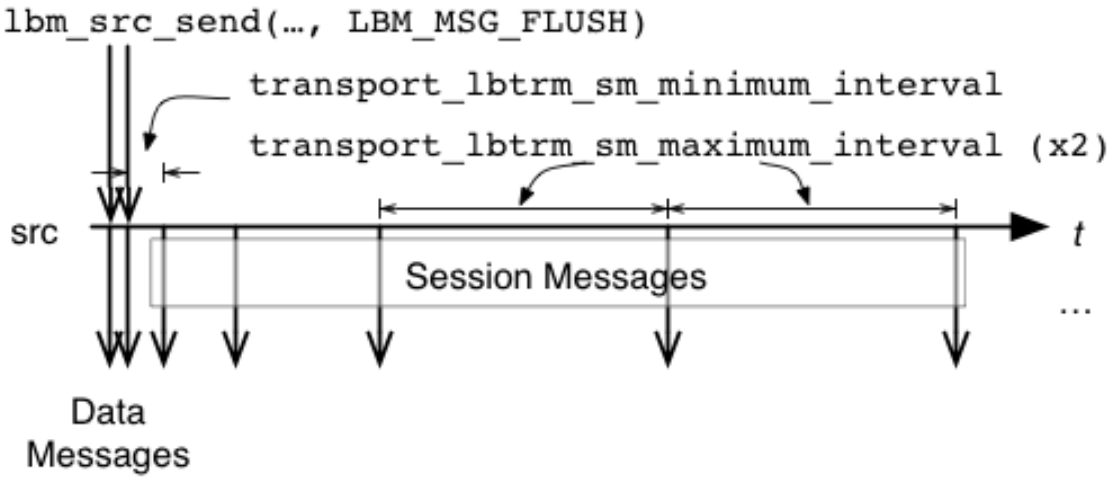
Scenario Timeline: LBT-RM Source Stops Sending

Reliable multicast protocols like LBT-RM rely on sequence numbers and the arrival of data after a loss as evidence that the loss happened. What would happen if the last packet sent by a source was lost? How would receivers learn of the loss if no further messages were sent?

LBT-RM generates session messages when the sources on a transport session stop sending. These messages contain the expected last sequence number for the session so that receivers can detect loss even when sources aren't sending. Session messages also help to maintain state in multicast routers and switches that require regular traffic to prevent the reclamation of unused forwarding entries.

The following timeline illustrates the case where an LBT-RM source stops sending.

An LBT-RM source stops sending

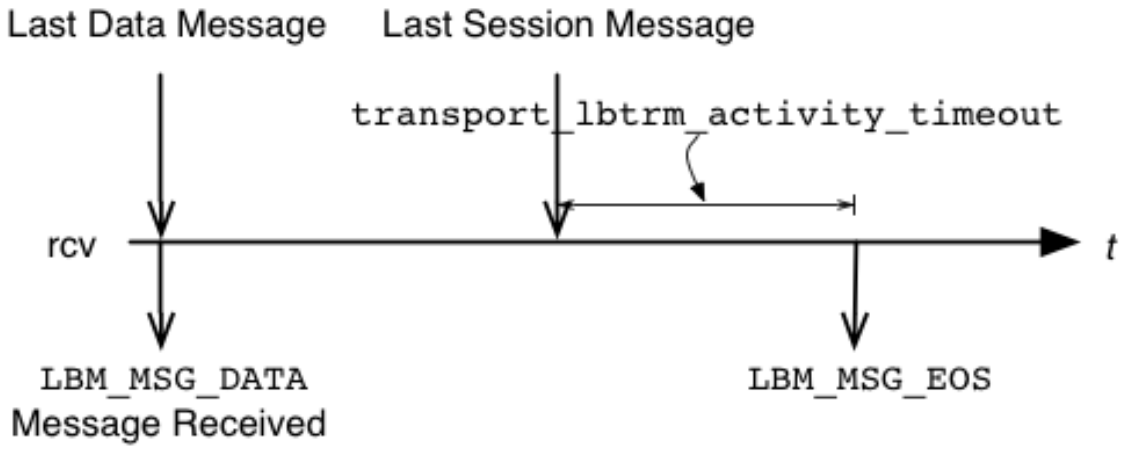


No session messages are generated as long as the interval between **lbm_src_send()** calls that generate writes to LBT-RM is less than “ transport_lbtrm_sm_minimum_interval (source) ” on page 117. The interval between session messages starts at “ transport_lbtrm_sm_minimum_interval (source) ” on page 117 and doubles till it reaches “ transport_lbtrm_sm_maximum_interval (source) ” on page 116.

Scenario Timeline: Receiver Detects End of LBT-RM Session

The absence of activity on a transport session is the only indication receivers get that a source is gone or no longer available through any network path. LBT-RM receivers reset a session activity timer for each data message or session message that arrives. If the activity timer ever expires, all receivers on the transport session receive an **LBM_MSG_EOS** event. This is illustrated in the following timeline:

A receiver detects the end of an LBT-RM session



The activity timer is controlled with the “ transport_lbtrm_activity_timeout (receiver) ” on page 113 option.

transport_lbtrm_activity_timeout (receiver)

For LBT-RM sessions only. The maximum time that an LBT-RM session may be quiescent before it is deleted and an EOS event is delivered for all topics using this transport session. This option affects the transport session underlying the receiver rather than the receiver itself. The transport session uses the value from the first receiver created on the session and ignores subsequent receivers. Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	60000 (60 seconds)
When to Set:	Can only be set during object initialization.

transport_lbtrm_coalesce_threshold (source)

UM passes implicitly batched messages to the Operating System **sendmsg()** as a set unless the size of the set exceeds the coalescing threshold at which point the messages are coalesced and passed to the O/S as one copy.

This option accommodates the different number of iovecs supported by different O/Ss. Tuning this option balances the efficiency of less iovecs handled by the OS vs. the expense of an additional copy operation of the messages before sending. The default value is also the maximum allowable value for Solaris, AIX and HPUX. For Linux and Microsoft® Windows® and Darwin, the maximum allowable value is 1023. These maximum allowable values are one less than what the O/S provides.

This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**int**
Units:	number of individual messages
Default value:	15
When to Set:	Can only be set during object initialization.

transport_lbtrm_data_rate_limit (context)

Maximum aggregate transmission rate of all LBT-RM sessions' original data plus retransmissions for this particular context.

Scope:	context
Type:	**unsigned long int**
Units:	bits per second
Default value:	10000000 (10 Mbps)
When to Set:	Can only be set during object initialization.

transport_lbtrm_datagram_max_size (context)

The maximum datagram size that can be generated for a LBT-RM transport session. The default value is 8192, the minimum is 500 bytes, and the maximum is 65535.

Scope:	context
Type:	**lbm_uint_t**
Units:	bytes

Default value:	8192
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1/UME 3.1/UMQ 1.1

transport_lbtrm_preactivity_timeout (receiver)

Use this option only if the receiver subscribes to a pre-LBM 3.3 source or if you have turned off " transport_topic_sequence_number_info_interval (source) " on page 67 messages in your post-LBM 3.3 implementation. Set it high enough so the source starts sending data messages before the timeout expires. This timeout begins when the receiver receives a topic advertisement. Pre-LBM 3.3 sources do not send TSNI messages which in effect inform receivers that the source is alive even though it has not started sending data. Session messages provide the same information but do not begin until after the source has started sending data. This option provides an additional " transport_lbtrm_activity_timeout (receiver) " on page 113 for the receiver that does not rely on TSNI or sessions messages. The default value of 0 (zero) essentially disables this option, giving precedence to the receiver's standard " transport_lbtrm_activity_timeout (receiver) " on page 113. This option affects the transport session underlying the receiver rather than the receiver itself. The transport session uses the value from the first receiver created on the session and ignores subsequent receivers. Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	0 (zero)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.4.1/UME 2.1.1.

transport_lbtrm_rate_interval (context)

Period that LBT-RM rate limiter runs. Reducing period reduces burst intensity, but also increases CPU load.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds

Default value:	100
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
"10" (Integer value as a string.)	**10**	LBT-RM rate limiter runs every 10 milliseconds.
"20" (Integer value as a string.)	**20**	LBT-RM rate limiter runs every 20 milliseconds.
"50" (Integer value as a string.)	**50**	LBT-RM rate limiter runs every 50 milliseconds.
"100" (Integer value as a string.)	**100**	LBT-RM rate limiter runs every 100 milliseconds. **Default for all.**

transport_lbtrm_retransmit_rate_limit (context)

Maximum aggregate transmission rate of all LBT-RM sessions' retransmissions for this particular context. This should always be less than the value used for original data.

Scope:	context
Type:	**unsigned long int**
Units:	bits per second
Default value:	5000000 (5 Mbps)
When to Set:	Can only be set during object initialization.

transport_lbtrm_sm_maximum_interval (source)

The maximum interval between LBT-RM session messages. In lieu of data being sent, LBT-RM sends session messages to inform receivers of sequence numbers and to let receivers know that the sender is still transmitting. This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10000 (10 seconds)
When to Set:	Can only be set during object initialization.

transport_lbtrm_sm_minimum_interval (source)

The minimum interval between LBT-RM session messages. In lieu of data being sent, LBT-RM sends session messages to inform receivers of sequence numbers and to let receivers know that the sender is still transmitting. This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	200 (0.2 seconds)
When to Set:	Can only be set during object initialization.

transport_lbtrm_tgsz (source)

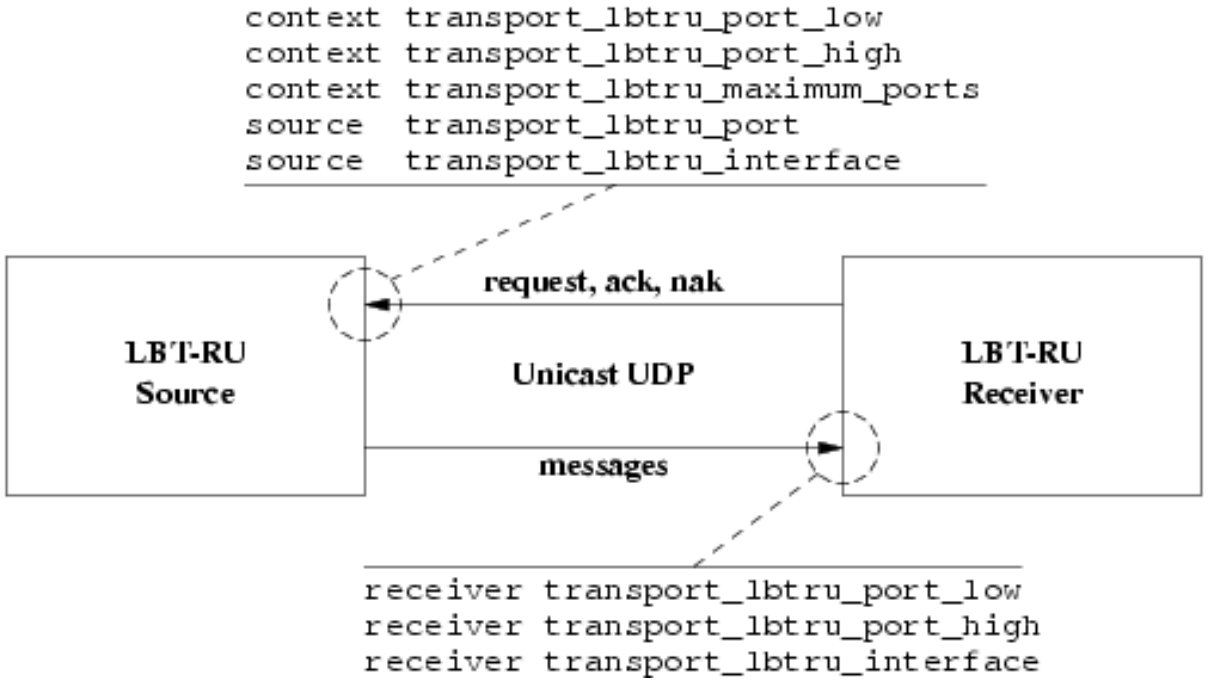
The transmission group size used for this Topic when LBT-RM is used. This value must be greater than 0 and must be a power of 2 no greater than 32K. This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_uint16_t**
Units:	packets
Default value:	8
When to Set:	Can only be set during object initialization.

Transport LBT-RU Network Options

LBT-RU receivers initiate UDP connections toward LBT-RU sources for delivering NAKs and ACKs. LBT-RU sources then initiate connections toward LBT-RU receivers for delivery of UMS messages.

LBT-RU network options



“ transport_lbtru_port_low (context) ” on page 121
is the lowest port that UMS will allocate for LBT-RU sources in a context; “ transport_lbtru_port_high (context) ” on page 120
is the highest. No more than “ transport_lbtru_maximum_ports (context) ” on page 119
ports will be assigned to LBT-RU sources within a single context.

Creation of an UMS source on an LBT-RU transport will allocate an unused UDP port from the range if less than “ transport_lbtru_maximum_ports (context) ” on page 119 ports have already been allocated. Setting “ transport_lbtru_maximum_ports (context) ” on page 119 to a fraction of the range allows the corresponding multiple number of UMS processes to share a common configuration.

If a particular UDP port is desired by a source, it may be given with “ transport_lbtru_port (source) ” on page 120. If the desired port is already in use, then an unused port will be sought as described above. A value of `0` (the default) expresses no preference and results in the default open port seeking behavior described above.

“ transport_lbtru_interface (source) ” on page 119 may be used on LBT-RU sources to choose particular interface, overriding the default `INADDR_ANY` which accepts connections on all interfaces. Similarly, “ transport_lbtru_interface (receiver) ” on page 119 may be used on receivers to choose a particular interface for outgoing connections.

transport_lbtru_interface (receiver)

Specifies the network interface over which UM LBT-RU receivers read application data messages. Can specify full IP address of interface, or just network part (see "Specifying Interfaces" on page 52 for details).

Scope:	receiver
Type:	**lbm_ipv4_address_mask_t**
Default value:	0.0.0.0 (**INADDR_ANY**)
When to Set:	Can only be set during object initialization.

transport_lbtru_interface (source)

Specifies the network interface over which UM LBT-RU sources receive connection requests from topic receivers. Can specify full IP address of interface, or just network part (see "Specifying Interfaces" on page 52 for details). Be aware that this option is applied to the transport session when the first topic is created on that session. Thus, setting a different interface for a subsequent topic that maps onto the same transport session will have no effect. Default is set to INADDR_ANY, meaning that it will accept incoming connection requests from any interface.

Scope:	source
Type:	**lbm_ipv4_address_mask_t**
Default value:	0.0.0.0 (**INADDR_ANY**)
When to Set:	Can only be set during object initialization.

transport_lbtru_maximum_ports (context)

Maximum number of unicast port numbers that LBT-RU will use for all implicitly allocated sessions.

Scope:	context
Type:	**lbm_uint16_t**
Units:	number of ports
Default value:	5
When to Set:	Can only be set during object initialization.

transport_lbtru_port (source)

The preferred unicast port number for this Topic. If 0, the context will attempt to find one in the given LBT-RU source port range.

Scope:	source
Type:	**lbm_uint16_t**
Default value:	0 (use open port)
Byte order:	Network
When to Set:	Can only be set during object initialization.

transport_lbtru_port_high (context)

High unicast port number to assign to LBT-RU sources. Clients send connection requests, ACKs, and NAKs to a port number in this range.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	14389
Byte order:	Host
When to Set:	Can only be set during object initialization.

transport_lbtru_port_high (receiver)

High port number to use for receiving LBT-RU data. All LBT-RU data for the topic will arrive on this range.

Scope:	receiver
Type:	**lbm_uint16_t**
Default value:	14379
Byte order:	Host
When to Set:	Can only be set during object initialization.

transport_lbtru_port_low (context)

Low unicast port number to assign LBT-RU sources to. Clients send connection requests, ACKs, and NAKs to a port number in this range.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	14380
Byte order:	Host
When to Set:	Can only be set during object initialization.

transport_lbtru_port_low (receiver)

Low port number to use for receiving LBT-RU data. All LBT-RU data for the topic will arrive on this range.

Scope:	receiver
Type:	**lbm_uint16_t**
Default value:	14360
Byte order:	Host
When to Set:	Can only be set during object initialization.

Transport LBT-RU Reliability Options

For every LBT-RU reliability option, there is a corresponding LBT-RM reliability option. For more information on how LBT-RU reliability options interact and for illustrations, please see the introduction to the "Transport LBT-RM Reliability Options" on page 105 section.

transport_lbtru_ignore_interval (source)

The interval to ignore NAKs after a retransmission is sent. This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds

Default value:	500 (0.5 seconds)
When to Set:	Can only be set during object initialization.

transport_lbtru_nak_backoff_interval (receiver)

For LBT-RU sessions only. The maximum interval between transmissions of a single NAK. The actual value is random (to reduce self-similar behaviors) and is uniform on the range [0.5*interval, 1.5*interval]. This option affects the transport session underlying the receiver rather than the receiver itself. The transport session uses the value from the first receiver created on the session and ignores subsequent receivers. Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* and "Interrelated Configuration Options" on page 42 for additional information.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	200 [100,300] (0.2 [0.1,0.3] seconds)
When to Set:	Can only be set during object initialization.

transport_lbtru_nak_generation_interval (receiver)

For LBT-RU sessions only. The maximum time that a piece of data may be outstanding before the data is unrecoverably lost. Although the minimum valid value is 5 milliseconds, larger values are advisable. This option affects the transport session underlying the receiver rather than the receiver itself. The transport session uses the value from the first receiver created on the session and ignores subsequent receivers. Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* and "Interrelated Configuration Options" on page 42 for additional information.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10000 (10 seconds)
When to Set:	Can only be set during object initialization.

transport_lbtru_nak_suppress_interval (receiver)

For LBT-RU sessions only. The maximum interval to suppress sending NAKs after an NCF is received. This option affects the transport session underlying the receiver rather than the receiver itself. The transport session uses the value from the first receiver created on the session and ignores subsequent receivers.

Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	1000 (1 second)
When to Set:	Can only be set during object initialization.

transport_lbtru_receiver_socket_buffer (context)

Value used to set **SO_RCVBUF** value of the LBT-RU receiver unicast socket (both sender and receiver sides). In some cases the OS will not allow all of this value to be used. See the section on "Socket Buffer Sizes" on page 52 for platform-dependent information. See also our white paper Topics in High Performance Messaging for background and guidelines on UDP buffer sizing.

Scope:	context
Type:	**lbm_ulong_t**
Units:	bytes
Default value:	524288 (512KB)
When to Set:	Can only be set during object initialization.

transport_lbtru_source_socket_buffer (context)

Value used to set **SO_SNDBUF** value of the LBT-RU send multicast socket. In some cases the OS will not allow all of this value to be used. See the section on "Socket Buffer Sizes" on page 52 for platform-dependent information. A value of 0 instructs UM to use the OS default.

Scope:	context
Type:	**lbm_ulong_t**
Units:	bytes
Default value:	0 (use OS default or 131072, whichever is larger)
When to Set:	Can only be set during object initialization.

transport_lbtru_transmission_window_limit (source)

Caps the total amount of memory that a transmission window uses, which includes data and overhead. For example, if the " transport_lbtru_transmission_window_size (source) " on page 124 is 24 MB (default) and

the source sends 20 byte messages with the "flush" flag, the actual amount of memory used can approximate 300 MB. The default value of this option does not limit the transmission window.

Scope:	source
Type:	**size_t**
Units:	bytes
Default value:	0 (zero)
When to Set:	Can only be set during object initialization.

transport_lbtru_transmission_window_size (source)

The maximum amount of buffered data that the LBT-RU source is allowed to retain for retransmissions. The minimum valid value is 65536 bytes. This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**size_t**
Units:	bytes
Default value:	25165824 (24 MB)
When to Set:	Can only be set during object initialization.

