UMDS (Version 6.12)

UMDS User Guide

Contents

Chapter 1

Introduction

1.1 UMDS Overview

This document introduces the basic concepts and design approaches used by Ultra Messaging Desktop Services (UMDS). The reader is assumed to already be familiar with basic UM concepts. See the UM Concepts Guide for more general information about UM.

Ultra Messaging Desktop Services (UMDS) extends Ultra Messaging to diverse desktop networks throughout an enterprise. With the UMDS client-server model, desktop applications can receive and send topic-based messages.

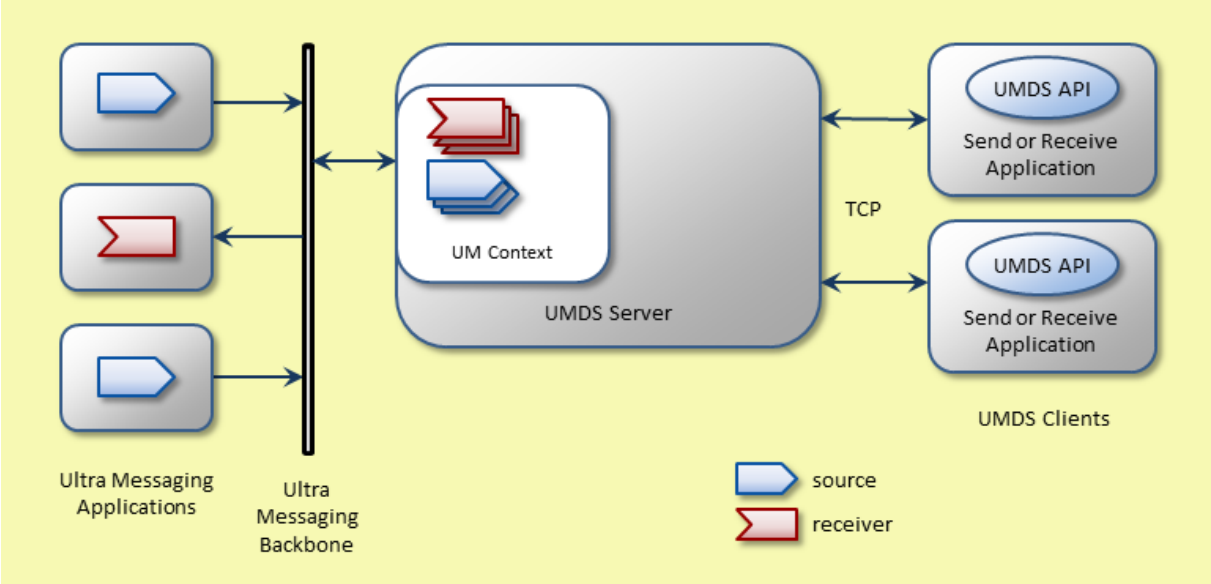
The UMDS Server runs on a server-class machine and communicates over TCP connections with desktop-class machines. A UMDS Server can also communicate with other Ultra Messaging components, such as UMS applications and other UMDS Servers.

Customer desktop applications call the UMDS API to send and receive messages over a TCP connection to the UMDS Server. This API is a subset of Ultra Messaging functionality, for Java and .NET. The UMDS Server routes messages to clients according to topic interest. The UMDS Server also routes messages between desktop UMDS Client applications and other Ultra Messaging components. For message security in the UMDS desktop environment, you can use application authentication and user authentication.

UMDS, you can centrally manage UMDS Client functionality from the UMDS Server, and use the Web Monitor to view client connection statistics and the server's current configuration file.

1.2 UMDS Architecture

A UMDS implementation comprises a UMDS server daemon and your applications written with UMDS client API calls. The UMDS server is an Ultra Messaging daemon that is part of an Ultra Messaging backbone. The UMDS server and the UMDS client application communicate using TCP.



Chapter 2

UMDS Client

UMDS includes the UMDS API, which is a library of Ultra Messaging functions for use by desktop applications. UMDS Clients communicate with the UMDS Server with TCP connections. You cannot use UDP to connect a UMDS Client to a UMDS Server.

2.1 UMDS API

The UMDS API is a more compact version of the Ultra Messaging API, intended to provide an easier and more consistent implementation of Ultra Messaging across enterprise desktops. The API is fully implemented for Java and .NET.

2.2 Server Connection

A UMDS Client application can create multiple server connections, which can be to the same UMDS Server or different UMDS Servers. The UMDS Client uses UMDS API `UMDSServerConnection` class calls to establish a server connection with the following general procedure:

1. UMDS Client application creates a `UMDSServerConnection` object.
2. UMDS Client application sets configuration options.
3. UMDS Client initiates the TCP connection.
4. UMDS Server confirms the connection.
5. UMDS Client logs into the server, and sends client configuration parameters.
6. UMDS Server authenticates the UMDS Client application and sets configuration parameters.
7. UMDS Client application creates a source or receiver, and participates in messaging functions.
8. After it no longer needs the connection, UMDS Client application closes sources or receivers.
9. UMDS Client receives ACKs and closes the connection.

2.2.1 Authenticating Applications and Users

You can authenticate either UMDS Client applications or individual desktop users when they connect to the server. By default, UMDS automatically authenticates all clients.

You can embed user passwords in a UMDS Client application or provide users with a login prompt. UMDS does not provide a login prompt facility. If you choose to authenticate applications or users, the application must deliver a user password to the UMDS Server with a `setProperty()` call. For an example, view the example application `umdssend.java` and search for `svrconn.setProperty("password", password)`.

If an application or desktop user requires authentication upon connection to the UMDS Server, set the application name, user name, and password in a Basic Authentication File. UMDS formats and transmits these parameters when requesting a connection.

2.2.2 Assigning Different Client Settings to Your Application

If your application requires different operating parameters from the UMDS Server, set the application name and parameters in a Basic Authentication File. UMDS formats and transmits these parameters when requesting a connection.

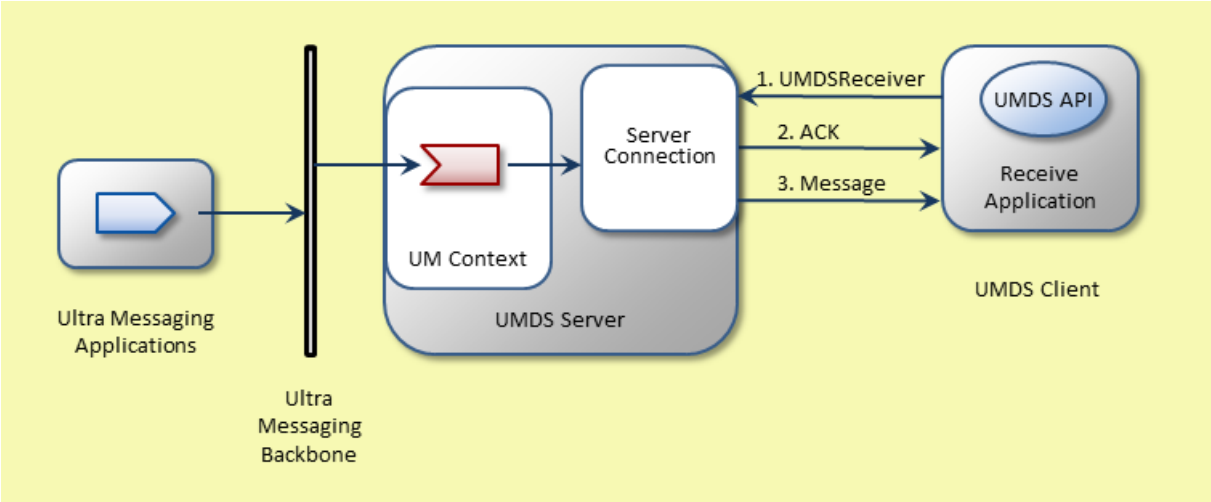
2.2.3 Application Name

UMDS Server administrators use the application name to identify the client applications connected to the server. Application developers should coordinate their application names with the server administrator to ensure proper connections and authentication.

2.3 Receiving

A UMDS Client application uses the `UMDSReceiver` class to start a receiver object and subscribe to a topic. This creates a UMS proxy receiver object at the UMDS server to listen for topic messages from other Ultra Messaging applications, including other UMDS client applications. As the UMDS Server receives messages for that topic, the UMDS Server routes the message to the proper UMDS Client applications.

In the following figure, a UMDS Client application subscribes to a topic. The UMDS Client application then receives a message on the topic from a remote Ultra Messaging sending application.

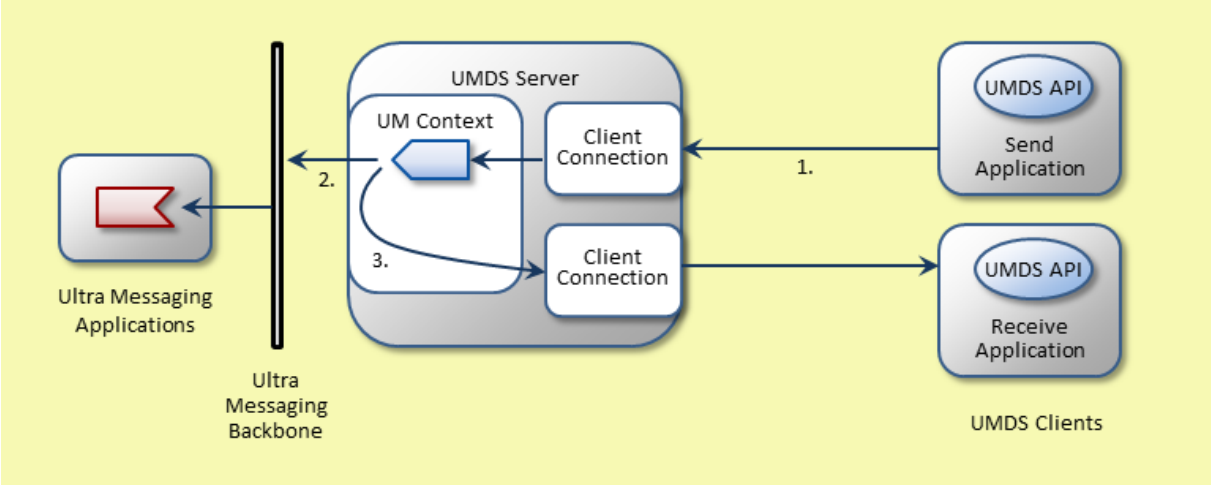


1. The UMDS Client application creates a regular receiver object at the UMDS Server. This action also specifies the topic of interest to the receiver.
2. The UMDS Server acknowledges the receiver creation.
3. The receiver begins receiving messages on the topic.

2.4 Sending

A UMDS Client application uses `UMDSSource.send` to send messages on a topic to the UMDS server. The UMDS server then uses a proxy source to stream these messages. You configure these sources in the umdsd Configuration File with the `lbm-source` Option Type. See UMDS Topic Options.

The following figure shows a UMDS Client application sending a message to all receivers listening on the topic.