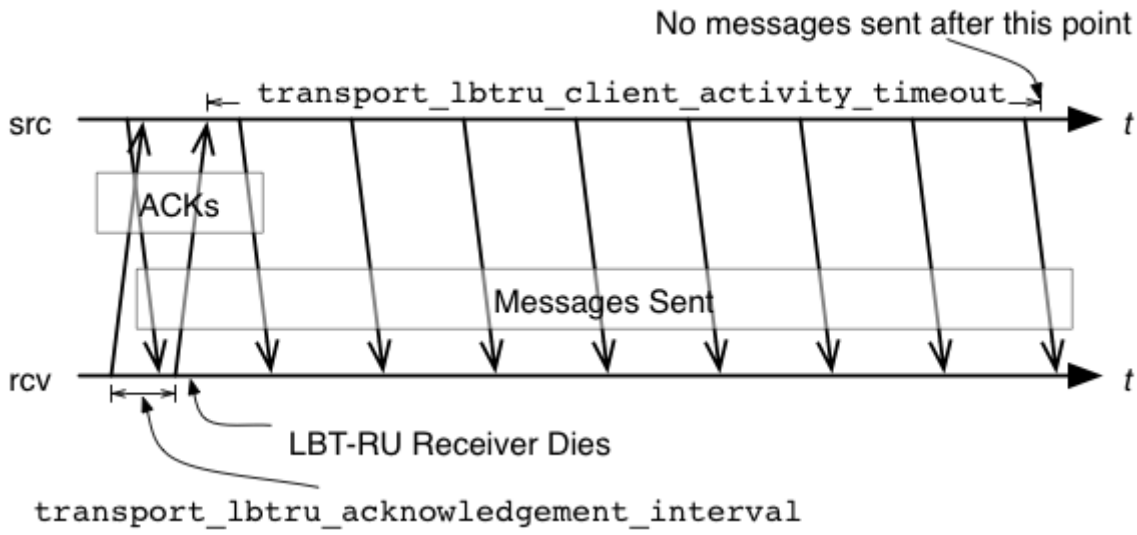
Transport LBT-RU Operation Options

For most LBT-RU operation options, there is a corresponding LBT-RM operation option. For more information on how LBT-RU operation options interact and for illustrations, please see the introduction to the “Transport LBT-RM Operation Options” on page 112 section.

Two options unique to LBT-RU are “ transport_lbtru_client_map_size (source) ” on page 126 and “ transport_lbtru_connect_interval (receiver) ” on page 127.

The illustration below shows the interaction of two more options unique to LBT-RU: “ transport_lbtru_acknowledgement_interval (receiver) ” on page 125 and “ transport_lbtru_client_activity_timeout (source) ” on page 126.

An LBT-RU receiver goes away



transport_lbtru_acknowledgement_interval (receiver)

For LBT-RU session only. The interval between sending acknowledgements. Each client continually sends acknowledgements to let the source know that the client is still alive. This option affects the transport session underlying the receiver rather than the receiver itself. The transport session uses the value from the first receiver created on the session and ignores subsequent receivers. Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	500 (0.5 seconds)
When to Set:	Can only be set during object initialization.

transport_lbtru_activity_timeout (receiver)

For LBT-RU sessions only. The maximum time that an LBT-RU session may be quiescent before it is deleted and an EOS event is delivered for all topics using this transport session. This option affects the transport session underlying the receiver rather than the receiver itself. The transport session uses the value from the

first receiver created on the session and ignores subsequent receivers. Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	60000 (60 seconds)
When to Set:	Can only be set during object initialization.

transport_lbtru_client_activity_timeout (source)

The maximum time that an LBT-RU client may be quiescent, i.e. not sending ACKs, before the sender assumes that it is dead and stops sending to it. This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10000 (10 seconds)
When to Set:	Can only be set during object initialization.

transport_lbtru_client_map_size (source)

The size of the hash table used to store client information and state. This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**size_t**
Units:	table entries
Default value:	7
When to Set:	Can only be set during object initialization.

transport_lbtru_coalesce_threshold (source)

UM passes implicitly batched messages to the Operating System **sendmsg()** as a set unless the size of the set exceeds the coalescing threshold at which point the messages are coalesced and passed to the O/S as one copy.

This option accommodates the different number of iovecs supported by different O/Ss. Tuning this option balances the efficiency of less iovecs handled by the OS vs. the expense of an additional copy operation of the messages before sending. The default value is also the maximum allowable value for Solaris, AIX and HPUX. For Linux and Microsoft® Windows® and Darwin, the maximum allowable value is 1023. These maximum allowable values are one less than what the O/S provides.

This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**int**
Units:	number of messages
Default value:	15
When to Set:	Can only be set during object initialization.

transport_lbtru_connect_interval (receiver)

For LBT-RU session only. The interval between sending connection requests. This option affects the transport session underlying the receiver rather than the receiver itself. The transport session uses the value from the first receiver created on the session and ignores subsequent receivers. Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	100 (0.1 seconds)
When to Set:	Can only be set during object initialization.

transport_lbtru_data_rate_limit (context)

Maximum aggregate transmission rate of all LBT-RU sessions original data for this particular context.

Scope:	context
Type:	**unsigned long int**

Units:	bits per second
Default value:	10000000 (10 Mbps)
When to Set:	Can only be set during object initialization.

transport_lbtru_datagram_max_size (context)

The maximum datagram size that can be generated for a LBT-RU transport session. The default value is 8192, the minimum is 500 bytes, and the maximum is 65535.

Scope:	context
Type:	**lbm_uint_t**
Units:	bytes
Default value:	8192
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1/UME 3.1/UMQ 1.1

transport_lbtru_maximum_connect_attempts (receiver)

The maximum number of connect attempts to make before this transport session is deleted and an EOS event is delivered for all topics using this transport session. This option affects the transport session underlying the receiver rather than the receiver itself. The transport session uses the value from the first receiver created on the session and ignores subsequent receivers. Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	receiver
Type:	**lbm_ulong_t**
Default value:	600
When to Set:	Can only be set during object initialization.

transport_lbtru_rate_interval (context)

Period that LBT-RU rate limiter runs. Reducing period reduces burst intensity, but also increases CPU load.

Scope:	context
Type:	`lbm_ulong_t`
Units:	milliseconds

Default value:	100
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
"10" (Integer value as a string.)	10	LBT-RU rate limiter runs every 10 milliseconds.
"20" (Integer value as a string.)	20	LBT-RU rate limiter runs every 20 milliseconds.
"50" (Integer value as a string.)	50	LBT-RU rate limiter runs every 50 milliseconds.
"100" (Integer value as a string.)	100	LBT-RU rate limiter runs every 100 milliseconds. **Default for all.**

transport_lbtru_retransmit_rate_limit (context)

Maximum aggregate transmission rate of all LBT-RU sessions retransmissions for this particular context. This should always be less than the value used for original data.

Scope:	context
Type:	**unsigned long int**
Units:	bits per second
Default value:	5000000 (5 Mbps)
When to Set:	Can only be set during object initialization.

transport_lbtru_sm_maximum_interval (source)

The maximum interval between LBT-RU session messages. In lieu of data being sent, LBT-RU sends session messages to each client to inform them of sequence numbers and to let receivers know that the sender is still transmitting. This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10000 (10 seconds)
When to Set:	Can only be set during object initialization.

transport_lbtru_sm_minimum_interval (source)

The minimum interval between LBT-RU session messages. In lieu of data being sent, LBT-RU sends session messages to each client to inform them of sequence numbers and to let receivers know that the sender is still transmitting. This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	200 (0.2 seconds)
When to Set:	Can only be set during object initialization.

transport_lbtru_use_session_id (source)

Flag to indicate whether the application desires LBT-RU to use a session ID or not. Older versions of UM may not understand session IDs with LBT-RU and may not be able to receive LBT-RU transport sessions that include session IDs.

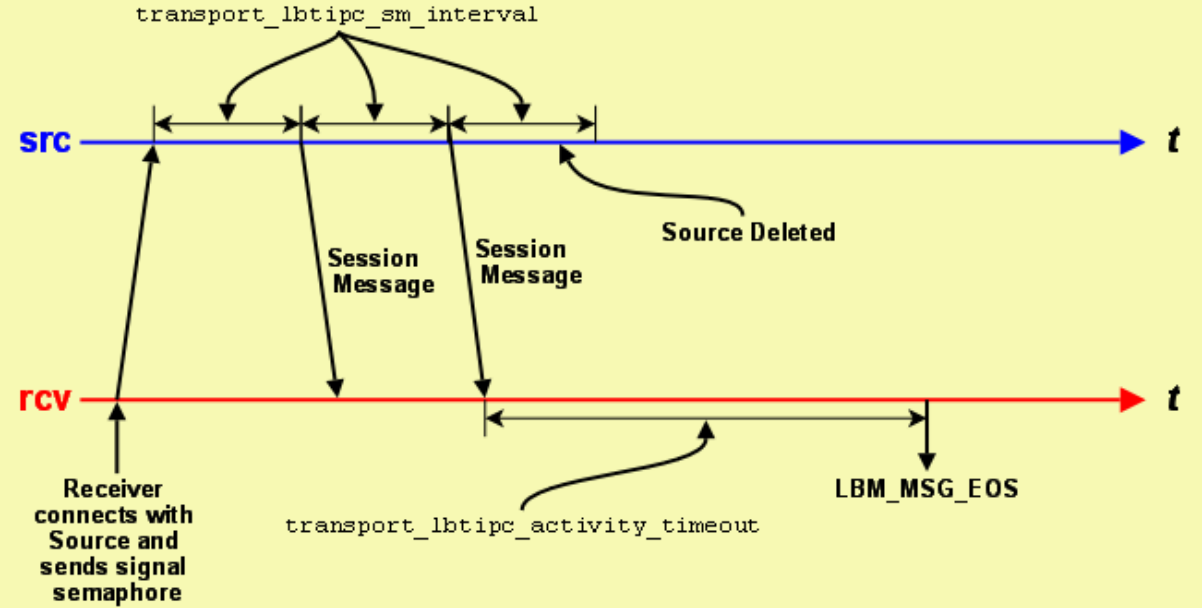
Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	Indicates the application desires LBT-RU to use a session ID. **Default for all.**
0	Indicates the application does not desire LBT-RU to use a session ID.

Transport LBT-IPC Operation Options

The following option descriptions and diagrams describe the Ultra Messaging Configuration Options available for the LBT-IPC transport.

An LBT-IPC source goes away



The Source Session Message mechanism enables the receiver to detect when a source goes away and works similarly to LBT-RU. It operates independently of message writes and reads in the Shared Memory Area.

transport_lbtipc_activity_timeout (receiver)

The maximum period of inactivity (lack of session messages) from an IPC source before the UM delivers an EOS event for all topics using the transport session. Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* and "Interrelated Configuration Options" on page 42 for additional information.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	60,000 (60 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.5ea2/UME 2.2ea1

transport_lbtipc_behavior (source)

Desired flow control behavior when multiple receivers have joined the same LBT-IPC transport session. See also *Transport LBT-IPC* in the *Ultra Messaging Concepts Guide*. This option affects the transport session underlying the source rather than the source itself. The transport session uses the value from the first source created on the session and ignores subsequent sources. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_ushort_t**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

String value	Integer value	Description
source_paced	**LBM_SRC_TOPIC_ATTR_LBTIPC_BEHAVIOR_SOURCE_PACED**	Your application writes as fast as it can to the LBT-IPC shared memory area. Slower receivers can experience loss. A source does not consider if any receivers have successfully read a message before it reclaims it. **Default for all.**
receiver_paced	**LBM_SRC_TOPIC_ATTR_LBTIPC_BEHAVIOR_RECEIVER_PACED**	Your application writes to the LBT-IPC shared memory area only as fast as the slowest receiver consumes data. A source will not reclaim a message until all receivers have successfully read the message. This slows down all receiver on the LBT-IPC transport session.

transport_lbtipc_datagram_max_size (context)

The maximum datagram size that can be generated for a LBT-IPC transport session. The default value is 65535, the minimum is 500 bytes, and the maximum is 65535.

Scope:	context
Type:	**lbm_uint_t**
Units:	bytes
Default value:	65535
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1/UME 3.1/UMQ 1.1

transport_lbtipc_id (source)

The preferred Transport ID for a specific source's LBT-IPC session. If 0, the UM context attempts to find one in the given Transport ID range of " transport_lbtipc_id_low (context) " on page 133 and " transport_lbtipc_id_high (context) " on page 133. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_uint16_t**
Default value:	0 (use open port)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.5ea2/UME 2.2ea1

transport_lbtipc_id_high (context)

Highest transport ID of the range of available LBT-IPC Transport IDs.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	20,005
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.5ea2/UME 2.2ea1

transport_lbtipc_id_low (context)

Lowest transport ID of the range of available LBT-IPC Transport IDs.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	20,001
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.5ea2/UME 2.2ea1

transport_lbtipc_maximum_receivers_per_transport (source)

The maximum number of receiving contexts that can join an IPC transport session. Once a receiving context joins an IPC transport session, it can receive messages on multiple topics. Increasing this value increases the amount of shared memory allocated per transport session by a negligible amount.

Scope:	source
Type:	**lbm_ushort_t**
Default value:	20
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

transport_lbtipc_receiver_operational_mode (context)

The mode in which UM operates to process LBT-IPC messages. See also *Embedded and Sequential Mode* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
embedded	**LBM_CTX_ATTR_OP_EMBEDDED**	UM spawns a thread to process received LBT-IPC messages. **Default for all.**
sequential	**LBM_CTX_ATTR_OP_SEQUENTIAL**	Your application must call **lbm_context_process_lbtipc_messages()** to process received LBT-IPC messages. If you also set the context's " operational_mode (context) " on page 61 option to `sequential`, your application must donate an additional thread to service the **lbm_context_process_events()** calls. **Note:** You can use sequential mode with the C API, but not with the Java API or .NET API. The Java and .NET APIs do not provide an equivalent **lbm_context_process_lbtipc_messages()** API for LBT-IPC.

transport_lbtipc_receiver_thread_behavior (context)

Receiver behavior for monitoring the signaling semaphore set by the IPC source when it writes new data to the shared memory area.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.5ea2/UME 2.2ea1

String value	Integer value	Description
pend	**LBM_CTX_ATTR_IPC_RCV_THREAD_PEND**	Receiver waits (sleep) for notification from OS that IPC source has updated the signaling semaphore. This option is best when the IPC source frequently writes new data to the shared area. **Default for all.**
busy_wait	**LBM_CTX_ATTR_IPC_RCV_THREAD_BUSY_WAIT**	Provides the lowest latency as the receiver monopolizes the CPU core looking for an incremented semaphore. This option works best for infrequent or sporadic message delivery from the IPC source, but involves a CPU cost.

transport_lbtipc_sm_interval (source)

Time period between sessions message sent from source to receivers. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* and "Interrelated Configuration Options" on page 42 for additional information.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10,000 (10 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.5ea2/UME 2.2ea1

transport_lbtipc_transmission_window_size (source)

Size of an LBT-IPC transport's shared memory area. This value may vary across platforms. The actual size of the shared memory area equals the value you specify for this option plus about 64 KB for header information.

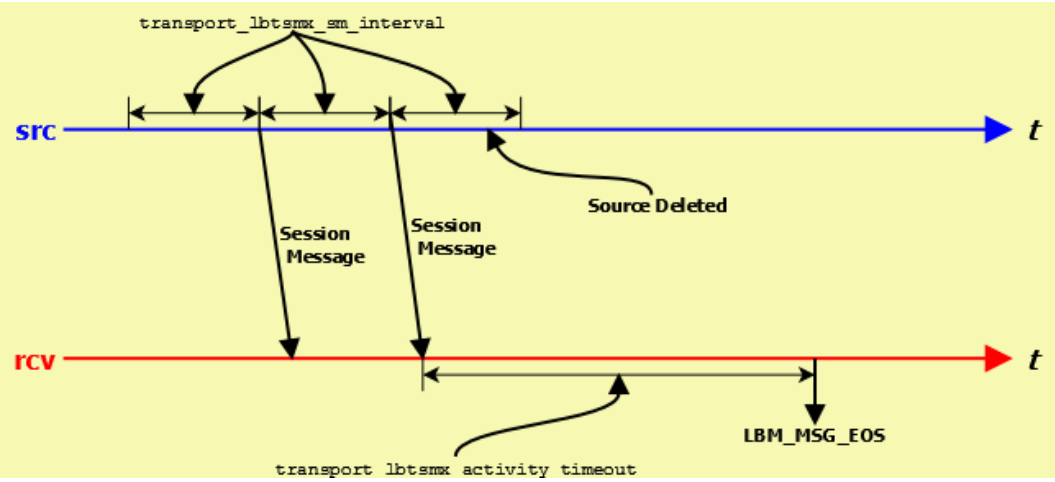
The minimum value for this option is 65,536. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**size_t**
Units:	bytes
Default value:	25165824 (24 MB)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.5ea2/UME 2.2ea1

Transport LBT-SMX Operation Options

The following option descriptions and diagram describe the Ultra Messaging Configuration Options available for the LBT-SMX transport.

Figure 2. LBT-SMX Session Messages



The Source Session Message mechanism enables the receiver to detect when a source goes away and works similarly to LBT-RU. It operates independently of message writes and reads in the Shared Memory Area.

transport_lbtsmx_activity_timeout (receiver)

The maximum period of inactivity (lack of session messages) from an LBT-SMX source before UM delivers an EOS event for all topics using the transport session. You should configure this option to a value greater than the source's `transport_lbtsmx_sm_interval` so receivers do not erroneously report a source as

inactive. Refer to *Receiver Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* and "Interrelated Configuration Options" on page 42 for additional information.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	60,000 (60 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.1

transport_lbtsmx_datagram_max_size (source)

The maximum datagram size that can be generated for a LBT-SMX transport session. This value includes 16 bytes of header information per message, plus an additional 24 bytes of reserved space for compatibility with other egress transports when re-sending SMX messages through a UM Dynamic Router. Therefore, the largest usable message size for the default setting of 8192 bytes would be 8176 bytes (8192 - 16 - 24). The minimum is 32 bytes. The maximum size is limited by available memory.

This option imposes a hard limit on message size because the LBT-SMX transport does not support datagram fragmentation or reassembly. Unlike other transports that do support fragmentation, attempts to send messages larger than the datagram size configured by this option fail.

Scope:	context
Type:	**lbm_uint_t**
Units:	bytes
Default value:	8192
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.1

transport_lbtsmx_id (source)

The preferred Transport ID for a specific source's LBT-SMX session. To use this option, configure a non-zero value. For the default value of 0 (zero), the UM context selects the next available Transport ID in the Transport ID range of `transport_lbtsmx_low` and `transport_lbtsmx_high`. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_uint16_t**
Default value:	0 (zero)

When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.1

transport_lbtsmx_id_high (context)

Highest transport ID in the range of available LBT-SMX Transport IDs.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	30,005
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.1

transport_lbtsmx_id_low (context)

Lowest transport ID in the range of available LBT-SMX Transport IDs.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	30,001
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.1

transport_lbtsmx_maximum_receivers_per_transport (source)

The maximum number of receiving contexts that can join an LBT-SMX transport session. Once a receiving context joins an LBT-SMX transport session, it can receive messages on multiple topics. If you increase this option's value, you increase the amount of shared memory allocated per transport session by a negligible amount.

Scope:	source
Type:	**lbm_ushort_t**
Default value:	64
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.1

transport_lbtsmx_message_statistics_enabled (context)

Controls whether or not UM records LBT-SMX transport statistics, which adds a small but measurable amount of latency.

Scope:	context
Type:	**int**
Default value:	0
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.1

String value	Integer value	Description
"1" (Integer value as a string.)	**1**	UM records source and receiver LBT-SMX transport statistics.
"0" (Integer value as a string.)	**0**	UM does not record source and receiver LBT-SMX transport statistics.**Default for all.**

transport_lbtsmx_sm_interval (source)

Time period between updates to an LBT-SMX source's shared activity counter, which enables connected receivers to determine the source's liveness. You should configure this option to a value less than the receivers' corresponding `transport_lbtsmx_activity_timeout` setting so receivers do not time out sources too early. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10,000 (10 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.1

transport_lbtsmx_transmission_window_size (source)

Size of an LBT-SMX transport's shared memory area, which must be a power of two and be twice a large as the source's `transport_lbtsmx_datagram_max_size`. If you configure a value that is not a power of 2 or is less than twice the size of the maximum datagram size, UM issues a warning log message and automatically rounds up the value of this option to the next power of 2 window size that can fit at least two maximum-sized

datagrams. The minimum value for this option is 64 bytes. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**size_t**
Units:	bytes
Default value:	131072 (128 KB)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.1

Transport LBT-RDMA Operation Options

Use of the LBT-RDMA transport requires the purchase and installation of the Ultra Messaging RDMA Transport Module. See your Ultra Messaging representative for licensing specifics.

See also *Transport LBT-RDMA* in the *Ultra Messaging Concepts Guide*.

transport_lbtrdma_datagram_max_size (context)

The maximum datagram size that can be generated for a LBT-RDMA transport session. The default value is 4096, the minimum is 500 bytes, and the maximum is 4096.

Scope:	context
Type:	**lbm_uint_t**
Units:	bytes
Default value:	4096
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1/UME 3.1/UMQ 1.1

transport_lbtrdma_interface (source)

Specifies the network interface over which UM LBT-RDMA sources receive connection requests from topic receivers. You can specify the full IP address of the interface, or just the network part (see "Specifying Interfaces" on page 52 for details). Be aware that the first source joining a transport session sets the interface with this option. Thus, setting a different interface for a subsequent topic that maps onto the same transport

session will have no effect. Default is set to INADDR_ANY, meaning that it accepts incoming connection requests from any interface.

Scope:	source
Type:	**lbm_ipv4_address_mask_t**
Default value:	0.0.0.0 (**INADDR_ANY**)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1

transport_lbtrdma_maximum_ports (context)

Maximum number of LBT-RDMA sessions to allocate.

Scope:	context
Type:	**lbm_uint16_t**
Units:	number of ports
Default value:	5
When to Set:	Can only be set during object initialization.

transport_lbtrdma_port (source)

Port number for a specific source's LBT-RDMA session that is outside the " transport_lbtipc_id_low (context) " on page 133 and " transport_lbtipc_id_high (context) " on page 133 range. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_uint16_t**
Default value:	0 (zero)
Byte order	Host
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1

transport_lbtrdma_port_high (context)

Highest port number that can be assigned to a LBT-RDMA session.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	20,020
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1

transport_lbtrdma_port_low (context)

Lowest port number that can be assigned to a LBT-RDMA session.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	20,001
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1

transport_lbtrdma_receiver_thread_behavior (context)

Receiver behavior for monitoring a LBT-RDMA source's shared memory area for new data.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1

String value	Integer value	Description
pend	**LBM_CTX_ATTR_RDMA_RCV_THREAD_PEND**	Receiver waits (sleep) for notification from RDMA that the source has updated the shared memory area with new data. Default. **Default for all.**
busy_wait	**LBM_CTX_ATTR_RDMA_RCV_THREAD_BUSY_WAIT**	UM polls the shared memory area for new data.

transport_lbtrdma_transmission_window_size (source)

Size of an LBT-RDMA transport's shared memory area. This value may vary across platforms. The actual size of the shared memory area equals the value you specify for this option plus about 64 KB for header information. The minimum value for this option is 65,536. Refer to *Source Configuration and Transport Sessions* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**size_t**
Units:	bytes
Default value:	25165824 (24 MB)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1

Transport Acceleration Options

Transport acceleration options enable kernel-bypass acceleration in conjunction with the following vendor solutions:

- Myricom® Datagram Bypass Layer (DBL™)
- Solarflare® Onload
- Mellanox® 10-Gigabit Ethernet or InfiniBand hardware

Myricom® Datagram Bypass Layer (DBL™)

DBL is a kernel-bypass technology that accelerates sending and receiving UDP traffic and operates with DBL-enabled Myricom 10-Gigabit Ethernet adapter cards for Linux and Microsoft® Windows.

DBL does not support fragmentation and reassembly, so do not send messages larger than the MTU size configured on the DBL interface.

DBL acceleration is compatible with the following Ultra Messaging transport types.

- LBT-RM (UDP-based reliable multicast)
- LBT-RU (UDP-based reliable unicast)
- Multicast Immediate Messaging
- Multicast Topic Resolution

To enable DBL Transport Acceleration:

1. Install the Myricom 10-Gigabit Ethernet NIC.
2. Install the DBL shared library.
3. Update your search path to include the location of the DBL shared library.
4. Set “ transport_datagram_max_size (context) ” on page 228 to a value of no more than 28 bytes smaller than the Myricom interface's configured MTU size.

Solarflare® Onload

Solarflare Onload is a kernel-bypass technology that accelerates message traffic and operates with Solarflare 10GbE Ethernet NICs.

Onload does not support fragmentation and reassembly, so do not send messages larger than the MTU size configured on the Solarflare interface.

UM loads the Onload library dynamically during UM initialization on the following UM platforms:

- Linux-glibc-2.3-i686
- Linux-glibc-2.3-x86_64
- Linux-glibc-2.5-x86_64

Note: If you set the LBM_SUPPRESS_ONLOAD environment variable to any value, UM does not dynamically load the Onload library at runtime.

Onload default behavior accelerates all sockets. You can access the Onload **onload_set_stackname** API extension to select the sockets you want to accelerate by using UM configuration options. Selecting sockets with a stackname ensures that you accelerate data transmission sockets and not sockets for control messages, topic resolution, or responses.

You can select a stackname with the configuration option *onload_acceleration_stack_name* in both the source and receiver scope for the following Ultra Messaging transport types.

- LBT-RM (UDP-based reliable multicast)
- LBT-RU (UDP-based reliable unicast)
- TCP

If you use the **onload_set_stackname** API directly for any other accelerated sockets, note that after UM accelerates a transport socket, UM resets the stackname to the default for all threads by calling:

onload_set_stackname(ONLOAD_ALL_THREADS, ONLOAD_SCOPE_NOCHANGE, "");

UM resets the stackname during source creation and when a receiver matched topic opens a transport session.

To enable Onload selective socket acceleration,

1. Install Onload.
2. Set the Onload environment variable EF_DONT_ACCELERATE = 1 to disable Onload default behavior.
3. Set UM configuration option *(source) onload_acceleration_stack_name* according to the thread the source uses.
4. Set UM configuration option *(receiver) onload_acceleration_stack_name* according to the thread the receiver uses.
5. Set “ transport_datagram_max_size (context) ” on page 228 to a value of no more than 28 bytes smaller than the Solarflare interface's configured MTU size.

For detailed information about **onload_set_stackname**, refer to the Solarflare® Onload User Guide.

UD Acceleration for Mellanox® Hardware Interfaces

UD (Unreliable Datagram) acceleration is a kernel-bypass technology that accelerates sending and receiving UDP traffic and operates with Mellanox 10-Gigabit Ethernet or InfiniBand adapter cards for 64-bit Linux on X86 platforms.

UD acceleration does not support fragmentation and reassembly, so do not send messages larger than the MTU size configured on the Mellanox interface.

UD acceleration is available for the following Ultra Messaging transport types.

- LBT-RM (UDP-based reliable multicast)
- LBT-RU (UDP-based reliable unicast)
- Multicast Immediate Messaging
- Multicast Topic Resolution

To enable UD acceleration,

1. Install the Mellanox NIC.
2. Install the VMA package, which is part of the UD acceleration option .
3. Include the appropriate transport acceleration options in your Ultra Messaging Configuration File .
4. Set " transport_datagram_max_size (context) " on page 228 to a value of no more than 28 bytes smaller than the Mellanox interface's configured MTU size.

resolver_ud_acceleration (context)

Flag indicating if Accelerated Multicast is enabled for Topic Resolution. Accelerated Multicast requires Mellanox InfiniBand or 10 Gigabit Ethernet hardware, and the purchase and installation of the Ultra Messaging Accelerated Multicast Module. See your Ultra Messaging representative for licensing specifics. UD Acceleration of topic resolution relies on hardware-supported loopback, which InfiniBand provides, but which the 10 Gigabit Ethernet ConnectX hardware does not provide.

- For InfiniBand, set this option to 1 to enable.
- For 10 Gigabit Ethernet ConnectX, set this option to 0 to disable.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 5.2.

Value	Description
1	Accelerated Topic Resolution is enabled.
0	Accelerated Topic Resolution is not enabled. **Default for all.**

ud_acceleration (context)

Flag indicating if Accelerated Multicast is enabled for LBT-RM. Accelerated Multicast requires InfiniBand or 10 Gigabit Ethernet hardware and the purchase and installation of the Ultra Messaging Accelerated Multicast Module. See your Ultra Messaging representative for licensing specifics.

Scope:	context
Type:	**int**

When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1.

Value	Description
1	Accelerated Multicast is enabled.
0	Accelerated Multicast is not enabled. **Default for all.**

onload_acceleration_stack_name (receiver)

The `stackname` to use when creating an OpenOnload® transport data socket. The `stackname` must be eight characters or less. Since this is a transport setting, the first receiver applies its configuration and all other subsequent receivers on the same transport inherit the original setting. A special, case sensitive string, `NULL`, disables the `stackname`.

Note: Use of this option requires Solarflare® OpenOnload® and applies to LBT-RM, LBT-RU, and TCP transports.

Scope:	source
Type:	**string**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.5.

onload_acceleration_stack_name (source)

The `stackname` to use when creating an OpenOnload® transport data socket. The `stackname` must be eight characters or less. Since this is a transport setting, the first source applies its configuration and all other subsequent sources on the same transport inherit the original setting. A special, case sensitive string, `NULL`, disables the `stackname`.

Note: Use of this option requires Solarflare® OpenOnload® and applies to LBT-RM, LBT-RU, and TCP transports.

Scope:	source
Type:	**string**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.5.

Multicast Immediate Messaging Network Options

The multicast address and port used for incoming and outgoing multicast immediate messages can be set with “ mim_address (context) ” on page 147 and “ mim_destination_port (context) ” on page 147. A context may use different multicast addresses and/or ports for incoming and outgoing messages by setting “ mim_incoming_address (context) ” on page 148, “ mim_outgoing_address (context) ” on page 148, “ mim_incoming_destination_port (context) ” on page 148, and/or “ mim_outgoing_destination_port (context) ” on page 148. In case of conflict, the most recently set option wins.

As with LBT-RM on multi-homed hosts, the interface UM uses for MIM follows the interface used with multicast topic resolution. See “ resolver_multicast_interface (context) ” on page 88.

Warning: The addresses and ports you configure for MIM traffic should not overlap with any addresses or ports - or address and port ranges - configured for LBT-RM transports or Topic Resolution traffic. For example, do not use the same multicast address for both Topic Resolution (`resolver_multicast_address`) and MIM (`mim_address`). Use different addresses and ports for all multicast address options and port options.

See also *Multicast Immediate Messaging* in the *Ultra Messaging Concepts Guide* for more information about this feature.

mim_address (context)

The IP multicast address that multicast immediate messages are sent to and received from.

Scope:	context
Type:	**struct in_addr**
Default value:	224.10.10.21
When to Set:	Can only be set during object initialization.

mim_destination_port (context)

The UDP destination port that multicast immediate messages are sent to and received from.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	14401
Byte order:	Network
When to Set:	Can only be set during object initialization.

mim_incoming_address (context)

The IP multicast address that multicast immediate messages are received from. Setting this option to 0.0.0.0 turns off multicast immediate messaging. (MIM).

Scope:	context
Type:	**struct in_addr**
Default value:	224.10.10.21
When to Set:	Can only be set during object initialization.

mim_incoming_destination_port (context)

The UDP destination port that multicast immediate messages are received from.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	14401
Byte order:	Network
When to Set:	Can only be set during object initialization.

mim_outgoing_address (context)

The IP multicast address that multicast immediate messages are sent to.

Scope:	context
Type:	**struct in_addr**
Default value:	224.10.10.21
When to Set:	Can only be set during object initialization.

mim_outgoing_destination_port (context)

The UDP destination port that multicast immediate messages are sent to.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	14401

Byte order:	Network
When to Set:	Can only be set during object initialization.

Multicast Immediate Messaging Reliability Options

For every MIM reliability option, there is a corresponding LBT-RM reliability option. For more information on how MIM reliability options interact and for illustrations, please see the introduction to "Transport LBT-RM Reliability Options" on page 105.

See also *Multicast Immediate Messaging* in the *Ultra Messaging Concepts Guide* for more information about this feature.

mim_ignore_interval (context)

For multicast immediate message senders only. See " transport_lbtrm_ignore_interval (source) " on page 108 for description.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	500 (0.5 seconds)
When to Set:	Can only be set during object initialization.

mim_nak_backoff_interval (context)

For multicast immediate message receivers only. See " transport_lbtrm_nak_backoff_interval (receiver) " on page 108 for description.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	200 (0.2 seconds)
When to Set:	Can only be set during object initialization.

mim_nak_generation_interval (context)

For multicast immediate message receivers only. See “ transport_lbtrm_nak_generation_interval (receiver) ” on page 109 for description.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10000 (10 seconds)
When to Set:	Can only be set during object initialization.

mim_nak_initial_backoff_interval (context)

For multicast immediate message receivers only. See “ transport_lbtrm_nak_initial_backoff_interval (receiver) ” on page 109 for description.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	50 (0.05 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.4/UME 2.1.

mim_nak_suppress_interval (context)

For multicast immediate message receivers only. See “ transport_lbtrm_nak_suppress_interval (receiver) ” on page 110 for description.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	1000 (1 second)
When to Set:	Can only be set during object initialization.

mim_send_naks (context)

For multicast immediate message receivers only. See “ transport_lbtrm_send_naks (receiver) ” on page 110 for description.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	NAKs are sent for missing packets to request retransmission. **Default for all.**
0	Do not send NAKs for missing packets.

mim_transmission_window_limit (context)

For multicast immediate message senders only. See “ transport_lbtrm_transmission_window_limit (source) ” on page 111 for description.

Scope:	context
Type:	**size_t**
Units:	bytes
Default value:	0 (zero)
When to Set:	Can only be set during object initialization.

mim_transmission_window_size (context)

For multicast immediate message senders only. See “ transport_lbtrm_transmission_window_size (source) ” on page 112 for description.

Scope:	context
Type:	**size_t**
Units:	bytes
Default value:	25165824 (24 MB)
When to Set:	Can only be set during object initialization.

Multicast Immediate Messaging Operation Options

For many MIM operation options, there is a corresponding LBT-RM operation option. For more information on how MIM operation options interact and for illustrations, please see the introduction to "Transport LBT-RM Operation Options" on page 112.

Note that the LBT-RM rate controller also governs MIM transmission rates. Hence there is no separate option for setting MIM transmission rate.

See also *Multicast Immediate Messaging* in the *Ultra Messaging Concepts Guide* for more information about this feature.

immediate_message_receiver_function (context)

Callback function (and associated event queue and client data pointer) called when a topic-less immediate message is received for which there is no receiver. A value of NULL for the callback prevents the callback from being called.

Scope:	context
Type:	**lbm_context_rcv_immediate_msgs_func_t**
Default value:	NULL
When to Set:	Can only be set during object initialization.
Config File:	Cannot be set from an UM configuration file.

immediate_message_topic_receiver_function (context)

Callback function (and associated event queue and client data pointer) that is called when an immediate message is received for a topic for which there is no receiver. A value of NULL for the callback prevents the callback from being called.

Scope:	context
Type:	**lbm_context_rcv_immediate_msgs_func_t**
Default value:	NULL
When to Set:	Can only be set during object initialization.
Config File:	Cannot be set from an UM configuration file.

mim_activity_timeout (context)

For multicast immediate message receivers only. See " transport_lbtrm_activity_timeout (receiver) " on page 113 for description. However, multicast immediate message channels do not deliver an EOS indication.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	60000 (60 seconds)
When to Set:	Can only be set during object initialization.

mim_delivery_control_activity_check_interval (context)

The interval between activity checks of a Multicast Immediate Messaging delivery controller. Multiple MIM delivery controllers may exist to accommodate multiple messages from a single MIM sender received across more than one UM Router. These multiple delivery controllers allow for duplicate message detection.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	5000 (5 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0.

mim_delivery_control_activity_timeout (context)

The maximum time that a Multicast Immediate Messaging delivery controller may be quiescent before it is deleted. MIM delivery controllers may be created to accommodate multiple messages from a single MIM sender received across more than one UM Router. These multiple delivery controllers allow for duplicate message detection.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	60000 (60 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0.

mim_delivery_control_order_tablesz (context)

For multicast immediate messages with ordered delivery, this controls the size of the hash table used to hold data.

Scope:	context
Type:	**size_t**
Units:	table entries
Default value:	1031
When to Set:	Can only be set during object initialization.

mim_implicit_batching_interval (context)

The maximum timeout between when the first message of an implicitly batched immediate message is queued until the batch is sent. A message will not stay in the queue longer than this value before being sent in the worst case.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	200 (0.2 seconds)
When to Set:	Can only be set during object initialization.

mim_implicit_batching_minimum_length (context)

The minimum length of an implicitly batched multicast immediate message. When the total length of the implicitly batched messages reaches or exceeds this value, the batch is sent.

Scope:	context
Type:	**size_t**
Units:	bytes
Default value:	2048 (8192 for Microsoft™ Windows™)
When to Set:	Can only be set during object initialization.

mim_ordered_delivery (context)

For multicast immediate messages only. Indicates whether or not the MIM source should have its data delivered in order. The default value also guarantees fragmentation and reassembly of large messages. Changing this option from the default value results in large messages being delivered as individual fragments

of less than 8K each, requiring the application to reassemble them. See also *Ordered Delivery* in the *Ultra Messaging Concepts Guide* for more information about large message fragmentation and reassembly.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	Indicates the source should have its data delivered in order. **Default for all.**
0	The source should have its data delivered as soon as possible and may come in out of order.

mim_sm_maximum_interval (context)

For multicast immediate message senders only. See " transport_lbtrm_sm_maximum_interval (source) " on page 116 for description.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10000 (10 seconds)
When to Set:	Can only be set during object initialization.

mim_sm_minimum_interval (context)

For multicast immediate message senders only. See " transport_lbtrm_sm_minimum_interval (source) " on page 117 for description.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	200 (0.2 seconds)
When to Set:	Can only be set during object initialization.

mim_sqn_window_increment (context)

For multicast immediate message receivers only. Determines the increment by which the sequence number window is moved when detecting the receipt of duplicate multicast immediate messages. Must be a multiple of 8 and an even divisor of “ mim_sqn_window_size (context) ” on page 156.

Scope:	context
Type:	**lbm_ulong_t**
Units:	messages
Default value:	8192
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2.8/UME 3.2.8/UMQ 2.1.8

mim_sqn_window_size (context)

For multicast immediate message receivers only. Determines the window size used to detect the receipt of duplicate multicast immediate messages. Must be a multiple of 8.

Scope:	context
Type:	**lbm_ulong_t**
Units:	messages
Default value:	16384
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2.8/UME 3.2.8/UMQ 2.1.8

mim_src_deletion_timeout (context)

The timeout after a multicast immediate message is sent before the internal source is deleted and cleaned up.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	30000 (30 seconds)
When to Set:	Can only be set during object initialization.

mim_tgsz (context)

For multicast immediate message senders only. See “ transport_lbtrm_tgsz (source) ” on page 117 for description.

Scope:	context
Type:	**lbm_uint16_t**
Units:	packets
Default value:	8
When to Set:	Can only be set during object initialization.

mim_unrecoverable_loss_function (context)

Callback function (and associated client data pointer) that is called when a MIM receiver has unrecoverable loss. This callback is called directly in line and does not use the event queue. Therefore the callback function used should not block or it will block the context thread processing. A value of NULL for the callback turns off the callback being called.

Scope:	context
Type:	**lbm_mim_unrecloss_func_t**
Default value:	NULL
When to Set:	Can only be set during object initialization.
Config File:	Cannot be set from an UM configuration file.