A UMDS Client send performs the following steps.

1. UMDS Client application uses `UMDSSource.send` to send a message to the UMDS Server.
2. UMDS Server multicasts the message to the Ultra Messaging Backbone.

3. UMDS Server uses Ultra Messaging to send the message to other UMDS Client applications subscribed to the topic.

Clients use nonblocking sends to send messages. If the send results in an EWOULDBLOCK, the UMDS Server temporarily disables the UMDS Client send socket, which applies back pressure to the client application. The UMDS Server automatically resends the message when the Ultra Messaging source transport unblocks.

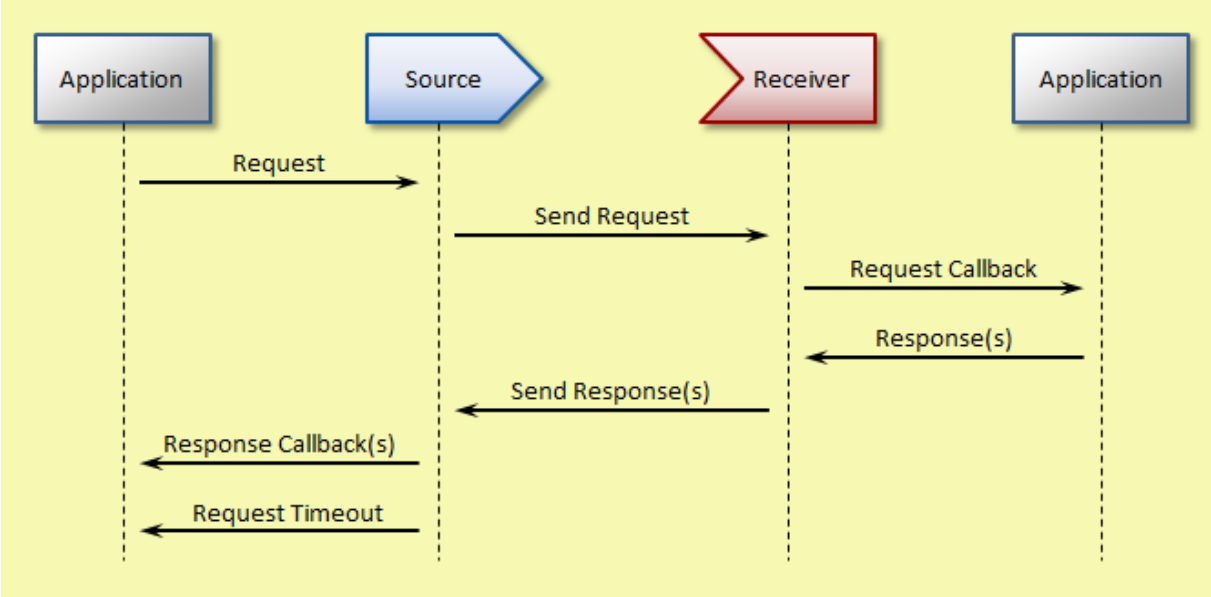
You can also write UMDS Client applications that send Immediate Messages.

2.5 Request and Response Capability

UMDS clients can issue requests, and can send and receive responses, by using the UMDS client interface. UMDS clients can exchange requests and responses with other UMDS clients or with non-UMDS Ultra Messaging sources and receivers.

You cannot explicitly cancel a request issued by a UMDS client. UMDS client requests automatically time out after a server-configured period. The server-configured period applies to all clients.

The following figure shows UMDS requests and responses:



The following table describes the request and response operations shown in the preceding figure:

Operation	Description	C# and Java Method
Request	The sending application sends a request through a UMDS source object. The `request_id` parameter must be a 32-bit integer.	request()
Send Request	The UMDS server forwards the request across the UM network.	
Request Callback	The receiver object issues a callback to the receiving application. The receipt of the callback indicates the receipt of a request.	onRequest()
Response(s)	The receiver sends zero or more responses.	respond()

Send Response(s)	The UMDS server forwards the response across the UM network.	n/a
Response Callback(s)	The source object that sent the request issues one or more callbacks to the sending application. The receipt of the callbacks indicates the receipt of a response.	onResponse()
Request Timeout	Each request has its own timeout period. When the configured timeout expires on the UMDS server, the UMDS server sends a request timeout notification to the sending client's `onEvent` callback. The timeout notification indicates that the request is closed, and that the source will deliver no more responses for that particular request. Requests always time out regardless of the number of responses received. A sending client must send new requests if it is dissatisfied with the number of responses. The server sends timeout notification messages to the sending client. Therefore, if the client disconnects from the server, the client cannot receive timeout notifications. When a client disconnects, the UMDS server cancels all outstanding requests without notification. If the client reconnects, the server does not send to the client any responses or timeout notifications for the requests that the client issued before it disconnected.	onEvent()