Late Join Options

Late Join Recovery

Overview

Late Join allows sources to save a predefined amount of their messaging traffic for late-joining receivers. Sources set the configuration options that determine whether they use Late Join or not, and receivers set options that determine whether they will participate in Late Join recovery if sources use Late Join.

UMP's persistent store is built on Late Join technology. In the *Estimating Recovery Time* discussion below, the terms *Late Join buffers* and *UMP store* are roughly equivalent.

For more, review the *Late Join* in the *Ultra Messaging Concepts Guide*, especially *Configuring Late Join for Large Numbers of Messages*.

Estimating Recovery Time

Late Join message recovery time is a function of how much data must be recovered and how fast messages are retransmitted. To estimate Late Join recovery time R in minutes, use the formula: **R = D / (1 - (txrate / rxrate))** where:

- **D** is the downtime (in minutes) across all receivers
- **txrate** is the average source transmission rate of normal (live stream) messages during recovery (in kmsgs/sec).
- **rxrate** is the average source retransmission rate from source-side Late Join buffers during recovery (in kmsgs/sec). This rate needs to be greater than txrate.

For example, consider the following scenario:

- **D** = 10 minutes
- **txrate** = 10k messages / second
- **rxrate** = 25k messages / second

Plugging these values into the formula gives an estimated recovery time in minutes: **R = 10 / (1 - (10 / 25))** or 16.67 minutes. Note that this formula assumes the following:

- Retransmit rate**(rxrate)** is as linear as possible with use of option response_tcp_nodelay 1
- Transmit rate **(txrate)** from *all* relevant sources is fairly constant and equal
- Retransmit rate **(rxrate)** from Late Join buffers is fairly constant and equal, and should be measured in a live test, if possible. You can adjust the recovery rate with two Late Join configuration options:
  - “ retransmit_request_outstanding_maximum (receiver) ” on page 161
  - “ retransmit_request_interval (receiver) ” on page 160

late_join (source)

Configure the source to enable both Late Join and Off-Transport Recovery (OTR) operation for receivers. See *Late Join* and *Off-Transport Recovery* in the *Ultra Messaging Concepts Guide*.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	Enable source for Late Join and OTR. (Forced on for UMP.)
0	Disable source for Late Join and OTR. **Default for all.**

late_join_info_request_interval (receiver)

The interval at which the receiver requests a Late Join Information Record (LJI) from the source. Controlling these requests helps reduce receiver start-up traffic on your network.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	1000 (1 second)
When to Set:	Can only be set during object initialization.
Version	This option was implemented in UMP 6.0

late_join_info_request_maximum (receiver)

The maximum number of requests the receiver issues for a Late Join Information Record (LJI) from the source. If the receiver has not received an LJI after this number of requests, it stops requesting.

Scope:	source
Type:	**lbm_ulong_t**
Default value:	60
When to Set:	Can only be set during object initialization.
Version	This option was implemented in UMP 6.0

retransmit_initial_sequence_number_request (receiver)

When a late-joining receiver detects (from the topic advertisement) that a source is enabled for Late Join but has sent no messages, this flag option lets the receiver request an initial sequence number from a source. Sources respond with a TSNI.

Scope:	receiver
Type:	**int**
Default value:	1

When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2.

Value	Description
1	The receiver requests an initial sequence number from Late Join enabled sources that have not sent any messages. **Default for all.**
0	The receiver does not request an initial sequence number.

retransmit_message_caching_proximity (receiver)

This option enables receiver caching of new messages during a recovery. The option value determines how close or proximate the current new sequence number must be to the latest retransmitted sequence number for the receiver to start caching. The receiver recovers uncached data later in the recovery process by the retransmit request mechanism. An option value greater than or equal to the default turns on caching of new data immediately. A smaller value means that caching does not begin until recovery has caught up somewhat with the source. A larger value means that caching can begin earlier during recovery. This value has meaning for only receivers using ordered delivery of data. See *Configuring Late Join for Large Numbers of Messages* in the *Ultra Messaging Concepts Guide* for additional information about this option.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	messages
Default value:	2147483647
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.3.2/UME 2.0.

retransmit_request_interval (receiver)

The interval between retransmission request messages to the source. See *Configuring Late Join for Large Numbers of Messages* in the *Ultra Messaging Concepts Guide* for additional information about this option.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	500 (0.5 seconds)
When to Set:	Can only be set during object initialization.

retransmit_request_maximum (receiver)

The maximum number of messages to request, counting backward from the current latest message, when late-joining a topic. Due to network timing factors, UM may transmit an additional message. For example, a value of 5 sends 5 or possibly 6 retransmit messages to the new receiver. (Hence, you cannot request and be guaranteed to receive only 1 last message--you may get 2.) A value of 0 indicates no maximum.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	messages
Default value:	0
When to Set:	Can only be set during object initialization.

retransmit_request_message_timeout (receiver)

The maximum time from when a receiver first sends a retransmission request to when the receiver gives up on receiving the retransmitted message and reports loss.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10000 (10 seconds)
When to Set:	Can only be set during object initialization.
version:	This option was implemented in UM 6.0.

retransmit_request_outstanding_maximum (receiver)

The maximum number of messages to request and to remain active at a single time. A value of 0 indicates no maximum. See *Configuring Late Join for Large Numbers of Messages* in the *Ultra Messaging Concepts Guide* for additional information about this option.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	messages
Default value:	200
When to Set:	Can only be set during object initialization.

retransmit_retention_age_threshold (source)

Specifies the minimum age of messages in the retained message buffer before UM can delete them. UM cannot delete any messages younger than this value. For UMS Late Joins, this and " retransmit_retention_size_threshold (source) " on page 162 are the only options that affect the retention buffer size. For UMP, these two options combined with " retransmit_retention_size_limit (source) " on page 162 affect the retention buffer size. UM deletes a message when it meets all configured threshold criteria, i.e., the message is older than this option (if set), and the size of the retention buffer exceeds the `retransmit_retention_size_threshold` (if set). A value of 0 sets the age threshold to be always triggered, in which case deletion is determined by other threshold criteria.

Scope:	source
Type:	**lbm_ulong_t**
Units:	seconds
Default value:	0 (threshold always triggered)
When to Set:	Can only be set during object initialization.

retransmit_retention_size_limit (source)

Sets a maximum limit on the size of the source's retransmit retention buffer when using a UMP store. With UMP, stability and delivery confirmation events can delay the deletion of retained messages, which can increase the size of the buffer above the " retransmit_retention_size_threshold (source) " on page 162. Hence, this option provides a hard size limit. UM sets a minimum value for this option of 8K for UDP and 64K for TCP, and issues a log warning if you set a value less than the minimum.

Scope:	source
Type:	**size_t**
Units:	bytes
Default value:	25165824 (24 MB)
When to Set:	Can only be set during object initialization.

retransmit_retention_size_threshold (source)

Specifies the minimum size of the retained message buffer before UM can delete messages. The buffer must reach this size before UM can delete any messages older than " retransmit_retention_age_threshold (source) " on page 162. For UMP, these options combined with " retransmit_retention_size_limit (source) " on page 162 affect the retention buffer size. A value of 0 sets the size threshold to be always triggered, in which case deletion is determined by other threshold criteria.

Scope:	source
Type:	**size_t**

Units:	bytes
Default value:	0 (threshold always triggered)
When to Set:	Can only be set during object initialization.

use_late_join (receiver)

Flag indicating if the receiver should participate in a late join operation or not.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	The receiver will participate in using late join if requested to by the source. **Default for all.**
0	The receiver will not participate in using late join even if requested to by the source.

Off-Transport Recovery Options

See also *Off-Transport Recovery (OTR)* in the *Ultra Messaging Concepts Guide* for more information about this feature.

otr_request_initial_delay (receiver)

The length of time a receiver waits before initiating OTR lost-message requests.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10000 (10 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 5.3

otr_request_log_alert_cooldown (receiver)

Each OTR request generates a log message. The first request's log message is a WARNING-level log message, and subsequent requests that quickly follow generate INFO-level log messages. After a time

interval defined by this option, the next request leading a new burst of requests again generates a WARNING-level log message.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	seconds
Default value:	300 (5 minutes)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 5.3

otr_request_maximum_interval (receiver)

The maximum time interval between a receiver's OTR lost-message requests. After the receiver initiates OTR and is waiting to receive the retransmission, the initial interval (set by " otr_request_minimum_interval (receiver) " on page 165) doubles in length for each request until it reaches this option's value, then continues at this interval (until timeout or UM recovers messages). NOTE: When using TCP Request/ Response, this value must be shorter than " response_tcp_deletion_timeout (context) " on page 170.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10000 (10 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 5.3

otr_message_caching_threshold (receiver)

This option sets the maximum number of messages a receiver can buffer. When the number of cached messages hits this threshold, Ultra Messaging drops and does not cache new streamed messages. Dropped messages can be requested later as retransmissions.

This option applies for only receivers using sequence-number ordered delivery of data.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	messages
Default value:	10000

When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.0

otr_request_message_timeout (receiver)

The maximum time from when a receiver first sends an OTR lost-message request to when the receiver gives up on receiving the retransmitted message and reports loss.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	20000 (20 seconds)
When to Set:	Can only be set during object initialization.
version:	This option was implemented in UM 6.0.

otr_request_minimum_interval (receiver)

The initial time interval between a receiver's OTR lost-message requests. While the receiver is waiting to receive the retransmission, the interval doubles in length for each request until it reaches the maximum interval set by " otr_request_maximum_interval (receiver) " on page 164.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	1000 (1 second)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 5.2

otr_request_outstanding_maximum (receiver)

The maximum number of OTR lost-message requests outstanding at any given time. Each message specifies an individual lost message. A value of 0 indicates no maximum.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	messages

Default value:	200
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 5.2

use_otr (receiver)

Flag indicating if the receiver can use OTR or not.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 5.2

Value	Description
1	The receiver is enabled to use OTR to recover lost messages.
0	The receiver is not enabled to use OTR to recover lost messages. **Default for all.**

Request Network Options

See also *Request/Response* in the *Ultra Messaging Concepts Guide* for more information about this feature.

request_tcp_bind_request_port (context)

Allows you to turn off request port binding. Setting this option to 0 prevents sockets from being bound to the request port. Turning off request port binding also turns off the UM features such as: *Request/Response*, *Late Join*, *OTR*, the reception of Unicast Immediate Messages and Unicast Immediate Request, along with UMP and UMQ.

Scope:	context
Type:	**int**
Default value:	1

When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.3.7/UME 2.0.5.

Value	Description
1	Set request port binding. **Default for all.**
0	Turn off request port binding.

request_tcp_interface (context)

Specifies the network interface over which UM accepts TCP connections in response to requests it has sent out. You can specify a full IP address of interface, or just the network part (see "Specifying Interfaces" on page 52 for details). Default is set to INADDR_ANY, meaning that it will not bind to a specific interface. You can also modify the default by setting the option to 0.0.0.0/0 which produces the same result.

Scope:	context
Type:	**lbm_ipv4_address_mask_t**
Default value:	0.0.0.0 (**INADDR_ANY**)
When to Set:	Can only be set during object initialization.

request_tcp_port (context)

Port number used for listening for responses from requests. If 0, use a random open port within the range of [" request_tcp_port_low (context) " on page 168, " request_tcp_port_high (context) " on page 167]. If nonzero, the specific port number is used instead. Each UM context will bind to a TCP port for requests when it is initialized.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	0 (use open port)
Byte order:	Network
When to Set:	Can only be set during object initialization.

request_tcp_port_high (context)

High port number to use for listening for responses from requests.

Scope:	context
Type:	**lbm_uint16_t**

Default value:	14395
Byte order:	Host
When to Set:	Can only be set during object initialization.

request_tcp_port_low (context)

Low port number to use for listening for responses from requests.

Scope:	context
Type:	**lbm_uint16_t**
Default value:	14391
Byte order:	Host
When to Set:	Can only be set during object initialization.

Request Operation Options

See also *Request/Response* in the *Ultra Messaging Concepts Guide* for more information about this feature.

request_tcp_exclusiveaddr (context)

Applicable only to Windows. Indicate whether the TCP accepting socket should set **SO_EXCLUSIVEADDRUSE** or not before it binds. The default setting in Windows allows multiple binds to the same port. By default, UM will set SO_EXCLUSIVEADDRUSE to minimize port sharing. Refer to Microsoft's web site for more information on SO_EXCLUSIVEADDRUSE.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	Set **SO_EXCLUSIVEADDRUSE**. **Default for Windows.**
0	Do not set **SO_EXCLUSIVEADDRUSE**.

request_tcp_listen_backlog (context)

The backlog used in the TCP **listen()** call to set the queue length for incoming connections.

Scope:	context
Type:	**int**
Default value:	5
When to Set:	Can only be set during object initialization.

request_tcp_reuseaddr (context)

Whether the TCP accepting socket should set **SO_REUSEADDR** or not before it binds. NOTE: For Microsoft® Windows®, UM always forces this value to "0" regardless of the value set in any configuration files.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	Set **SO_REUSEADDR**.
0	Do not set **SO_REUSEADDR**. **Default for all.**

Response Operation Options

See also *Request/Response* in the *Ultra Messaging Concepts Guide* for more information about this feature.

response_session_maximum_buffer (context)

Value used to control the maximum amount of data buffered in UM for each response session (unicast connection to a requester).

Scope:	context
Type:	**lbm_ulong_t**
Units:	bytes
Default value:	65536
When to Set:	Can only be set during object initialization.

response_session_sender_socket_buffer (context)

Value used to set the **SO_SNDBUF** value of the response session (unicast connection to a requester). In some cases the OS will not allow all of this value to be used. A value of 0 instructs UM to use the OS defaults. See the section on "Socket Buffer Sizes" on page 52 for platform-dependent information.

Scope:	context
Type:	**lbm_ulong_t**
Units:	bytes
Default value:	0 (use OS defaults)
When to Set:	Can only be set during object initialization.

response_tcp_deletion_timeout (context)

After UM deletes a TCP response, this is the timeout period after which UM closes the connection and reclaims its memory. NOTE: When using Off-Transport Recovery, this value must be longer than " otr_request_maximum_interval (receiver) " on page 164.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	2000 (2 seconds)
When to Set:	Can only be set during object initialization.

response_tcp_interface (context)

Specifies the network interface over which UM initiates TCP connections for responses. You can specify the full IP address of interface, or just the network part (see "Specifying Interfaces" on page 52 for details). Default is set to INADDR_ANY, meaning that it will not bind to a specific interface. You can also modify the default by setting the option to 0.0.0.0/0 which produces the same result.

Scope:	context
Type:	**lbm_ipv4_address_mask_t**
Default value:	0.0.0.0 (**INADDR_ANY**)
When to Set:	Can only be set during object initialization.

response_tcp_nodelay (context)

Whether the TCP sockets used for sending responses should set **TCP_NODELAY** or not. (Setting TCP_NODELAY disables Nagle's algorithm.)

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	TCP response sockets should set **TCP_NODELAY** (disable Nagle).
0	TCP response sockets should not set **TCP_NODELAY** (leave Nagle enabled). **Default for all.**

Implicit Batching Options

implicit_batching_interval (source)

The maximum timeout between when the first message of an implicit batch is queued until the batch is sent. A message will not stay in the queue longer than this value before being sent in the worst case. Refer to *Message Batching* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	200 (0.2 seconds)
When to Set:	May be set during operation.

implicit_batching_minimum_length (source)

The minimum length of an implicitly batched message. When the total length of the implicitly batched messages reaches or exceeds this value, the batch is sent. Refer to *Message Batching* in the *Ultra Messaging Concepts Guide* for additional information.

Scope:	source
Type:	**size_t**

Units:	bytes
Default value:	2048 (8192 for Microsoft™ Windows™)
When to Set:	May be set during operation.

implicit_batching_type (source)

The implicit batching algorithm to use which controls when messages sent on a transport session are flushed or batched, if batching is in use.

Scope:	source
Type:	**int**
When to Set:	May be set during operation.

String value	Integer value	Description
default	**LBM_SRC_TOPIC_ATTR_IMPLICIT_BATCH_TYPE_DEFAULT**	Implicit batching is controlled entirely by the implicit_batching_minimum_length (source)on page 171 and " implicit_batching_interval (source) " on page 171 options. Refer to *Message Batching* for additional information. **Default for all.**
adaptive	**LBM_SRC_TOPIC_ATTR_IMPLICIT_BATCH_TYPE_ADAPTIVE**	Source-paced batching method that attempts to adjust the amount of messages sent in each batch automatically. The options, and implicit_batching_minimum_length (source)on page 171" implicit_batching_interval (source) " on page 171, limit batch sizes and intervals but sizes and intervals will usually be much smaller. Setting this option may have a negative impact on maximum throughput.

Delivery Control Options

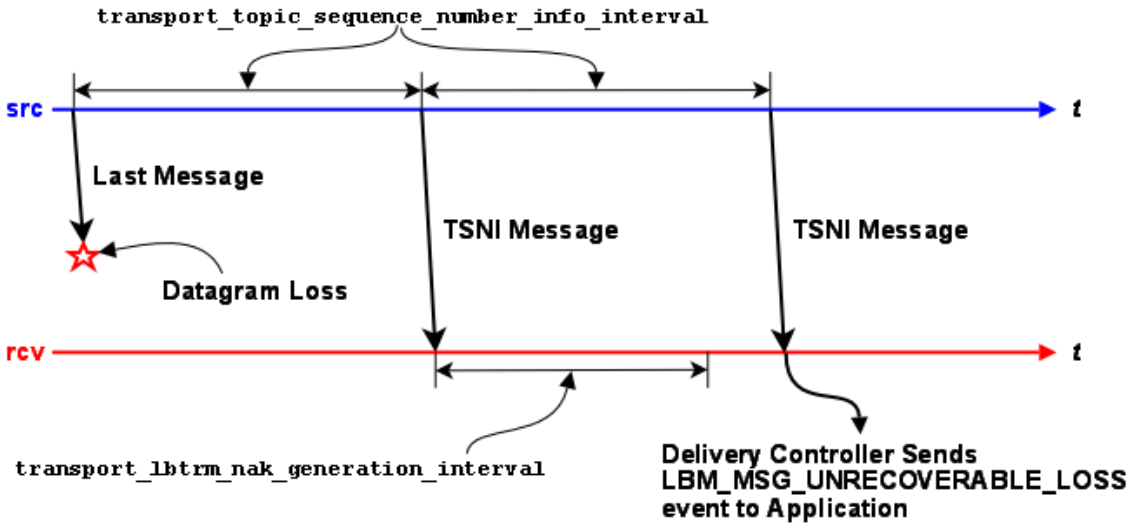
A Delivery Controller is a receiver-side object created for each source identified by the receiver through topic resolution. A delivery controller performs the following.

- Delivers messages to multiple receivers subscribed to the same topic.
- Orders received topic messages if " ordered_delivery (receiver) " on page 61 is set to 1 (default). This option applies to LBT-RU and LBT-RM transports.

- Determines unrecoverable loss and burst loss events for the receiver's topic over LBT-RU and LBT-RM transports.

Unlike the loss depicted in *LBT-RM Datagram Loss Resulting in Unrecovered Message Loss* which is due to the inability of the transport or network to perform message retransmission, Generation of Unrecoverable Loss Eventon page 173 demonstrates how a receiver's Delivery Controller detects the loss of a topic message and notifies the receiving application. The *TSNI* messages contain the sequence number of the last message sent by the source.