2.6 Using UMDS Late Join

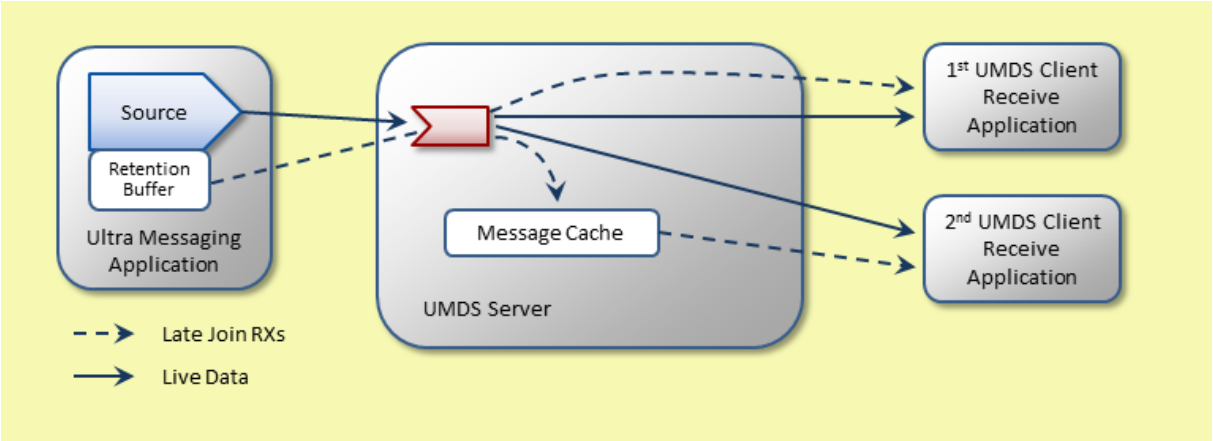
You can use Ultra Messaging Late Join functionality, however some aspects of Late Join work differently for UMDS Client receive applications.

With Late Join enabled, the first application that causes the UMDS Server to create a receiver for a particular topic initiates a Late Join request to the Ultra Messaging application's source for that topic. The UMDS Server also creates a message cache for the topic and stores the messages there to service subsequent late join requests more efficiently. When the UMDS Server receiver receives Late Join retransmissions, it sends them to the UMDS Client, flagged as RX messages.

When the receiver switches to live data, it streams the data to the client and continues to add to the message cache. When the message cache fills to maximum capacity, it deletes older messages as new messages arrive.

When a second UMDS Client receive application launches, it receives Late Join retransmissions from the UMDS Server message cache, not from the source's retention buffer. The second application might not receive the earlier messages that the first application received.

The following image shows how a first UMDS Client receive application receives its Late Join retransmissions from the source, while subsequent client applications receive Late Join retransmissions from the UMDS Server message cache.



There might be multiple sources on a topic, however the UMDS Server stores all messages on a topic into a single message cache. The message cache stores and dispatches data as whole messages, not message fragments.

To configure UMDS Client receiving applications to use Late Join, set the options in the umdsd configuration file. By default, Late Join is disabled for UMDS Client receive applications. The following example shows part of a umdsd configuration file that sets the options related to a UMDS Client receiving application that uses Late Join on topic `orderAW`.

```
<topic pattern="orderAW" type="direct">
  <umds-attributes>
    <option type="umds-receiver" name="use-late-join" value="1" />
    <option type="umds-receiver" name="message-cache-size" value="10" />
    <option type="lbm-receiver" name="use_late_join" value="1" />
    <option type="lbm-receiver" name="late_join_info_request_interval"
          value="1000" />
    <option type="lbm-receiver" name="late_join_info_request_maximum"
          value="60" />
    <option type="lbm-receiver" name="retransmit_initial_sequence_number_request"
          value="1" />
    <option type="lbm-receiver" name="retransmit_message_caching_proximity"
          value="2147483647" />
    <option type="lbm-receiver" name="retransmit_request_interval"
          value="500" />
    <option type="lbm-receiver" name="retransmit_request_maximum"
          value="0" />
    <option type="lbm-receiver" name="retransmit_request_message_timeout"
          value="10000" />
  </umds-attributes>
</topic>
```

2.6.1 Late Join UMDS Sources

You can enable Late Join for UMDS sources in the same manner as for standard Ultra Messaging sources. This must be done on the UMDS server via its configuration file. The following example excerpt from a umdsd configuration file shows how to enable Late Join for topic `orderAW`. The example also shows other relevant source Late Join options.

```
<topic pattern="orderAW" type="direct">
  <umds-attributes>
    <option type="lbm-source" name="late_join" value="1" />
    <option type="lbm-source" name="retransmit_retention_age_threshold"
```

```
          value="0" />
    <option type="lbm-source" name="retransmit_retention_size_limit"
          value="25165824" />
    <option type="lbm-source" name="retransmit_retention_size_threshold"
          value="100" />
  </umds-attributes>
</topic>
```

For more information about Late Join source configuration options, see the Ultra Messaging Configuration Guide.

Chapter 3

UMDS Example Client Applications

This section shows usages of included example applications. The same information can be displayed interactively by running the example with the "-h" command-line option.

3.1 Java Example Applications

3.1.1 umdsreceive.java

Receive messages on a single topic.

```
Usage: umdssend [options] -S address[:port] topic
  -S address[:port] = Server address/name and optionally port
                      A comma separated list of multiple servers may be provided
Available options:
  -A Suppress sending the application name to the server on login
   -c filename = read config parameters from filename
  -I Immediate Mode
  -h = help
  -l len = send messages of len bytes
  -L linger = Allow traffic to drain for up to linger seconds
              before closing the connection
  -M msgs = send msgs number of messages
  -N num_topics = Number of topics to send on
  -P msec = pause after each send msec milliseconds
  -s num_secs = Print statistics every num_secs
  -U username = set the user name and prompt for password
  -v = be verbose in reporting to the console
```

3.1.2 umdssend.java

Send messages on a single topic.

```
Usage: umdsreceive [options] -S address[:port] topic
  -S address[:port] = Server address/name and optionally port
                      A comma separated list of multiple servers may be provided
Available options:
  -A Suppress sending the application name to the server on login
   -c filename = read config file filename
  -h = help
  -M num_msgs = End after num_msgs received
  -N num_topics = Number of topics (receivers)
  -s num_secs = print statistics every num_secs along with bandwidth
  -S address:port = Server address and port
  -U username = set the user name and prompt for password
  -v = be verbose about each message
  -W = Wildcard topic
```