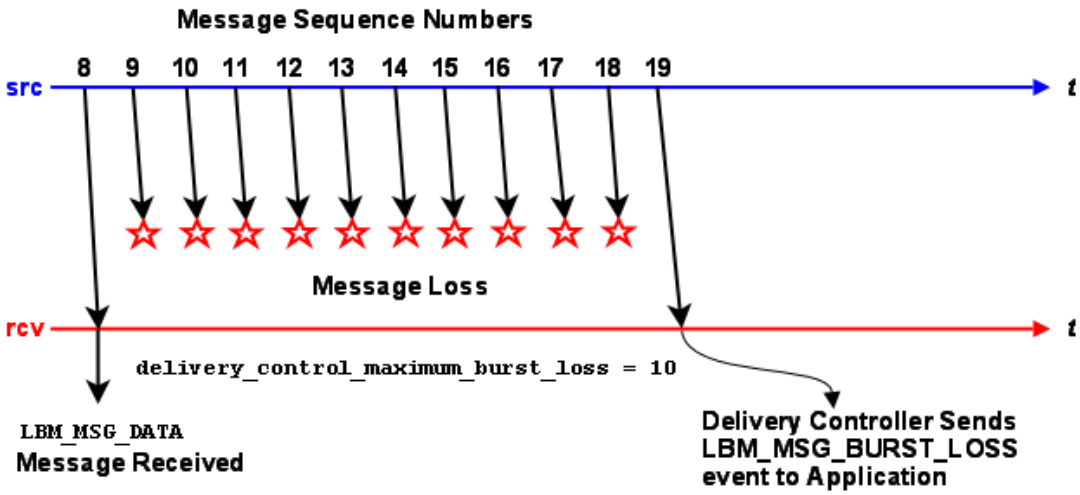
Generation of Unrecoverable Loss Event



The Delivery Controller detects burst loss by comparing the sequence numbers of the last two messages received. If the resulting gap in sequence numbers equals or exceeds the

“ delivery_control_maximum_burst_loss (receiver) ” on page 175, the delivery controller sends `LBM_MSG_BURST_LOSS` to the application.

Generation of Burst Loss Event



channel_map_tablesz (receiver)

The size of the hash table that the receiver uses to store channel subscriptions. A larger table means more channels can be stored more efficiently, but takes up more memory. A smaller table uses less memory, but costs more CPU time for large numbers of channel subscriptions.

Scope:	receiver
Type:	**size_t**
Default value:	10273
When to Set:	Can only be set during object initialization.

delivery_control_loss_check_interval (receiver)

This controls the interval between mandatory topic loss checks for a receiver. A value of 0 turns this loss check off.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds

Default value:	0 (disabled)
When to Set:	Can only be set during object initialization.

delivery_control_loss_tablesz (receiver)

For LBT-RM and other datagram-based transport sessions only. This controls the size of the hash table index used for storing unrecoverable loss state on a per source per topic basis. Larger values mean larger hash tables and probably better CPU usage under loss scenarios at the cost of more memory per source per topic. Smaller values mean smaller hash tables and probably worse CPU usage under loss scenarios but with less memory usage. The value used should be a prime number for efficiency.

Scope:	receiver
Type:	**size_t**
Units:	table entries
Default value:	131
When to Set:	Can only be set during object initialization.
Version:	Deprecated

delivery_control_maximum_burst_loss (receiver)

This controls the maximum tolerable burst loss before a burst loss message is delivered to the application. A burst loss less than or equal to this size is treated normally. Larger burst loss is treated as unrecoverable immediately. When using OTR, set this to a significantly high value to let OTR recover lost messages.

Scope:	receiver
Type:	**lbm_uint_t**
Units:	number of messages
Default value:	512
When to Set:	Can only be set during object initialization.

delivery_control_maximum_total_map_entries (context)

The maximum total buffered map entries (unrecoverable loss messages as well as data) that all topics can buffer. When this is exceeded, unrecoverable loss is signaled for data until the total buffered subsides. A value of 0 implies no maximum value setting and allows any amount required to be buffered.

If you use OTR with cache management (otr_message_caching), consider disabling this option (set to 0).

Scope:	context
Type:	**size_t**
Units:	map entries
Default value:	200000
When to Set:	Can only be set during object initialization.

delivery_control_message_batching (context)

Controls whether or not to use receive-side batching, which can improve receiver throughput when using event queues, but might add latency in other cases.

If you enable this option, and you use an event queue that is in polling mode, using `lbm_event_dispatch(evq, LBM_EVENT_QUEUE_POLL)`, then rather than dispatching exactly one event per call to `lbm_event_dispatch`, you may get multiple events dispatched with a single call.

Scope:	context
Type:	**int**
Units:	0 or 1
Default value:	0 (receive-side batching not enabled)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMS 6.7.

delivery_control_order_tablesz (receiver)

For LBT-RM and other datagram-based transport sessions only. This controls the size of the hash table index used for storing buffered data on a per source per topic basis when ordered delivery is used. Larger values mean larger hash tables and probably better CPU usage under loss scenarios at the cost of more memory per source per topic. Smaller values mean smaller hash tables and probably worse CPU usage under loss scenarios but with less memory usage. The value used should be a prime number for efficiency.

Scope:	receiver
Type:	**size_t**
Units:	table entries
Default value:	131
When to Set:	Can only be set during object initialization.
Version:	Deprecated

mim_delivery_control_loss_check_interval (context)

This controls the interval between mandatory loss checks for a Multicast Immediate Messaging (MIM) transport session. A value of 0 turns this loss check off.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	0 (disabled)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2

null_channel_behavior (receiver)

Behavior desired when a message without channel information (i.e. a standard UM message) is received by UM.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
deliver	**LBM_RCV_TOPIC_ATTR_CHANNEL_BEHAVIOR_DELIVER_MSGS**	Messages sent without channel information will be delivered to the callback specified upon receiver creation. **Default for all.**
discard	**LBM_RCV_TOPIC_ATTR_CHANNEL_BEHAVIOR_DISCARD_MSGS**	Messages sent without channel information will be discarded.

source_notification_function (receiver)

Callback functions (and associated client data pointer) that are called when a receiver creates or deletes a delivery controller associated with a source. For the creation function, the application has the ability to set the source client data pointer to be used in each message received from the source. This callback is called directly in line and does not use the event queue. Therefore the callback function used should not block or it will block the context thread processing. A value of NULL for the callback turns off the callback being called.

Scope:	receiver
Type:	**lbm_rcv_src_notification_func_t**

Default value:	NULL
When to Set:	Can only be set during object initialization.
Config File:	Cannot be set from an UM configuration file.

unrecognized_channel_behavior (receiver)

Behavior desired when a message with channel information for a channel not in the receiver's set of subscribed channels is received by UM.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
deliver	**LBM_RCV_TOPIC_ATTR_CHANNEL_BEHAVIOR_DELIVER_MSGS**	Messages sent with channel information for a channel not in the receiver's set of subscribed channels will be delivered to the callback specified upon receiver creation. **Default for all.**
discard	**LBM_RCV_TOPIC_ATTR_CHANNEL_BEHAVIOR_DISCARD_MSGS**	Messages sent with channel information for a channel not in the receiver's set of subscribed channels will be discarded.

Wildcard Receiver Options

pattern_callback (wildcard_receiver)

Callback function (and associated client data pointer) that is called when a pattern match is desired for a topic discovered for a wildcard receiver if the pattern type is set to "appcb". This callback is called directly in line and does not use the event queue. A return value of 0 indicates the given topic should be considered part of the wildcard. A value of 1 or more indicates the topic should NOT be considered matching the wildcard.

Scope:	wildcard_receiver
Type:	**lbm_wildcard_rcv_compare_func_t**
Default value:	NULL

When to Set:	Can only be set during object initialization.
Config File:	Cannot be set from an UM configuration file.

pattern_type (wildcard_receiver)

The type of expression UM uses to compare wildcard receiver patterns to new topics seen in topic advertisements or responses to wildcard receiver queries. As of UM Version 6.1, wildcard receivers must use **pcre** expressions.

Scope:	wildcard_receiver
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
pcre	**LBM_WILDCARD_RCV_PATTERN_TYPE_PCRE**	The pattern is a regular expression usable by PCRE (Perl Compatible Regular Expressions) library. **Default for all.**
regex Deprecated in UM Version 6.1.	**LBM_WILDCARD_RCV_PATTERN_TYPE_REGEX**	The pattern is a regular expression usable by POSIX Extended Regular Expressions.
appcb Deprecated in UM Version 6.1.	**LBM_WILDCARD_RCV_PATTERN_TYPE_APP_CB**	The wildcard receiver ignores the pattern and calls an application callback set by the `pattern_callback` option.

receiver_create_callback (wildcard_receiver)

Callback function (and associated client data pointer) that is called when a receiver is about to be created for a topic which matched a wildcard receiver pattern. This callback is called directly in line and does not use the event queue. The callback function should always return 0.

Scope:	wildcard_receiver
Type:	**lbm_wildcard_rcv_create_func_t**
Default value:	NULL
When to Set:	Can only be set during object initialization.
Config File:	Cannot be set from an UM configuration file.
Version:	This option was implemented in LBM 3.4/UME 2.1.

receiver_delete_callback (wildcard_receiver)

Callback function (and associated client data pointer) that is called when a receiver is about to be deleted. This callback is called directly in line and does not use the event queue. The callback function should always return 0.

Scope:	wildcard_receiver
Type:	**lbm_wildcard_rcv_delete_func_t**
Default value:	NULL
When to Set:	Can only be set during object initialization.
Config File:	Cannot be set from an UM configuration file.
Version:	This option was implemented in LBM 3.4/UME 2.1.

resolver_no_source_linger_timeout (wildcard_receiver)

This sets the linger timeout value before a topic with no sources is removed and cleaned up. Since wildcard receivers set the “ resolution_no_source_notification_threshold (receiver) ” on page 71 to 10, the linger timer starts after the wildcard receiver sends 10 queries and subsequently receives a no-source notification.

Scope:	wildcard_receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	1000 (1 second)
When to Set:	Can only be set during object initialization.

resolver_query_maximum_interval (wildcard_receiver)

The longest - and last - interval in wildcard receiver topic querying. A value of 0 disables wildcard receiver topic querying. See also “Disabling Aspects of Topic Resolution ” on page 39.

Scope:	wildcard_receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	1000 (1 second)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_query_minimum_duration (wildcard_receiver)

The duration of wildcard queries in wildcard receiver topic querying. Only PCRE and regex pattern types can use wildcard queries. A value of 0 guarantees that wildcard receiver topic querying never completes.

Scope:	wildcard_receiver
Type:	**lbm_ulong_t**
Units:	seconds
Default value:	60 (1 minute)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_query_minimum_interval (wildcard_receiver)

Interval between the first topic query sent upon creation of the wildcard receiver and the second query sent by the receiver. A value of 0 disables wildcard receiver topic querying. See also "Disabling Aspects of Topic Resolution " on page 39. This option has an effective minimum of 30 ms. See "Minimum Values for Advertisement and Query Intervals" on page 70.

Scope:	wildcard_receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	50 (0.05 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_wildcard_queries_per_second (context)

Maximum number of queries sent within a one second period during wildcard receiver topic querying. A value of 0 sets no rate limit on queries in wildcard receiver topic querying.

Scope:	context
Type:	**lbm_ulong_t**
Units:	advertisements
Default value:	0
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_wildcard_query_bps (context)

Maximum query rate during wildcard receiver topic querying. A value of 0 sets no rate limit on queries in wildcard receiver topic querying.

Scope:	context
Type:	**lbm_uint64_t**
Units:	bits per second
Default value:	1000000
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0

resolver_wildcard_receiver_map_tablesz (context)

The size of the hash table used for storing wildcard receiver patterns. A value of 0 disables caching wildcard receiver patterns. This value should be a prime number.

Scope:	context
Type:	**size_t**
Units:	map entries
Default value:	10273
When to Set:	Can only be set during object initialization.

Event Queue Options

event_queue_name (event_queue)

The name of an event queue, limited to 128 alphanumeric characters, hyphens or underscores.

Scope:	event_queue
Type:	**string**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.3/UMP 3.3/UMQ 2.3.

queue_age_enabled (event_queue)

Controls whether the length of time each event spends on the event queue is measured. Useful only if you are monitoring event queue statistics.

Scope:	event_queue
Type:	**int**
Default value:	0
When to Set:	May be set during operation.

Value	Description
1	Enables measuring of event queue entry ages.
0	Disables measuring of event queue entry ages. **Default for all.**

queue_cancellation_callbacks_enabled (event_queue)

Flag indicating whether the event queue is to do appropriate locking to provide cancellation callback support for cancel/delete functions.

Scope:	event_queue
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	Provide support for cancellation callbacks.
0	Do not provide cancellation callback support. **Default for all.**

queue_count_enabled (event_queue)

Controls whether the numbers of each type of queue entry are counted. Useful only if you are monitoring event queue statistics.

Scope:	event_queue
Type:	**int**

Default value:	0
When to Set:	May be set during operation.

Value	Description
1	Enables counting event queue entries.
0	Disables counting of event queue entries. **Default for all.**

queue_delay_warning (event_queue)

The event queue delay threshold (in microseconds) at which the monitor function for the event queue is called. This delay is the time that an event has been queued before being dispatched. A value of 0 indicates the event queue delay is not to be monitored and checked.

Scope:	event_queue
Type:	**lbm_ulong_t**
Units:	microseconds
Default value:	0 (not monitored)
When to Set:	May be set during operation.

queue_enqueue_notification (event_queue)

Flag indicating whether to call the monitor function when an event is enqueued into the given event queue. The thread enqueuing the event is the one that calls this function. So, when this is called, the monitoring function in use should only assume this is only notification of enqueuing. The monitor function should not dispatch events directly.

Scope:	event_queue
Type:	**int**
When to Set:	May be set during operation.

Value	Description
1	Enable notification.
0	Disable notification. **Default for all.**

queue_objects_purged_on_close (event_queue)

Flag indicating whether the event queue should be immediately purged of any pending events associated with a recently closed object (e.g. source, receiver) during the close operation, or be left on the queue to be

discarded as the event queue drains normally. In either case, UM does not deliver the defunct events to the application. The `Immediate purge` setting reclaims memory immediately, while the `Delay purge` setting spreads the reclamation work over time, reducing the CPU impact of closing objects associated with the queue.

Scope:	event_queue
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	Immediate purge. **Default for all.**
0	Delay purge.

queue_service_time_enabled (event_queue)

Controls whether the amount of time required to service each event on the event queue is measured. Useful only if you are monitoring event queue statistics.

Scope:	event_queue
Type:	**int**
Default value:	0
When to Set:	May be set during operation.

Value	Description
1	Enables measuring of event queue service times.
0	Disables measuring of event queue service times. **Default for all.**

queue_size_warning (event_queue)

The event queue size threshold (in number of events) at which the monitor function for the event queue is called. A value of 0 indicates the event queue size is not to be monitored and checked.

Scope:	event_queue
Type:	**lbm_ulong_t**
Units:	number of events
Default value:	0 (not monitored)
When to Set:	May be set during operation.

Ultra Messaging Persistence Options

ume_ack_batching_interval (context)

The interval between checks by UMP of consumed, unacknowledged messages. See also “ ume_use_ack_batching (receiver) ” on page 211.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	100 (0.1 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMS 5.0, UMP 5.0, UMQ 5.0.

ume_activity_timeout (receiver)

Establishes the period of time from a receiver's last activity to the release of the receiver's Reg ID. Stores return an error to any new request for the receiver's Reg ID during this period. Overrides the `receiver-activity-timeout` setting configured for the receiver's topic on the store. The default value of 0 (zero) disables this option. See also *Proxy Sources* in the *Ultra Messaging Guide for Persistence*.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	0 (zero)
When to Set:	Can only be set during object initialization.

ume_activity_timeout (source)

Establishes the period of time from a source's last activity to the release of the source's Reg ID. Stores return an error to any new source requesting the source's Reg ID during this period. If proxy sources are enabled (“ ume_proxy_source (source) ” on page 194), the store does not release the source's Reg ID and UMP elects a proxy source. Overrides the `source-activity-timeout` setting configured for the source's topic on the store. The default value of 0 (zero) disables this option. If neither proxy sources nor “ ume_state_lifetime (source) ” on page 208 are configured, the store also deletes the source's state and cache. See also *Proxy Sources* in the *Ultra Messaging Guide for Persistence*.

Scope:	source
Type:	**lbm_ulong_t**

Units:	milliseconds
Default value:	0 (zero)
When to Set:	Can only be set during object initialization.

ume_allow_confirmed_delivery (receiver)

Specifies whether or not UMP allows the sending of confirmed delivery notifications back to the source.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMS 5.0, UMP 5.0, UMQ 5.0.

Value	Description
1	Indicates that UMP can send confirmed delivery notifications. **Default for all.**
0	Indicates that UMP can not send confirmed delivery notifications.

ume_application_outstanding_maximum (receiver)

This UMP receiver option enables the UMP Throttled Delivery feature and sets an upper threshold on the number of message fragments from a single source that are delivered or in an event queue, but not yet consumed. When the number of message fragments exceeds this threshold, the receiver stops buffering all incoming message fragments. Thus, messages from the source transport stream might be dropped and recovered via OTR or UMP late-join mechanisms.

This feature effectively limits the recovery rate and live stream rate to the receiver message consumption rate. If OTR is disabled for the receiver, this threshold applies only during initial Late Join recovery. Setting this option to 0 (zero) disables the UMP Throttled Delivery feature.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	message fragments
Default value:	0 (disabled)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMP 6.7

ume_confirmed_delivery_notification (source)

Flag indicating the source is interested in receiving notifications of delivery of messages to receivers (confirmed delivery) via the source event mechanism. When turned off, receivers do not send delivery confirmation notifications to the source unless the release policy dictates the need for them.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
0	**LBM_SRC_TOPIC_ATTR_UME_CDELV_EVENT_NONE**	The source does not wish to receive delivery confirmation notifications.
1	**LBM_SRC_TOPIC_ATTR_UME_CDELV_EVENT_PER_FRAGMENT**	The source wishes to receive delivery confirmation notifications for all messages and message fragments. **Default for all.**
2	**LBM_SRC_TOPIC_ATTR_UME_CDELV_EVENT_PER_MESSAGE**	The source wishes to receive only one delivery confirmation for a message regardless of how many fragments it comprised.
3	**LBM_SRC_TOPIC_ATTR_UME_CDELV_EVENT_FRAG_AND_MSG**	The source wishes to receive delivery confirmation notifications for all messages and message fragments. In addition, the notification contains a WHOLE_MESSAGE_CONFIRMED flag when the last fragment of a message has been delivered.

ume_consensus_sequence_number_behavior (receiver)

The behavior that the receiver will follow when determining the consensus sequence number used as the sequence number to begin reception at upon re-registration after a failure or suspension. This setting is only used when quorum-consensus is also used on the source.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
lowest	**LBM_RCV_TOPIC_ATTR_UME_QC_SQN_BEHAVIOR_LOWEST**	Consensus is determined as the lowest of the latest sequence numbers seen from any store.
majority	**LBM_RCV_TOPIC_ATTR_UME_QC_SQN_BEHAVIOR_MAJORITY**	Consensus is determined as the latest sequence number agreed upon by the majority of stores within a group. Between groups, the latest of all majority decisions is used. **Default for all.**
highest	**LBM_RCV_TOPIC_ATTR_UME_QC_SQN_BEHAVIOR_HIGHEST**	Consensus is determined as the highest of the latest sequence numbers seen from any store.

ume_consensus_sequence_number_behavior (source)

The behavior that the source will follow when determining the consensus sequence number used as the first message of a source upon re-registration after a failure or suspension. This setting is only used when quorum-consensus is also used.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
lowest	**LBM_SRC_TOPIC_ATTR_UME_QC_SQN_BEHAVIOR_LOWEST**	Consensus is determined as the lowest of the latest sequence numbers seen from any store.
majority	**LBM_SRC_TOPIC_ATTR_UME_QC_SQN_BEHAVIOR_MAJORITY**	Consensus is determined as the latest sequence number agreed upon by the majority of stores within a group. Between groups, the latest of all majority decisions is used. **Default for all.**
highest	**LBM_SRC_TOPIC_ATTR_UME_QC_SQN_BEHAVIOR_HIGHEST**	Consensus is determined as the highest of the latest sequence numbers seen from any store.

ume_explicit_ack_only (receiver)

Flag indicating if the receiver should automatically send acknowledgements to any stores and to the source or if the application desires to explicitly generate acknowledgements itself. See also *Explicit Acknowledgments* in the *Ultra Messaging Guide for Persistence*.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	The receiving application will generate acknowledgements explicitly and the UMP receiver should not automatically generate them.
0	The UMP receiver will automatically generate and send acknowledgements based on message consumption. **Default for all.**

ume_flight_size (source)

Specifies the number of messages allowed to be in flight (unstabilized at a store and without delivery confirmation) before a new message send either blocks or triggers a notification (source event).