3.1.3 umdsresponse.java

Send responses on a single topic.

```
Usage: umdsresponse [options] -S address[:port] topic
  -S address[:port] = Server address/name and optionally port
                      A comma separated list of multiple servers may be provided
Available options:
  -A Suppress sending the application name to the server on login
   -c filename = read config file filename
  -h = help
  -M num_msgs = End after num_msgs received
  -N num_topics = Number of topics (receivers)
  -r len = send responses of len bytes
  -s num_secs = print statistics every num_secs along with bandwidth
  -S address:port = Server address and port
  -U username = set the user name and prompt for password
  -v = be verbose about each message
  -W = Wildcard topic
```

3.1.4 umdsrequest.java

Send requests and messages on a single topic.

```
  -S address[:port] = Server address/name and optionally port
                      A comma separated list of multiple servers may be provided
Available options:
  -A Suppress sending the application name to the server on login
   -c filename = read config parameters from filename
  -I Immediate Mode
  -h = help
  -l len = send messages of len bytes
  -L linger = Allow traffic to drain for up to linger seconds
              before closing the connection
  -M msgs = send msgs number of messages
  -N num_topics = Number of topics to send on
  -P msec = pause after each send msec milliseconds
  -r len = send requests of len bytes
```

```
  -s num_secs = Print statistics every num_secs
  -U username = set the user name and prompt for password
  -v = be verbose in reporting to the console
```

3.2 .NET Example Applications

3.2.1 umdssend.cs

Send messages on a single topic.

```
Usage: umdssend [options] -S address[:port] topic
  -S address[:port] = Server address/name and optionally port
                      A comma separated list of multiple servers may be provided
Available options:
  -A Suppress sending the application name to the server on login
   -c filename = read config parameters from filename
  -I Immediate Mode
  -h = help
  -l len = send messages of len bytes
  -L linger = Allow traffic to drain for up to linger seconds
              before closing the connection
  -M msgs = send msgs number of messages
  -N num_topics = Number of topics to send on
  -P msec = pause after each send msec milliseconds
  -s num_secs = Print statistics every num_secs
  -U username = set the user name and prompt for password
  -v = be verbose in reporting to the console
```

3.2.2 umdsreceive.cs

Receive messages on a single topic.

```
Usage: umdsreceive [options] -S address[:port] topic
  -S address[:port] = Server address/name and optionally port
                      A comma separated list of multiple servers may be provided
Available options:
  -A Suppress sending the application name to the server on login
   -c filename = read config file filename
  -h = help
  -M num_msgs = End after num_msgs received
  -N num_topics = Number of topics (receivers)
  -s num_secs = print statistics every num_secs along with bandwidth
  -S address:port = Server address and port
  -U username = set the user name and prompt for password
  -v = be verbose about each message
  -W = Wildcard topic
```

3.2.3 umdsresponse.cs

Send responses on a single topic.

```
Usage: umdsresponse [options] -S address[:port] topic
  -S address[:port] = Server address/name and optionally port
                      A comma separated list of multiple servers may be provided
Available options:
  -A Suppress sending the application name to the server on login
  -c filename = read config file filename
  -h = help
  -M num_msgs = End after num_msgs received
  -N num_topics = Number of topics (receivers)
  -r response message length
  -s num_secs = print statistics every num_secs along with bandwidth
  -S address:port = Server address and port
  -U username = set the user name and prompt for password
  -v = be verbose about each message
  -W = Wildcard topic
```

3.2.4 umdsrequest.cs

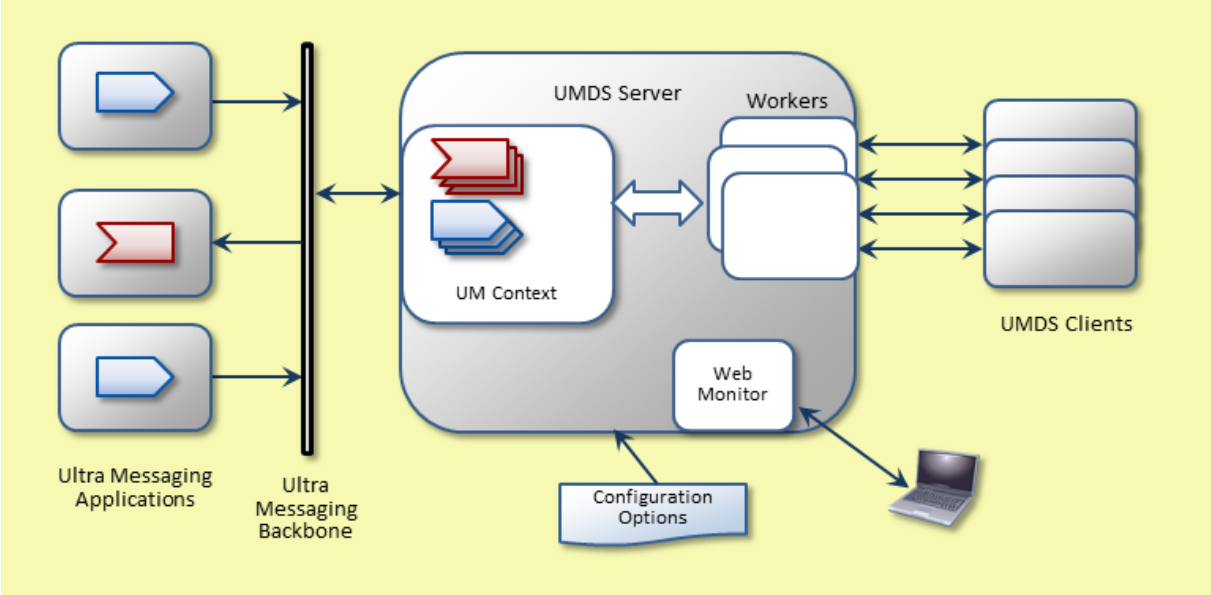
Send requests and messages on a single topic.

```
Usage: umdsrequest [options] -S address[:port] topic
  -S address[:port] = Server address/name and optionally port
                      A comma separated list of multiple servers may be provided
Available options:
  -A Suppress sending the application name to the server on login
  -c filename = read config parameters from filename
  -I Immediate Mode
  -h = help
  -l len = send messages of len bytes
  -L linger = Allow traffic to drain for up to linger seconds
              before closing the connection
  -M msgs = send msgs number of messages
  -N num_topics = Number of topics to send on
  -P msec = pause after each send msec milliseconds
  -r len = send requests of len bytes
  -s num_secs = Print statistics every num_secs
  -U username = set the user name and prompt for password
  -v = be verbose in reporting to the console
```

Chapter 4

UMDS Server

The UMDS Server is a daemon that enables UMDS Clients to exchange messages with standard Ultra Messaging sending and receiving applications. The following image shows some of the UMDS Server components and functionality.



The UMDS Server consists of the following components:

- **UMDS Server** - The UMDS Server is a daemon that contains a standard Ultra Messaging context, which sends and receives messages.
- **Workers** - Workers exchange messages with UMDS Client applications over TCP connections. You can configure and run multiple worker instances to provide parallelism.
- **Web Monitor** - Use the Web Monitor, a web-based user interface, to control operation of, and view the status of, the UMDS Server.
- **Configuration Options** - When the umdsd UMDS Server starts, it reads configuration options from a umdsd configuration file.

4.1 User Authentication

You can assign a user or application name, or both, and a password, to authenticate your client applications or individual desktop users. Authentication occurs when a user or application requests a server connection.

You assign user passwords in the <user> element in the Basic Authentication File, which you specify in the <authentication> element in the umdsd Configuration File. You can specify an application or user name in the UMDS user Element. If you specify <none/> for the <authentication> element in the umdsd configuration file, the UMDS Server authenticates all applications and users.

4.2 Client Application Parameters

When a client application requests a server connection, the UMDS Server looks at a sequence of <client>element settings to determine what parameters to apply to the client application.

Operating parameters control the degree of resource utilization allowed by a client application, such as keep-alive intervals and thresholds. You can override all client application parameter values from multiple sources.

With overrides, you can select the optimal trade offs between flexibility and centralized control of client configuration. For example, a deployment that requires control would allow the client application to override fewer settings, which might simplify the job of the application programmer, but increase the responsibility of the server administrator.

UMDS Server factory defaults are the least restrictive, allowing clients to change any setting. However, you can configure a more restrictive, generic set of UMDS daemon <client>element settings that disable client overrides for certain settings. You can also set up acceptable ranges of values for other settings. Plus, you can then configure other applications or users to use different settings, which maybe more or less restrictive than the generic set of parameters.

The following table shows the sequence of <client>element settings the UMDS Server goes through when choosing the operating parameters for a particular UMDS Client application connection.

Step	Client Settings Used by UMDS Server	Can Configure Parameters	Can Authenticate User
1	**Factorydefaults.** Requires no action by either application programmers or UMDS Server administrator.	Yes	No
2	**umdsdconfiguration file <client>element settings.** Overrides factory defaults. You can apply different settings to different UMDS Servers across the enterprise.	Yes	No
3	**BasicAuthentication File <application>settings.** Overrides umdsd configuration file. These settings indicate one or more client applications that require different settings from the server settings in the umdsd configuration file.	Yes	No
4	**BasicAuthentication File <user>settings.** Override Basic Authentication File <application>. These settings indicate one or more users that require either authentication or different operating parameters or both.	Yes	Yes

Step	Client Settings Used by UMDS Server	Can Configure Parameters	Can Authenticate User
5	**Clientapplication requests of certain settings.** umdsd configuration file and Basic Authentication File settings can deny the client application requests with the `client-write`attribute for any operation parameter.	Yes	No

The UMDS Server sends the resulting settings to the client application as the final phase of the initial connection handshake.

Note

For Steps 3 and 4, if you specify no Basic Authentication File and the client application does not provide a password, the UMDS Server permits the connection. However, the UMDS Server does not apply any `<application>`or `<user>`settings requested by the client application.