Scope:	source
Type:	**unsigned int**
Units:	messages
Default value:	1000
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1.1/UME 3.1.1

ume_flight_size_behavior (source)

The behavior that UMP follows when a message send exceeds the source's " ume_flight_size (source) " on page 191.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.1.1/UME 3.1.1

String value	Integer value	Description
Block	**LBM_FLIGHT_SIZE_BEHAVIOR_BLOCK**	The send call blocks when a source sends a message that exceeds its flight size. If the source uses a non-blocking send, the send returns an LBM_EWOULD_BLOCK. **Default for all.**
Notify	**LBM_FLIGHT_SIZE_BEHAVIOR_NOTIFY**	A message send that exceeds the configured flight size does not block but triggers a flight size notification (source event), indicating that the flight size has been surpassed. UMP also sends a source event notification if the number of in-flight messages falls below the configured flight size.

ume_flight_size_bytes (source)

Specifies the message payload in bytes allowed to be in flight (unstabilized at a store and without delivery confirmation) before a new message send either blocks or triggers a notification source event. UMP monitors both this option and " ume_flight_size (source) " on page 191. If either threshold is met, the configured

blocking or notification behavior executes. See “ ume_flight_size_behavior (source) ” on page 191. When using Receiver-paced Persistence, set this option greater than 0 (zero) but less than or equal to the repository's `source-flight-size-bytes-maximum` value.

Scope:	source
Type:	**lbm_uint64_t**
Units:	bytes
Default value:	0 (disabled)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMP 5.3

ume_force_reclaim_function (source)

Callback function (and associated client data pointer) that is called when a source is forced to release a retained message due to size limitations specified. This callback is called directly in line and does not use the event queue. Therefore the callback function used should not block. A value of NULL for the callback turns off the callback being called.

Scope:	source
Type:	**lbm_ume_src_force_reclaim_func_t**
Default value:	NULL
When to Set:	Can only be set during object initialization.
Config File:	Cannot be set from an UM configuration file.

ume_late_join (source)

Flag indicating the source should allow late join operation for receivers and persistent stores. This is a compatibility setting. The “ late_join (source) ” on page 158 setting should be used instead.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	The source allows late join receivers and persistent stores.
0	The source does not allow late join receivers or persistent stores. **Default for all.**

ume_message_stability_lifetime (source)

The total time in milliseconds from the initial send of a message before a UMP source gives up entirely on receiving a stability acknowledgement for the message. The source then delivers a forced reclaim notice to the application. This option is part of the Proactive Retransmissions feature.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	1200000 (20 minutes)
When to Set:	Can only be set during object initialization.
Version	This option was implemented in UM 6.0

ume_message_stability_notification (source)

Flag indicating the source is interested in receiving notifications of message stability from persistent stores via the source event mechanism. Even when turned off, stores continue to send message stability notifications to the source for retention purposes. However, no notification will be delivered to the application.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
0	**LBM_SRC_TOPIC_ATTR_UME_STABLE_EVENT_NONE**	The source does not wish to receive message stability notifications from the store.
1	**LBM_SRC_TOPIC_ATTR_UME_STABLE_EVENT_PER_FRAGMENT**	The source wishes to receive all message and message fragment stability notifications from the store. **Default for all.**

String value	Integer value	Description
2	**LBM_SRC_TOPIC_ATTR_UME_STABLE_EVENT_PER_MESSAGE**	The source wishes to receive only a single message stability notifications from the store when the entire message has been stabilized. This notification contains the Sequence Number of the last fragment of the whole message but does NOT contain store information.
3	**LBM_SRC_TOPIC_ATTR_UME_STABLE_EVENT_FRAG_AND_MSG**	The source wishes to receive all message and message fragment stability notifications from the store. In addition, the notification contains a WHOLE_MESSAGE_STABLE flag when the last fragment of a message has been stabilized.

ume_message_stability_timeout (source)

The time in milliseconds from initial send of a message until it is resent by the source because the source has not received a stability acknowledgement for the store (or a quorum of stores). Setting this option to 0 (zero) disables the Proactive Retransmissions feature.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	20000 (20 seconds)
When to Set:	Can only be set during object initialization.
Version	This option was implemented in UM 6.0

ume_proxy_source (source)

Controls whether any stores with which the source registers should provide a proxy source in the event the actual source terminates. Proxy source support is only available for quorum/consensus store configurations. In addition, proxy source support requires that the source register with an actual registration ID, and not request that the store assign it a registration ID.

Scope:	source
Type:	**int**

Default value:	0
When to Set:	Can only be set during object initialization.

Value	Description
1	Enables proxy source support.
0	Disables proxy source support. **Default for all.**

ume_receiver_liveness_interval (context)

The maximum interval between delivery confirmations or keepalive messages send to the source. Expiration of this interval triggers another keepalive and an interval reset.

Scope:	context
Type:	**int**
Units:	milliseconds
Default value:	0 (disable; do not send keepalives)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMP 5.2.

ume_receiver_paced_persistence (receiver)

Specifies that the receiver is a Receiver-paced Persistence (RPP) receiver. If the repository has set `repository-allow-receiver-paced-persistence` to 0 (disable), setting this option to 1 creates a store registration error.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMP 5.3

Value	Description
1	Indicates that the receiver is a RPP receiver.
0	Indicates that the receiver is not a RPP receiver. **Default for all.**

ume_receiver_paced_persistence (source)

Specifies that the source is a Receiver-paced Persistence (RPP) source and may change certain topic repository options to values allowed by the repository. If the repository has set `repository-allow-receiver-paced-persistence` to 0 (disable), setting this option to 1 creates a store registration error.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMP 5.3

Value	Description
1	Indicates that source is a RPP source.
0	Indicates that source is not a RPP source. **Default for all.**

ume_recovery_sequence_number_info_function (receiver)

Callback function (and associated client data pointer) that is called when a receiver is about to complete registration from the stores in use by the source and the low sequence number is to be determined. The application has the ability to modify the sequence number to use if it desires. This callback is called directly in line and does not use the event queue. Therefore the callback function used should not block or it will block the context thread processing. A value of NULL for the callback turns off the callback being called.

Scope:	receiver
Type:	**lbm_ume_rcv_recovery_info_ex_func_t**
Default value:	NULL
When to Set:	Can only be set during object initialization.
Config File:	Cannot be set from an UM configuration file.

ume_registration_extended_function (receiver)

Callback function (and associated client data pointer) that is called when a receiver is about to attempt to register with a persistent store. The app must return the registration ID to request from the store or 0 if it will allow the store to allocate one. This function passes additional extended information, such as the store being used and a source client data pointer, etc. This callback is called directly in line and does not use the event queue. Therefore the callback function used should not block or it will block the context thread processing. A value of NULL for the callback turns off the callback being called.

Scope:	receiver
Type:	**lbm_ume_rcv_regid_ex_func_t**

Default value:	NULL
When to Set:	Can only be set during object initialization.
Config File:	Cannot be set from an UM configuration file.

ume_registration_function (receiver)

Callback function (and associated client data pointer) that is called when a receiver is about to attempt to register with a persistent store. The app must return the registration ID to request from the store or 0 if it will allow the store to allocate one. This callback is called directly in line and does not use the event queue. Therefore the callback function used should not block or it will block the context thread processing. A value of NULL for the callback turns off the callback being called. This setting is provided for compatibility. The " ume_registration_extended_function (receiver) " on page 196 setting should be used instead.

Scope:	receiver
Type:	**lbm_ume_rcv_regid_func_t**
Default value:	NULL
When to Set:	Can only be set during object initialization.
Config File:	Cannot be set from an UM configuration file.

ume_registration_interval (receiver)

The interval between registration attempts by the receiver to a persistent store in use by the source.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	500 (0.5 seconds)
When to Set:	Can only be set during object initialization.

ume_registration_interval (source)

The interval between registration attempts by the source. Before declaring Registration Complete, sources wait at least one full interval, unless all stores have registered.

When using the round-robin store behavior, this is the value between registration attempts with the various stores. In other words, attempt to register with primary, wait interval, attempt to register with secondary, wait interval, etc.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	500 (0.5 seconds)
When to Set:	Can only be set during object initialization.

ume_repository_ack_on_reception (source)

For topics with a `repository-type` of `disk` or `reduced-fd`, specifies that the stability acknowledgement should be sent upon message reception by the store instead of when the message has been written to disk. When using Receiver-paced Persistence, if the repository has set `stability-ack-on-reception` to 0 (disable), setting this option to 1 creates a store registration error. This option has no effect on Source-paced Persistence repositories.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMP 5.3

Value	Description
1	The repository sends a stability acknowledgement for a message as soon as it has received the message.
0	The repository sends a stability acknowledgement for a message once it has been written to disk. **Default for all.**

ume_repository_disk_file_size_limit (source)

For topics with a `repository-type` of `memory`, `disk` or `reduced-fd`, specifies the maximum amount of disk space used to store retained messages. Using the default value of 0 (zero) implements the repository's `repository-disk-file-size-limit` value. When not set to 0, UMP enforces a minimum value of 196992. When using Receiver-paced Persistence, you must set this option greater than 0 (zero) but less than or equal to the repository's `repository-disk-file-size-limit` value.

Scope:	source
Type:	**lbm_uint64_t**
Units:	bytes

Default value:	0 (disabled)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMP 5.3

ume_repository_size_limit (source)

For topics with a `repository-type` of `memory`, `disk` or `reduced-fd`, specifies the maximum number of message bytes retained (includes payload only). When using Receiver-paced Persistence, you must set this option greater than 0 (zero) but less than or equal to the repository's `repository-size-limit` value. For the `disk` or `reduced-fd` repository type, this value configures the size of the memory cache. Using the default value of 0 (zero) implements the repository's value for this option.

Scope:	source
Type:	**size_t**
Units:	bytes
Default value:	0 (disabled)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMP 5.3

ume_repository_size_threshold (source)

For topics with a `repository-type` of `memory`, `disk` or `reduced-fd`, specifies the minimum number of message bytes retained (includes payload only). When using Receiver-paced Persistence, you must set this option greater than 0 (zero) but less than or equal to the repository's `repository-size-threshold` value. For the `disk` or `reduced-fd` repository type, this value configures the size of the memory cache. Using the default value of 0 (zero) implements the repository's value for this option.

Scope:	source
Type:	**size_t**
Units:	bytes
Default value:	0 (disabled)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMP 5.3

ume_retention_intergroup_stability_behavior (source)

The behavior that the source will follow when determining, across store groups, both message stability and registration completion. A source cannot release a message until the message is stable. To be stable, a message must first be stable within the group and then stable between groups.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
any, **any-group**	**LBM_SRC_TOPIC_ATTR_UME_STABLE_BEHAVIOR_ANY**	Registration is complete when it is complete in any group. Messages are stable when they are stable in any group. **Default for all.**
all-active	**LBM_SRC_TOPIC_ATTR_UME_STABLE_BEHAVIOR_ALL_ACTIVE**	A group is active if it has at least a quorum of registered stores, or as determined by the `ume_retention_intragroup_stability_behavior` option. Registration is complete when it is complete in all active groups. At least one group must be active. Messages are stable when they are stable in all active groups.
majority	**LBM_SRC_TOPIC_ATTR_UME_STABLE_BEHAVIOR_MAJORITY**	Registration is complete when it is complete in a majority of groups. Messages are stable when they are stable in a majority of groups.
all, **all-groups**	**LBM_SRC_TOPIC_ATTR_UME_STABLE_BEHAVIOR_ALL**	Registration is complete when it is complete in all groups. Messages are stable when they are stable in all groups.

ume_retention_intragroup_stability_behavior (source)

The behavior that the source will follow when determining, within a store group, both message stability and group registration completion. A source cannot release a message until the message is stable. To be stable, a message must first be stable within the group and then stable between groups.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
quorum	**LBM_SRC_TOPIC_ATTR_UME_STABLE_BEHAVIOR_QUORUM**	Registration is complete for the group when a majority of the stores in the group are registered. A message is stable within the group when a majority of the stores have acknowledged the message as stable. **Default for all.**
all-active	**LBM_SRC_TOPIC_ATTR_UME_STABLE_BEHAVIOR_ALL_ACTIVE**	Registration is complete for the group when a majority of the stores in the group are registered. Stores registered with a source are active stores. A message is stable within the group when each active store in that group has acknowledged the message as stable.
all, **all-stores**	**LBM_SRC_TOPIC_ATTR_UME_STABLE_BEHAVIOR_ALL**	Registration is complete for the group when all stores in the group are registered. A message is stable within the group when all stores in the group are registered and have acknowledged the message as stable.

ume_retention_size_limit (source)

The release policy regarding aggregate size limit before messages are forced to be released. If the total number of bytes retained for the source is less than this amount, they may be released depending on other retention settings. If the total number of bytes exceeds this amount, then the message is forced to be released and a log message generated. This setting is provided for compatibility. The " retransmit_retention_size_limit (source) " on page 162 setting should be used instead.

Scope:	source
Type:	**size_t**
Units:	bytes

Default value:	25165824 (24 MB)
When to Set:	Can only be set during object initialization.

ume_retention_size_threshold (source)

The release policy regarding aggregate size threshold before messages are released. If the total number of bytes retained for the source is less than this amount, they will not be released. If the total number of bytes exceeds this amount, then the message may be released if no other release policy setting overrides the decision. A value of 0 indicates there is no size threshold set. This setting is provided for compatibility. The " retransmit_retention_size_threshold (source) " on page 162 setting should be used instead.

Scope:	source
Type:	**size_t**
Units:	bytes
Default value:	0 (no threshold)
When to Set:	Can only be set during object initialization.

ume_retention_unique_confirmations (source)

The release policy regarding the number of confirmations from different receivers required before the source can release a message. This option enhances, but does not supersede, message stability notification from the store(s). If the number of unique confirmations for a message is less than this amount, the message will not be released. If the number of unique confirmations for a message exceeds or equals this amount, then the message may be released if no other release policy setting overrides the decision. A value of 0 indicates there is no unique number of confirmations required for reclamation.

Scope:	source
Type:	**size_t**
Units:	number of confirmations
Default value:	0 (none required)
When to Set:	Can only be set during object initialization.

ume_retransmit_request_generation_interval (receiver)

The maximum interval between when a retransmission request is first sent and when it is given up on and loss is reported. This setting is provided for compatibility. The “ retransmit_request_generation_interval (receiver) ” on page 227 setting should be used instead.

Scope:	receiver
Type:	**unsigned long int**
Units:	milliseconds
Default value:	10000 (10 seconds)
When to Set:	Can only be set during object initialization.

ume_retransmit_request_interval (receiver)

The interval between retransmission request messages to the persistent store or to the source. This setting is provided for compatibility. The “ retransmit_request_interval (receiver) ” on page 160 setting should be used instead.

Scope:	receiver
Type:	**unsigned long int**
Units:	milliseconds
Default value:	500 (0.5 seconds)
When to Set:	Can only be set during object initialization.

ume_retransmit_request_maximum (receiver)

The maximum number of messages to request back from the current latest message when late joining a topic or when registering with a UMP store. A value of 0 indicates no maximum. This setting is provided for compatibility. The “ retransmit_request_maximum (receiver) ” on page 161 setting should be used instead.

Scope:	receiver
Type:	**unsigned long int**
Units:	messages
Default value:	0
When to Set:	Can only be set during object initialization.

ume_retransmit_request_outstanding_maximum (receiver)

The maximum number of messages to request at a single time from the store or source. A value of 0 indicates no maximum. This setting is provided for compatibility. The " retransmit_request_outstanding_maximum (receiver) " on page 161 setting should be used instead.

Scope:	receiver
Type:	**unsigned long int**
Units:	messages
Default value:	200
When to Set:	Can only be set during object initialization.

ume_session_id (context)

Specifies the default Session ID to use for sources and receivers within a context. A value of 0 (zero) indicates no Session ID is to be set. See also *Managing RegIDs with Session IDs* in the *Ultra Messaging Guide for Persistence*. Valid formats for session IDs are as follows: A hexadecimal string with a maximum value of FFFFFFFFFFFFFFFE, prefixed with '0x'. An octal string with a maximum value of 1777777777777777777776 prefixed with '0'. A decimal string with a maximum value of 18446744073709551614. Prior to LBM 5.2.2, all UME session IDs were interpreted as hexadecimal, and did not accept the '0x' prefix. If upgrading from an earlier version to LBM 5.2.2 or later, prepend '0x' to the original setting to use the originally assigned session ID.

Scope:	context
Type:	**lbm_uint64_t**
Default value:	0 (zero)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2/UME 3.2

ume_session_id (receiver)

Specifies the Session ID to use for a receiver. A value of 0 (zero) indicates the context ume_session_id will be used. See also *Managing RegIDs with Session IDs* in the *Ultra Messaging Guide for Persistence*. Valid formats for session IDs are as follows: A hexadecimal string with a maximum value of FFFFFFFFFFFFFFFE, prefixed with '0x'. An octal string with a maximum value of 1777777777777777777776 prefixed with '0'. A decimal string with a maximum value of 18446744073709551614. Prior to LBM 5.2.2, all UME session IDs were interpreted as hexadecimal, and did not accept the '0x' prefix. If upgrading from an earlier version to LBM 5.2.2 or later, prepend '0x' to the original setting to use the originally assigned session ID.

Scope:	receiver
Type:	**lbm_uint64_t**

Default value:	0 (zero)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2/UME 3.2

ume_session_id (source)

Specifies the Session ID to use for a source. A value of 0 (zero) indicates the context ume_session_id will be used. See also *Managing RegIDs with Session IDs* in the *Ultra Messaging Guide for Persistence*. Valid formats for session IDs are as follows: A hexadecimal string with a maximum value of FFFFFFFFFFFFFFFE, prefixed with '0x'. An octal string with a maximum value of 1777777777777777777776 prefixed with '0'. A decimal string with a maximum value of 18446744073709551614. Prior to LBM 5.2.2, all UME session IDs were interpreted as hexadecimal, and did not accept the '0x' prefix. If upgrading from an earlier version to LBM 5.2.2 or later, prepend '0x' to the original setting to use the originally assigned session ID.

Scope:	source
Type:	**lbm_uint64_t**
Default value:	0 (zero)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2/UME 3.2

ume_source_liveness_timeout (context)

The expected maximum interval between keepalive or delivery confirmation messages from a receiver. If neither are received within the interval, the source declares the receiver "dead".

Scope:	context
Type:	**int**
Units:	milliseconds
Default value:	0 (disable; do not track receivers)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMP 5.2.

ume_sri_flush_sri_request_response (source)

This option determines if a source flushes the Implicit Batching buffer after it sends a Source Registration Information (SRI) record in response to a SRI request from a receiver. Flushing this buffer places the SRI record immediately on the transport.

Scope:	source
Type:	**lbm_ulong_t**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMP 6.0

Value	Description
1	The source places a SRI record in the Implicit Batching buffer and then flushes the buffer.
0	The source places a SRI record in the Implicit Batching buffer and lets normal batch scheduling determine when to place the SRI on the transport. **Default for all.**

ume_sri_immediate_sri_request_response (source)

This option determines how a source responds to a receiver's request for a Source Registration Information (SRI) record. The default setting for this option is the fastest response to a receiver's SRI request.

Scope:	source
Type:	**lbm_ulong_t**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMP 6.0

Value	Description
1	Indicates that the source sends an SRI record and also flushes the implicit batching buffer to immediately put the SRI record on the transport. **Default for all.**
0	Indicates that the source waits for the period of time defined by "ume_sri_request_response_latency (source)" on page 208 before sending an SRI record.

ume_sri_inter_sri_interval (source)

The interval between the sending of SRI packets by a source. The default value results in the source sending 10 SRI packets every second.