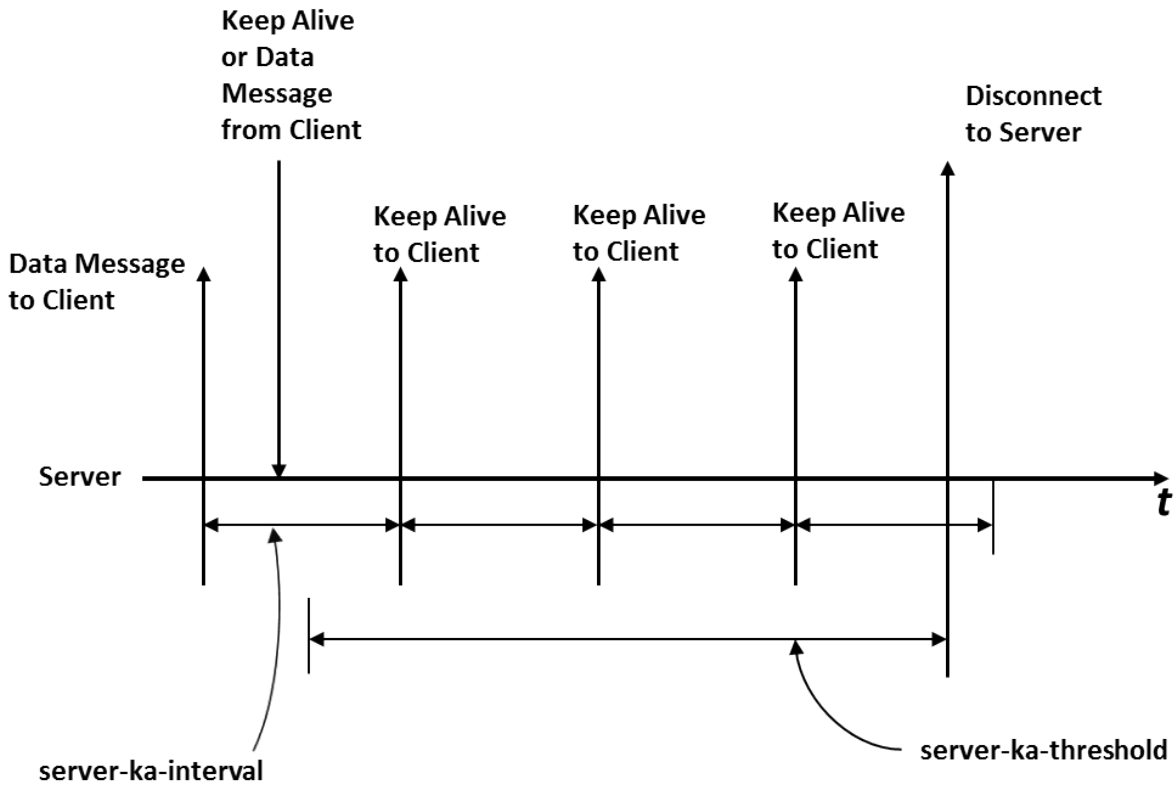
4.3 Keep Alive Timers During Idle Periods

UMDS provides keep alive timers so during periods of message inactivity, servers and clients can be aware of any unresponsiveness. Both the server and client have a keep alive interval and threshold. You configure these timers in the UMDS Element "<client>" of the umds configuration file.

- The interval defines the time period in milliseconds between keep alive messages sent from either a client or server to the other (UMDS Element "<client-ka-interval>" and UMDS Element "<server-ka-interval>"). When a client or server sends a data message or keep alive message, it resets the interval.

- The threshold defines the timeout for message traffic from the server or client to the other (keep alive or otherwise). A timeout results in disconnection. (UMDS Element "<client-ka-threshold>" and UMDS Element "<server-ka-threshold>"). When a client or server receives a data message or keep alive message, it resets the threshold.

The following figure shows the interaction of the interval and threshold configuration elements for a server when traffic from a client ceases.



1. When the server sends a message (data or keepalive) to the client, it resets the `server-ka-interval`. During the period when the server sends no data messages to the client, at the end of the `server-ka-interval`, it sends a keepalive message and resets the `server-ka-interval`. It continues to send keepalive messages at the expiration of the `server-ka-interval` as long as the connection exists.
2. When the server receives a message (data or keepalive) from the client, it resets the `server-ka-threshold`.
3. If the server receives no messages from the client, it disconnects the client when the `server-ka-threshold` timer expires.

During periods of inactivity, the interval used by one side prevents the threshold from being reached on the other side. Therefore, there is a relationship between `client-ka-interval` and `server-ka-threshold` and also between `server-ka-interval` and `client-ka-threshold`. The interval should be less than the related threshold.

4.4 Message Queue

The UMDS Server maintains a default message queue for the messages that it forwards to UMDS Client receiving applications. Because connections to UMDS Client applications use receiver-paced TCP, the UMDS Server queues these messages to prevent loss from fast senders.

In the UMDS Server configuration file, you can configure the queue's age limit and size limit with the `<server>` element `msg-age-limit` and `msg-q-size-limit` attributes. The UMDS Server deletes messages that stay in the queue longer than the age limit. If the queue reaches the configured size limit, the UMDS Server deletes the oldest messages in the queue to make room for new messages.

Note

When considering memory usage, be aware that when the UMDS Server creates sources and receivers, it also creates other buffers and caches. Examples include the topic Late Join message cache, and buffers created using the standard Ultra Messaging configuration options. Note also that when UMDS configuration files do not specify standard Ultra Messaging configuration options, these options use default values.

4.4.1 Per-Topic Message Queues

When the default queue reaches its size limit and a new message arrives, the UMDS Server deletes the oldest message. This might be undesirable if the oldest message is for a different topic than the newest message. The UMDS Server can also maintain per-topic message queues to address the possibility of message loss across topics when using the single default message queue. You can configure topics in the UMDS Server configuration file to use their own message queue or the default queue with the `<topics>` element. You can configure topics by pattern matching or explicit names. Each queue maintains its own statistics, which you can view in the Web Monitor Client Details page.

4.4.2 Configuring Message Queue Size

Use the following information and examples to determine the optimum settings for the `msg-q-size-limit` parameter, and to properly size the physical memory contained in the UMDS Server host machine.

n

4.4.3 Approximating Per-Queue Memory Use

The `msg-q-size-limit` parameter sets the maximum number of message payload bytes that the UMDS Server allows before deleting older messages to make room for new messages. This limit does not include the Ultra Messaging overhead of approximately 900 bytes per message.