Scope:	source
Type:	**lbm_ulong_t**

Units:	milliseconds
Default value:	100 (0.1 seconds)
When to Set:	Can only be set during object initialization.
Version	This option was implemented in UMP 6.0

ume_sri_max_number_of_sri_per_update (source)

The maximum number of SRI packets sent by a source after it has re-registered with a store.

Scope:	source
Type:	**lbm_ulong_t**
Default value:	10
When to Set:	Can only be set during object initialization.
Version	This option was implemented in UMP 6.0

ume_sri_request_interval (receiver)

The interval at which the receiver requests a Store Information Record (SRI) from the source. Controlling these requests helps reduce receiver start-up traffic on your network.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	1000 (1 second)
When to Set:	Can only be set during object initialization.
Version	This option was implemented in UMP 6.0

ume_sri_request_maximum (receiver)

The maximum number of requests the receiver issues for a Store Information Record (SRI) from the source. If the receiver has not received an SRI after this number of requests, it stops requesting.

Scope:	source
Type:	**lbm_ulong_t**
Default value:	60

When to Set:	Can only be set during object initialization.
Version	This option was implemented in UMP 6.0

ume_sri_request_response_latency (source)

The interval a source waits before sending an SRI packet in response to a SRI request from a receiver. At the expiration of this interval, the SRI record may also be slightly delayed by normal batch scheduling unless "ume_sri_flush_sri_request_response (source)" on page 206 is set to 1.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	100 (0.1 seconds)
When to Set:	Can only be set during object initialization.
Version	This option was implemented in UMP 6.0

ume_state_lifetime (receiver)

Establishes the period of time from a receiver's last activity to the deletion of the receiver's state and cache by the store. You can also configure a `receiver-state-lifetime` for the receiver's topic on the store. The store uses whichever is shorter. The default value of 0 (zero) disables this option. See also *Proxy Sources* in the *Ultra Messaging Guide for Persistence*.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	0 (zero)
When to Set:	Can only be set during object initialization.

ume_state_lifetime (source)

Establishes the period of time from a source's last activity to the deletion of the source's state and cache by the store, regardless of whether a proxy source has been created or not. You can also configure a `source-state-lifetime` for the source's topic on the store. The store uses whichever is shorter. The default value of 0 (zero) disables this option. See also *Proxy Sources* in the *Ultra Messaging Guide for Persistence*.

Scope:	source
Type:	**lbm_ulong_t**

Units:	milliseconds
Default value:	0 (zero)
When to Set:	Can only be set during object initialization.

ume_store (source)

Add a store specification to the list of stores specified for the source. Unlike most other UMP settings, every time this setting is called, it adds another store specification to the list and does NOT overwrite previous specifications.

Each entry contains the IP address, TCP port, registration ID, and group index for the store. For the configuration file as well as string versions of this option, format the string value as `DomainID:IP:port:RegID:GroupIDX` where `DomainID` is the store's UM domain ID, `IP` is the stores IP address, `port` is the TCP port for the store, `RegID` is the registration ID that the source desires to use, and `GroupIDX` is the group index that the store belongs to. The DomainID, RegID and GroupIDX pieces may be left off the string if desired. If so, UMP assumes the value of 0 for them.

Because each entry adds a new store specification and does not overwrite previous values, an entry or string with the IP address of 0.0.0.0 and TCP port of 0 causes the removal of all previous store specifications. A single store specification means the source uses persistence. If no stores are specified, then persistence will not be provided for the source.

Scope:	source
Type:	**lbm_ume_store_entry_t**
When to Set:	Can only be set during object initialization.

ume_store_activity_timeout (source)

The timeout value used to indicate when a store is unresponsive. The store must not be active within this interval to be considered unresponsive. This value must be much larger than the check interval.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	3000 (3 seconds)
When to Set:	Can only be set during object initialization.

ume_store_behavior (source)

The behavior that the source follows for handling store failures.

Scope:	source
Type:	**int**
When to Set:	Can only be set during object initialization.

String value	Integer value	Description
rr, **round-robin**	**LBM_SRC_TOPIC_ATTR_UME_STORE_BEHAVIOR_RR**	The source uses a single store at a time, and when a store is unresponsive due to failure or disconnect, the next store in the list will be used. This continues in a round-robin fashion until an available store is found that is available. **Default for all. This selection is deprecated.**
qc, **quorum-consensus**	**LBM_SRC_TOPIC_ATTR_UME_STORE_BEHAVIOR_QC**	The source uses multiple stores at the same time based on store and store group configuration.

ume_store_check_interval (source)

The interval between activity checks of the current store. This interval also governs how often a source checks outstanding unstabilized messages to see if they have reached the configured " ume_message_stability_timeout (source)" on page 194 value yet.

Scope:	source
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	500 (0.5 seconds)
When to Set:	Can only be set during object initialization.

ume_store_group (source)

Add a store group specification to the list of store groups specified for the source. Unlike other UMP settings, every time this setting is called, it adds another store group specification to the list and does NOT overwrite previous specifications. Each entry contains the group index and group size for the group. For the configuration file as well as string versions of setting this option, the string value is formatted as `GroupIDX:GroupSZ` where `GroupIDX` is the index of the group and `GroupSZ` is the size of the group. Because each entry adds a new store specification and does not overwrite previous values, an entry or string with the group index of 0 and group size of 0 will cause all previous store group specifications to be removed.

Note: When setting this option multiple times, you must set this option in group-index order, from lowest to highest. In other words, do not set this option for a group index lower in value than any previously set group index value.

Scope:	source
Type:	**lbm_ume_store_group_entry_t**
When to Set:	Can only be set during object initialization.

ume_store_name (source)

Add a store specification to the list of stores specified for the source. Unlike other UMP settings, every time this setting is called, it adds another store specification to the list and does NOT overwrite previous specifications. Each entry contains the store name, registration ID, and group index for the store. For the configuration file as well as string versions of setting this option, the string value is formatted as `name:RegID:GroupIDX` where `name` is the name of the store configured with the store attribute, `context-name` in the `umestored` XML configuration file, `RegID` is the registration ID that the source desires to use, and `GroupIDX` is the group index that the store belongs to. The RegID and GroupIDX pieces may be left off the string if desired. If so, then the value of 0 is assumed for them. Store names are restricted to 128 characters in length, and may contain only alphanumeric characters, hyphens, and underscores.

Scope:	source
Type:	**lbm_ume_store_name_entry_t**
When to Set:	Can only be set during object initialization.

ume_use_ack_batching (receiver)

Specifies whether or not UMP allows the batching of consumption acknowledgments sent to the store(s). If enabled, UMP checks for contiguous sequence numbered messages at the " ume_ack_batching_interval (context) " on page 186. See also *Batching Acknowledgments* in the *Ultra Messaging Guide for Persistence*.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMS 5.0, UMP 5.0, UMQ 5.0.

Value	Description
1	Indicates that UMP can acknowledge the consumption of a batch of messages.
0	Indicates that UMP acknowledges the consumption of individual messages by the receiver. **Default for all.**

ume_use_late_join (receiver)

Flag indicating if the receiver should participate in late join operation or not. This is a compatibility setting. The " use_late_join (receiver) " on page 163 setting should be used instead.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	The receiver will participate in using late join if requested to by the source. **Default for all.**
0	The receiver will not participate in using late join even if requested to by the source.

ume_use_store (receiver)

Flag indicating if the receiver should participate in using a persistent store or not.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	The receiver will participate in using a persistent store if requested to by the source. **Default for all.**
0	The receiver will not participate in using a persistent store even if requested to by the source.

ume_user_receiver_registration_id (context)

32-bit value that is used as a user set identifier to be included as the receiver registration ID in acknowledgements send by any receivers in the context to sources as confirmed delivery notifications. The value is not interpreted by UMP in any way and has no relation to registration IDs used by the receiver. A value of 0 indicates no user set value is in use and should not be sent with acknowledgements

Scope:	context
Type:	**lbm_uint_t**
Units:	identifier

Default value:	0 (no user set value in use)
When to Set:	Can only be set during object initialization.

ume_write_delay (source)

For topics with a `repository-type` of `disk`, specifies the delay in milliseconds before the repository persists a message to disk. When using Receiver-paced Persistence, you must set this option greater than 0 (zero) but less than or equal to the repository's `write-delay` value.

Scope:	source
Type:	**lbm_uint32_t**
Units:	milliseconds
Default value:	0 (disabled)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMP 5.3

Hot Failover Operation Options

Hot Failover (HF) allows your applications to build in sender redundancy. See *Hot Failover* in the *Ultra Messaging Concepts Guide* for a discussion of using Hot Failover within a single receiver context or across multiple receiver contexts.

delivery_control_loss_check_interval (hfx)

The interval between periodic forced loss checks. This option defaults to 0, indicating that loss checks should only be made when a new message arrives.

Scope:	hfx
Type:	**lbm_ulong_t**
Units:	msec
Default value:	0 (no periodic loss checks)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2

delivery_control_max_delay (hfx)

The minimum interval that must expire before the HFX Receiver declares a message unrecoverable and delivers an unrecoverable loss message the application. By default, the HFX Receiver only checks loss when it receives new messages. To enable periodic loss checks, set the " delivery_control_loss_check_interval (hfx) " on page 213 option.

Scope:	hfx
Type:	**lbm_ulong_t**
Units:	msec
Default value:	10000 (10 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2

delivery_control_maximum_burst_loss (hfx)

Specifies the largest permissible gap between the next expected message and the most recently received message. When the difference in sequence numbers between the most recently received message and the next expected message exceeds this amount, the HFX Receiver delivers a burst loss notification. The HFX Receiver discards any messages currently pending delivery. Normal delivery resumes with the most recently received message.

Scope:	hfx
Type:	**lbm_uint_t**
Units:	number of messages
Default value:	512
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2

delivery_control_maximum_total_map_entries (hfx)

The maximum number of map entries for the HFX order and loss maps. This is a soft limit. When the sum of the number of loss records and the number of messages held for ordering (messages that will be delivered once all prior messages have been delivered) is greater than this value, the oldest consecutive sequence of loss records will be declared lost immediately to reduce the number of outstanding map entries. A value of 0 indicates that the map should be allowed to grow without bound.

Scope:	hfx
Type:	**size_t**
Units:	map entries

Default value:	200000
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2

duplicate_delivery (hfx)

Flag indicating whether duplicate messages should be discarded or simply marked as duplicates. Setting this to 1 overrides the " hf_duplicate_delivery (receiver) " on page 215 setting on all underlying HFX Receivers.

Scope:	hfx
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2

Value	Description
1	The HFX delivers duplicate messages.
0	The HFX does not deliver duplicate messages. **Default for all.**

hf_duplicate_delivery (receiver)

Flag indicating if the Hot Failover receiver delivers duplicate messages or not.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.

Value	Description
1	The Hot Failover receiver delivers duplicate messages.
0	The Hot Failover receiver does not deliver duplicate messages. **Default for all.**

hf_optional_messages (receiver)

Indicates if a Hot Failover receiver can receive optional messages. See also *Hot Failover Optional Messages* in the *Ultra Messaging Concepts Guide*.

Scope:	receiver
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2.5/UME 3.2.5/UMQ 2.1.5

Value	Description
1	Hot Failover receivers can receive optional messages. **Default for all.**
0	Hot Failover receivers do not receive optional messages.

hf_receiver (wildcard_receiver)

Specifies whether to create hot failover receivers for each topic that maps to the wildcard receiver pattern.

Scope:	wildcard_receiver
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UMS 5.2.2

Value	Description
1	Create hot failover receivers for each matched topic.
0	Normal wildcard receiver operation. Hot failover sequence numbers are ignored. **Default for all.**

ordered_delivery (hfx)

Flag indicating if the HFX Receiver orders messages before delivery.

Scope:	hfx
Type:	**int**

When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.2

String value	Integer value	Description
"1" (Integer value as a string.)	**1**	The HFX Receiver delivers messages in order. **Default for all.**
"-1" (Integer value as a string.)	**-1**	The HFX Receiver delivers messages as soon as they are received. In the case of fragmented messages, as soon as all fragments have been received and reassembled.

Automatic Monitoring Options

The Monitoring Options below apply to a given UMS context. You can override the default values of these options and apply monitoring option values to all UMS contexts (transports and event queues) with the following environment variables.

- LBM_MONITOR_INTERVAL
- LBM_MONITOR_TRANSPORT
- LBM_MONITOR_TRANSPORT_OPTS
- LBM_MONITOR_APPID

These variables will not override any Monitoring Options you specifically set. The environment variables only override Monitoring Options default values.

If you do not specify any monitoring options either in an UMS configuration file or via `lbm_context_attr_setopt()` calls, no monitoring will occur. However, if you then set the LBM_MONITOR_INTERVAL environment variable to 5, you will turn on automatic monitoring for every UMS context your application creates at 5 second intervals. If you then set `monitor_interval` to 10 for a particular context, all transport sessions in that context will be monitored every 10 seconds.

For XML configuration files, you can configure an automatic monitoring context by setting `<context>` attribute `name=29west_statistics_context`.

See also *Automatic Monitoring* in the *Ultra Messaging Operations Guide* for more information about this feature.

monitor_appid (context)

An application ID string used by automatic monitoring to identify the application generating the statistics.

Scope:	context
Type:	**string**

When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.4/UME 2.1.

monitor_appid (event_queue)

An application ID string used by automatic monitoring to identify the application generating the statistics.

Scope:	event_queue
Type:	**string**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.4/UME 2.1.

monitor_interval (context)

Interval at which automatic monitoring retrieves the statistics for all transport sessions on a context. Setting this option to zero (the default) disables the automatic monitoring of a context's transport sessions.

Scope:	context
Type:	**lbm_ulong_t**
Units:	seconds
Default value:	0
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.4/UME 2.1.

monitor_interval (event_queue)

Interval at which automatic monitoring retrieves the statistics for an event queue. Setting this option to zero (the default) disables the automatic monitoring of an event queue. When monitoring Event Queue statistics you must enable the Event Queue UM Configuration Options, `queue_age_enabled` , `queue_count_enabled` and `queue_service_time_enabled` . UM disables these options by default, which produces no event queue statistics.

Scope:	event_queue
Type:	**lbm_ulong_t**
Units:	seconds
Default value:	0

When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.4/UME 2.1.

monitor_interval (receiver)

Interval at which automatic monitoring retrieves the topic interest information for all receivers using a UM configuration file with this option set to a non-zero value. Topic interest information contains source and topic information if the receiver has joined the source transport session. If the topic interest information is blank, the receiver has not joined a source transport session. UM System Monitoring uses this information to monitor the number of subscribed topics. Setting this option to zero (the default) disables the automatic monitoring of receiver interest.

Scope:	context
Type:	**lbm_ulong_t**
Units:	seconds
Default value:	0
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.5.

monitor_interval (wildcard_receiver)

Interval at which automatic monitoring retrieves the topic interest information for all receivers interested in topics that match the wildcard receiver pattern. Topic interest information contains source and topic information if the receiver has joined the source transport session. If the topic interest information is blank, the receiver has not joined a source transport session. UM System Monitoring uses this information to monitor the number of subscribed topics. Setting this option to zero (the default) disables the automatic monitoring of a wildcard receiver interest.

Scope:	context
Type:	**lbm_ulong_t**
Units:	seconds
Default value:	0
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in UM 6.5.

monitor_transport (context)

The LBMMON transport module to be used for automatic monitoring.

Scope:	context
Type:	**string**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.4/UME 2.1.

String value	Integer value	Description
lbm	**LBM_CTX_ATTR_MON_TRANSPORT_LBM**	Use the LBMMON lbm transport module. **Default for all.**
lbmsnmp	**LBM_CTX_ATTR_MON_TRANSPORT_LBMSNMP**	Use the LBMMON lbmsnmp transport module. This value is required if you use the Ultra Messaging SNMP Agent.

monitor_transport (event_queue)

The LBMMON transport module to be used for automatic monitoring.

Scope:	event_queue
Type:	**string**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.4/UME 2.1.

String value	Integer value	Description
lbm	**LBM_CTX_ATTR_MON_TRANSPORT_LBM**	Use the LBMMON lbm transport module. **Default for all.**
lbmsnmp	**LBM_CTX_ATTR_MON_TRANSPORT_LBMSNMP**	Use the LBMMON lbmsnmp transport module. This value is required if you use the Ultra Messaging SNMP Agent.

monitor_transport_opts (context)

An option string to be passed to the LBMMON transport module for automatic monitoring. See *The UM Transport Module* in the *Ultra Messaging Concepts Guide* for more information about Transport Options. (Options for the `lbm` transport module and the `lbmsnmp` transport module are identical.)

Scope:	context
Type:	**string**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.4/UME 2.1.

monitor_transport_opts (event_queue)

An option string to be passed to the LBMMON transport module for automatic monitoring. See *The UM Transport Module* in the *Ultra Messaging Concepts Guide* for more information about Transport Options. (Options for the `lbm` transport module and the `lbmsnmp` transport module are identical.)

Scope:	event_queue
Type:	**string**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.4/UME 2.1.

Deprecated Options

dbl_lbtrm_acceleration (context)

Flag indicating if DBL acceleration is enabled for LBT-RM transports.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0.

Value	Description
1	DBL acceleration is enabled for LBT-RM.
0	DBL acceleration is not enabled for LBT-RM. **Default for all.**

dbl_lbtru_acceleration (context)

Flag indicating if DBL acceleration is enabled for LBT-RU transports.

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0.

Value	Description
1	DBL acceleration is enabled for LBT-RU.
0	DBL acceleration is not enabled for LBT-RU. **Default for all.**

dbl_mim_acceleration (context)

Flag indicating if DBL acceleration is enabled for multicast immediate messaging (MIM).

Scope:	context
Type:	**int**
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0.

Value	Description
1	DBL acceleration is enabled for MIM.
0	DBL acceleration is not enabled for MIM. **Default for all.**

dbl_resolver_acceleration (context)

Flag indicating if DBL acceleration is enabled for topic resolution.

Scope:	context
Type:	**int**

When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 4.0.

Value	Description
1	DBL acceleration is enabled for topic resolution.
0	DBL acceleration is not enabled for topic resolution. **Default for all.**

otr_request_duration (receiver)

The length of time a receiver continues to send OTR lost-message requests before giving up.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	20000 (20 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in UM 6.0

resolver_active_source_interval (context)

Interval between sending Topic Resolution advertisements for active sources. A value of 0 indicates that periodic advertisements should not be sent (sources will still respond to queries). When set to 0, the `resolver_active_threshold` should typically also be set to 0. See also "Disabling Aspects of Topic Resolution " on page 39.

Note: Although this option is eligible to be set during operation, two considerations exist.

- If this option is disabled at initialization (set to 0), you cannot re-set the option during operation.
- Disabling this option by setting it to 0 (zero) during operation prevents you from re-setting the option a second time during operation.

Scope:	context
Type:	**unsigned long int**
Units:	milliseconds
Default value:	1000 (1 second)
When to Set:	May be set during operation.
Version:	This option was deprecated in LBM 4.0

resolver_active_threshold (context)

Number of seconds since the last application message was sent to a source that causes that source to be marked inactive. Inactive sources are not advertised periodically (but will continue to respond to queries). A value of 0 indicates that sources will advertise periodically regardless of how often the application sends messages. Note that for publishers with large numbers of sources, this can increase the topic resolution traffic load. However, also note that this option SHOULD be set to 0 if periodic advertisements are disabled (by setting `resolver_active_source_interval` to 0). See also "Disabling Aspects of Topic Resolution " on page 39 and "Interrelated Configuration Options" on page 42.

Scope:	context
Type:	**unsigned long int**
Units:	seconds
Default value:	60
When to Set:	May be set during operation.
Version:	This option was deprecated in LBM 4.0

resolver_context_advertisement_interval (context)

Interval between context advertisements. Setting this option to 0 disables context advertisements, though UM Router and other functionality depends upon context advertisements, so a value of 0 is not generally recommended.

Scope:	context
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10000 (10 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in UM 6.0.

resolver_maximum_advertisements (context)

Maximum number of topics that will be advertised per active source interval. A value of 0 means to advertise all topics.

Scope:	context
Type:	**unsigned long int**
Units:	Number of topics
Default value:	0 (all topics)

When to Set:	May be set during operation.
Version:	This option was deprecated in LBM 4.0

resolver_query_interval (context)

Interval between query transmissions for receivers attempting Topic Resolution. A value of 0 indicates queries should not be sent. See also *Disabling Aspects of Topic Resolution*.

Note: Although this option is eligible to be set during operation, two considerations exist.

- If this option is disabled at initialization (set to 0), you cannot re-set the option during operation.
- Disabling this option by setting it to 0 (zero) during operation prevents you from re-setting the option a second time during operation.

Scope:	context
Type:	**unsigned long int**
Units:	milliseconds
Default value:	100 (0.1 seconds)
When to Set:	May be set during operation.
Version:	This option was deprecated in LBM 4.0

resolver_maximum_queries (context)

Maximum number of topics that will be queried for per query interval. A value of 0 means to query for all topics that do not have at least one source.

Scope:	context
Type:	**unsigned long int**
Units:	Number of topics
Default value:	0 (all topics with no source)
When to Set:	May be set during operation.
Version:	This option was deprecated in LBM 4.0

resolver_query_max_interval (wildcard_receiver)

This sets the maximum interval between wildcard queries in topic resolution (when used). Only PCRE and regex pattern types can use wildcard queries. A value of 0 indicates wildcard queries should not be sent. UM currently queries a maximum of 250 unique wildcard patterns (receivers).

Note: Although this option is eligible to be set during operation, two considerations exist.

- If this option is disabled at initialization (set to 0), you cannot re-set the option during operation.
- Disabling this option by setting it to 0 (zero) during operation prevents you from re-setting the option a second time during operation.

Scope:	wildcard_receiver
Type:	**unsigned long int**
Units:	milliseconds
Default value:	0 (do not query)
When to Set:	May be set during operation.
Version:	This option was deprecated in LBM 4.0

resolver_unicast_address (context)

The IP address to send unicast topic resolution messages to. If set to 0.0.0.0 (**INADDR_ANY**), then topic resolution uses multicast (the default). If set to anything else, then topic resolution messages go to the IP address specified.

Scope:	context
Type:	**struct in_addr**
Default value:	0.0.0.0 (**INADDR_ANY**)
When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in UMS 5.0. See resolver_unicast_daemon instead.

resolver_unicast_destination_port (context)

The UDP port to send unicast topic resolution messages to. This is the UDP port used by the UM resolution daemon (lbmrd).

Scope:	context
Type:	**lbm_uint16_t**
Default value:	15380
Byte order:	Network
When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in UMS 5.0. See resolver_unicast_daemon instead.

resolver_unicast_port (context)

The local UDP port used for unicast topic resolution messages. The UM resolution daemon (lbmrd) will send unicast topic resolution messages to this UDP port. A value of 0 indicates that UM should pick an open port in the range (“ resolver_unicast_port_low (context) ” on page 93, “ resolver_unicast_port_high (context) ” on page 93).

Scope:	context
Type:	**lbm_uint16_t**
Default value:	0 (pick open port)
Byte order:	Network
When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in UMS 5.0. See resolver_unicast_daemon instead.

retransmit_message_map_tablesz (source)

The size of the hash table that the source uses to store messages for the retention policy in effect. A larger table means more messages can be stored more efficiently, but takes up more memory. A smaller table uses less memory, but costs more CPU time as more messages are retained. See *Configuring Late Join for Large Numbers of Messages* in the *Ultra Messaging Concepts Guide* for additional information about this option.

Scope:	source
Type:	**size_t**
Default value:	131
When to Set:	Can only be set during object initialization.
Version:	This option has been deprecated.

retransmit_request_generation_interval (receiver)

The maximum interval between when a receiver first sends a retransmission request and when the receiver stops and reports loss on the remaining RXs not received. See *Configuring Late Join for Large Numbers of Messages* in the *Ultra Messaging Concepts Guide* for additional information about this option.

Scope:	receiver
Type:	**lbm_ulong_t**
Units:	milliseconds
Default value:	10000 (10 seconds)

When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in UM 6.0

transport_datagram_max_size (context)

The maximum datagram size that can be generated by UM. The default value is 8192, the minimum is 400 bytes, and the maximum is 65535.

Scope:	context
Type:	**unsigned int**
Units:	bytes
Default value:	8192
When to Set:	Can only be set during object initialization.
Version:	This option was implemented in LBM 3.3.5/UME 2.0.3.
Version:	This option was deprecated in LBM 4.1

transport_lbtipc_acknowledgement_interval (receiver)

Period of time between acknowledgement (keepalive) messages sent from the receiver to the IPC source. See also " transport_lbtipc_client_activity_timeout (source) " on page 228. See also "Disabling Aspects of Topic Resolution " on page 39 and "Interrelated Configuration Options" on page 42.

Scope:	receiver
Type:	**unsigned long int**
Units:	milliseconds
Default value:	500 (0.5 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in LBM 4.0

transport_lbtipc_client_activity_timeout (source)

The maximum period of inactivity (lack of acknowledgement keepalive messages) from a receiver before the source deletes the receiver from its active receiver table. The IPC source signals all receivers in it's active receiver's table when it writes new data to the shared memory area. See also

“ transport_lbtipc_acknowledgement_interval (receiver) ” on page 228. See also “Disabling Aspects of Topic Resolution ” on page 39 and “Interrelated Configuration Options” on page 42.

Scope:	source
Type:	**unsigned long int**
Units:	milliseconds
Default value:	10,000 (10 seconds)
When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in LBM 4.0

ume_message_map_tablesz (source)

The size of the hash table that the source uses to store messages for the retention policy in effect. A larger table means more messages can be stored more efficiently, but takes up more memory. A smaller table uses less memory, but costs more CPU time as more messages are retained. This setting no longer has any effect.

Scope:	source
Type:	**size_t**
Default value:	131
When to Set:	Can only be set during object initialization.
Version:	This option has been deprecated.

ume_primary_store_address (source)

IPv4 address of the persistent store to be used as the primary store. A value of 0.0.0.0 (or **INADDR_ANY**) indicates no store is set as the primary. In other words, persistence is not enabled for the source. This setting is deprecated. Its use is not recommended except by legacy systems. Please use the ume_store option instead.

Scope:	source
Type:	**struct in_addr**
Default value:	0.0.0.0 (**INADDR_ANY**)
When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in UME 2.0

ume_primary_store_port (source)

TCP port of the primary persistent store. This setting is deprecated. Its use is not recommended except by legacy systems. Please use the ume_store option instead.

Scope:	source
Type:	**lbm_uint16_t**
Default value:	14567
Byte order:	Network
When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in UME 2.0

ume_registration_id (source)

32-bit value that is used by a persistent store to identify a source. If a source desires to identify itself as a previously known source (after a crash or shutdown), it should set the ID to the value it was using before. A value of 0 indicates the source will allow the persistent store to assign an ID. This setting is deprecated. Its use is not recommended except by legacy systems. Please use the ume_store option instead.

Scope:	source
Type:	**lbm_uint_t**
Units:	identifier
Default value:	0 (allow persistent store to assign ID)
When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in UME 2.0

ume_secondary_store_address (source)

IPv4 address of the persistent store to be used as the secondary store. A value of 0.0.0.0 (or **INADDR_ANY**) indicates no store is set as the secondary. This setting is deprecated. Its use is not recommended except by legacy systems. Please use the ume_store option instead.

Scope:	source
Type:	**struct in_addr**
Default value:	0.0.0.0 (**INADDR_ANY**)
When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in UME 2.0

ume_secondary_store_port (source)

TCP port of the secondary persistent store. This setting is deprecated. Its use is not recommended except by legacy systems. Please use the ume_store option instead.

Scope:	source
Type:	**lbm_uint16_t**
Default value:	14567
Byte order:	Network
When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in UME 2.0

ume_tertiary_store_address (source)

IPv4 address of the persistent store to be used as the tertiary store. A value of 0.0.0.0 (or **INADDR_ANY**) indicates no store is set as the tertiary. This setting is deprecated. Its use is not recommended except by legacy systems. Please use the ume_store option instead.

Scope:	source
Type:	**struct in_addr**
Default value:	0.0.0.0 (**INADDR_ANY**)
When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in UME 2.0

ume_tertiary_store_port (source)

TCP port of the tertiary persistent store. This setting is deprecated. Its use is not recommended except by legacy systems. Please use the ume_store option instead.

Scope:	source
Type:	**lbm_uint16_t**
Default value:	14567
Byte order:	Network
When to Set:	Can only be set during object initialization.
Version:	This option was deprecated in UME 2.0

UMS Port Values

This section lists the default port values use by UMS.

UMS UDP Port Values

Table 5. Default UMS UDP Port Values

Configuration Option	Scope	Default Value
mim_destination_port	context	14401
mim_incoming_destination_port	context	14401
mim_outgoing_destination_port	context	14401
resolver_multicast_incoming_port	context	12965
resolver_multicast_outgoing_port	context	12965
resolver_multicast_port	context	12965
resolver_unicast_destination_port	context	15380
resolver_unicast_port	context	0 (pick open port)
resolver_unicast_port_high	context	14406
resolver_unicast_port_low	context	14402
transport_lbtrm_destination_port	source	14400
transport_lbtrm_source_port_high	context	14399
transport_lbtrm_source_port_low	context	14390
transport_lbtru_maximum_ports	context	5
transport_lbtru_port	source	0 (use open port)
transport_lbtru_port_high	context	14389
transport_lbtru_port_high	receiver	14379
transport_lbtru_port_low	context	14380
transport_lbtru_port_low	receiver	14360

UMS TCP Port Values

Table 6. Default UMS TCP Port Values

Configuration Option	Scope	Default Value
request_tcp_port	context	0 (use open port)
request_tcp_port_high	context	14395
request_tcp_port_low	context	14391
transport_tcp_maximum_ports	context	10
transport_tcp_port	source	0 (pick open port)
transport_tcp_port_high	context	14390
transport_tcp_port_low	context	14371
ume_primary_store_port	source	14567
ume_secondary_store_port	source	14567
ume_tertiary_store_port	source	14567

UMS Multicast Group Values

This section lists the default multicast group values use by UMS.

Table 7. Default UMS Multicast Group Values

Configuration Option	Scope	Default Value
mim_address	context	224.10.10.21
mim_incoming_address	context	224.10.10.21
mim_outgoing_address	context	224.10.10.21
resolver_multicast_address	context	224.9.10.11
resolver_multicast_incoming_address	context	224.9.10.11
resolver_multicast_outgoing_address	context	224.9.10.11
transport_lbtrm_multicast_address	source	0.0.0.0 (INADDR_ANY)
transport_lbtrm_multicast_address_high	context	224.10.10.14
transport_lbtrm_multicast_address_low	context	224.10.10.10

UMS Timer Interval Values

This section lists the default timer interval values use by UMS. All values are in milliseconds.

Table 8. Default UMS Timer Interval Values

Configuration Option	Scope	Default Value
delivery_control_loss_check_interval	receiver	0 (disabled)
implicit_batching_interval	source	200 (0.2 seconds)
mim_activity_timeout	context	60000 (60 seconds)
mim_delivery_control_loss_check_interval	context	0 (disabled)
mim_ignore_interval	context	500 (0.5 seconds)
mim_implicit_batching_interval	context	200 (0.2 seconds)
mim_nak_backoff_interval	context	200 (0.2 seconds)
mim_nak_generation_interval	context	10000 (10 seconds)
mim_nak_initial_backoff_interval	context	50 (0.05 seconds)
mim_nak_suppress_interval	context	1000 (1 second)
mim_sm_maximum_interval	context	10000 (10 seconds)
mim_sm_minimum_interval	context	200 (0.2 seconds)
mim_src_deletion_timeout	context	30000 (30 seconds)
rcv_sync_cache_timeout	receiver	2000 (2 seconds)
resolver_active_source_interval	context	1000 (1 second)
resolver_advertisement_maximum_initial_interval	source	500 (0.5 seconds)
resolver_advertisement_minimum_initial_duration	source	5000 (5 seconds)
resolver_advertisement_minimum_initial_interval	source	10 (0.01 seconds)
resolver_advertisement_minimum_sustain_duration	source	60 (1 minute)
resolver_advertisement_sustain_interval	source	1000 (1 second)

Configuration Option	Scope	Default Value
resolver_context_advertisement_interval	context	10000 (10 seconds)
resolver_no_source_linger_timeout	wildcard_receiver	1000 (1 second)
resolver_query_interval	context	100 (0.1 seconds)
resolver_query_max_interval	wildcard_receiver	0 (do not query)
resolver_query_maximum_initial_interval	receiver	200 (0.2 seconds)
resolver_query_maximum_interval	wildcard_receiver	1000 (1 second)
resolver_query_minimum_duration	wildcard_receiver	60 (1 minute)
resolver_query_minimum_initial_duration	receiver	5000 (5 seconds)
resolver_query_minimum_initial_interval	receiver	20 (0.02 seconds)
resolver_query_minimum_interval	wildcard_receiver	50 (0.05 seconds)
resolver_query_minimum_sustain_duration	receiver	60 (1 minute)
resolver_query_sustain_interval	receiver	1000 (1 second)
response_tcp_deletion_timeout	context	2000 (2 seconds)
retransmit_request_generation_interval	receiver	10000 (10 seconds)
retransmit_request_interval	receiver	500 (0.5 seconds)
transport_lbtipc_acknowledgement_interval	receiver	500 (0.5 seconds)
transport_lbtipc_activity_timeout	receiver	60,000 (60 seconds)
transport_lbtipc_client_activity_timeout	source	10,000 (10 seconds)
transport_lbtipc_sm_interval	source	10,000 (10 seconds)

Configuration Option	Scope	Default Value
transport_lbtrm_activity_timeout	receiver	60000 (60 seconds)
transport_lbtrm_ignore_interval	source	500 (0.5 seconds)
transport_lbtrm_nak_backoff_interval	receiver	200 (0.2 seconds)
transport_lbtrm_nak_generation_interval	receiver	10000 (10 seconds)
transport_lbtrm_nak_initial_backoff_interval	receiver	50 (0.05 seconds)
transport_lbtrm_nak_suppress_interval	receiver	1000 (1 second)
transport_lbtrm_preactivity_timeout	receiver	0 (zero)
transport_lbtrm_rate_interval	context	100
transport_lbtrm_sm_maximum_interval	source	10000 (10 seconds)
transport_lbtrm_sm_minimum_interval	source	200 (0.2 seconds)
transport_lbtsmx_activity_timeout	receiver	60,000 (60 seconds)
transport_lbtsmx_sm_interval	source	10,000 (10 seconds)
transport_lbtru_acknowledgement_interval	receiver	500 (0.5 seconds)
transport_lbtru_activity_timeout	receiver	60000 (60 seconds)
transport_lbtru_client_activity_timeout	source	10000 (10 seconds)
transport_lbtru_connect_interval	receiver	100 (0.1 seconds)
transport_lbtru_ignore_interval	source	500 (0.5 seconds)
transport_lbtru_nak_backoff_interval	receiver	200 [100,300] (0.2 [0.1,0.3] seconds)
transport_lbtru_nak_generation_interval	receiver	10000 (10 seconds)

Configuration Option	Scope	Default Value
transport_lbtru_nak_suppress_interval	receiver	1000 (1 second)
transport_lbtru_rate_interval	context	100
transport_lbtru_sm_maximum_interval	source	10000 (10 seconds)
transport_lbtru_sm_minimum_interval	source	200 (0.2 seconds)
transport_tcp_activity_timeout	receiver	0
transport_topic_sequence_number_info_active_threshold	source	60
transport_topic_sequence_number_info_interval	source	5000 (5 second)
ume_ack_batching_interval	context	100 (0.1 seconds)
ume_activity_timeout	receiver	0 (zero)
ume_activity_timeout	source	0 (zero)
ume_message_stability_lifetime	source	1200000 (20 minutes)
ume_message_stability_timeout	source	20000 (20 seconds)
ume_receiver_liveness_interval	context	0 (disable; do not send keepalives)
ume_registration_interval	receiver	500 (0.5 seconds)
ume_registration_interval	source	500 (0.5 seconds)
ume_retransmit_request_generation_interval	receiver	10000 (10 seconds)
ume_retransmit_request_interval	receiver	500 (0.5 seconds)
ume_source_liveness_timeout	context	0 (disable; do not track receivers)
ume_state_lifetime	receiver	0 (zero)
ume_state_lifetime	source	0 (zero)
ume_store_activity_timeout	source	3000 (3 seconds)
ume_store_check_interval	source	500 (0.5 seconds)

Configuration Option	Scope	Default Value
umq_command_interval	context	500 (0.5 seconds)
umq_delayed_consumption_report_interval	receiver	0
umq_hold_interval	receiver	10000 (10 seconds)
umq_message_retransmission_interval	context	500 (0.5 seconds)
umq_msg_total_lifetime	context	0 (zero)
umq_msg_total_lifetime	source	0 (zero)
umq_queue_activity_timeout	context	3000 (3.0 seconds)
umq_queue_check_interval	context	500 (0.5 seconds)
umq_queue_query_interval	context	200 (0.2 seconds)
umq_retransmit_request_interval	receiver	500 (0.5 seconds)
umq_ulb_check_interval	source	1000 (1 second)
umq_ulb_source_activity_timeout	receiver	10000 (10 seconds)
umq_ulb_source_check_interval	receiver	1000 (1 second)

Options That May Be Set During Operation

This section lists options that may be set during operation with a lbm_*_setopt() function.

Table 9. Options That May Be Set During Operation

Configuration Option	Scope	Default Value
implicit_batching_interval	source	200 (0.2 seconds)
implicit_batching_minimum_length	source	2048 (8192 for Microsoft Windows)
implicit_batching_type	source	
queue_age_enabled	event_queue	0
queue_count_enabled	event_queue	0

Configuration Option	Scope	Default Value
queue_delay_warning	event_queue	0 (not monitored)
queue_enqueue_notification	event_queue	
queue_service_time_enabled	event_queue	0
queue_size_warning	event_queue	0 (not monitored)
resolution_no_source_notification_threshold	receiver	0 (do not notify)
resolution_number_of_sources_query_threshold	receiver	10000000 (10 million)
resolver_active_source_interval	context	1000 (1 second)
resolver_active_threshold	context	60
resolver_maximum_advertisements	context	0 (all topics)
resolver_maximum_queries	context	0 (all topics with no source)
resolver_multicast_ttl	context	16
resolver_query_interval	context	100 (0.1 seconds)
resolver_query_max_interval	wildcard_receiver	0 (do not query)

Options (Callbacks) That Cannot Be Set From a UM Configuration File

This section lists options that require function pointers as their value and cannot be set in a UM Configuration File. These options must be set with API functions.

Table 10. Options That Cannot Be Set From a UM Configuration File

Configuration Option	Scope	Default Value
immediate_message_receiver_function	context	NULL
immediate_message_topic_receiver_function	context	NULL
mim_unrecoverable_loss_function	context	NULL
pattern_callback	wildcard_receiver	NULL
receiver_create_callback	wildcard_receiver	NULL
receiver_delete_callback	wildcard_receiver	NULL

Configuration Option	Scope	Default Value
resolver_source_notification_function	context	NULL
resolver_string_hash_function_ex	context	NULL
source_cost_evaluation_function	context	NULL
source_event_function	context	NULL
source_notification_function	receiver	NULL
ume_force_reclaim_function	source	NULL
ume_recovery_sequence_number_info_function	receiver	NULL
ume_registration_extended_function	receiver	NULL
ume_registration_function	receiver	NULL

INDEX

D

E

F

H

I

L

M

N

O

P

Q

R

S

T

U

W