For example, if a message queue contains 2,000 25-byte messages, this total of 50,000 payload bytes does not exceed the `msg-q-size-limit` parameter default size of 1,048,576 bytes. However, with the overhead of 900 bytes, the queue actually uses 925 x 2,000 = 1,850,000 bytes.

4.4.4 Approximating the Number of Messages Per Queue

You can calculate the maximum number of messages that fit into a queue if you know the message payload size. For example, if all messages have a payload size of 25 bytes, then the default configuration of 1,048,576 bytes indicates that up to (1,048,576 / 25) = 41,943 messages can be enqueued.

When using individual topic queues, each UMDS Client has multiple queues. Thus, the total memory use is the sum of the memory use of all topic queues.

4.4.5 Calculating Optimal Queue Size Limits

If all connected clients are able to keep up with average message traffic, then the message queue consumes little or no memory. However, if a burst of high-rate traffic occurs, queues can fill up quickly as clients struggle to keep up. The following example demonstrates peak memory utilization.

- If each message contains 25 bytes of user data, and the default queue size limit is set to 1,048,576 bytes, then the queue can grow to contain 41,943 messages.
- If each 25-byte message consumes a total of 925 bytes including the UMDS overhead, a full queue consumes about 37 MB of memory.
- If 80 client queues all fill to capacity, these queues collectively consume about 3 GB of memory.

4.5 Worker Configuration Guidelines

The default configuration of the UMDS Server assigns three workers to service all client connections. For each worker, four primary threads process data as follows:

- **One Client-to-Server Data Thread:** handles all data produced by all sending Clients
- **One Server-to-Client Data Thread:** handles all data dispatched to all receiving Clients
- **Two Ultra Messaging Backbone Context Processing Threads:** shared by all workers
  - UMS context worker thread processes all UMS streaming data from the Ultra Messaging Backbone, using UMS unicast and multicast transports.
  - MIM context worker thread sends MIM data from sending Clients to the Ultra Messaging Backbone.

Four additional threads handle low volume internal command and control. These threads require a small fraction of the processing done by the primary threads.

4.5.1 Increasing Number of UMDS Workers

For each additional worker configured, the number of Client to Server Data Threads and Server to Client Data Threads increases by one. For example, four workers will result in four Client to Server Data Threads and four Server to Client Data Threads. Regardless of the number of workers configured, UMDS uses only two Informatica High Speed Message Backbone Processing Threads and 4 additional low volume command and control threads.

4.5.2 Workers CPU Cores and Performance

Due to the number of primary threads as described above, increasing the number of workers does not necessarily increase performance if the number of CPU Cores is four or less. For systems with greater than four cores, set the number of workers so the number of primary threads never exceeds the number of CPU Cores. See the table below.

Workers	Client to Server Threads	Server to Client Threads	Ultra Messaging Threads	Number of Primary Threads	Number of CPU Cores Recommended
1	1	1	2	4	4 or less
2	2	2	2	6	6
3	3	3	2	8	8
4	4	4	2	10	10

Proper performance analysis to determine the optimal configuration is recommended.

4.5.3 Workers Versus Client Load

Using multiple workers distributes work load among multiple threads. UMDS assigns each new client connection to a worker in round robin fashion. For the case of 1 worker, the single worker manages all client connections. If you configure two workers, then each worker services half the clients. Note, however, the following two limitations:

- The server does not perform any load balancing of clients. For example, if two workers are servicing three clients each, but one of the six clients produces and/or consumes all the data, that client's worker will not distribute any processing chores to the other worker.
- UMDS assigns new clients to workers in a round-robin fashion without regard to the current load. If, for example, a number of clients are evenly distributed across the workers and then all the clients assigned to a single worker disconnect, the UMDS Server does not move any clients to the idle worker. In addition, new connections continue to be assigned in a round-robin fashion.

Chapter 5

Umdsd Man Page

`umdsd` *options configfile*

Description

`umdsd` runs the UMDS Server and requires a `umdsd` configuration file.

Options

`-d` or `-dump-dtd` - option dumps the DTD file used to validate the umdsd configuration file to standard output. After dumping the DTD, `umdsd` exits instead of initiating the UMDS Server.

`-v` or `-validate` - option validates the `umdsd` configuration file against the DTD. After attempting validation, `umdsd` exits instead of initiating the UMDS Server. The exit status will be 0 for a configuration file validated by the DTD and non-zero otherwise.

`-f` or `-detach` option forks `umdsd`, detaches the child from the controlling terminal and the parent exits immediately. The `umdsd` normally remains attached to the controlling terminal and runs until interrupted.

`-h` or `-help` - option provides command line help.

Note

> The UMDS Server may, under some conditions, return an error message similar to `error, not enough file descriptors`. This may be caused by exceeding the default limit of 1024 file descriptors per process. To override this limit, edit `/etc/security/limits.conf` and add a line for the user name that starts the UMDS Server and increase it to 2048 or higher. This enables use of the `ulimit -n` command (or `limit openfiles` on some systems). Use `ulimit -n` just before starting the UMDS Server to activate the new limit.

Exit Status

The exit status from `umdsd` is 0 for success and some non-zero value for failure.

Chapter 6

Daemon Statistics

The UMDS Server has a simple web server which provides operational information. This information is important for monitoring the operation and performance of these components. However, while the web-based presentation is convenient for manual, on-demand monitoring, it is not suitable for automated collection and recording of operational information for continuous monitoring and historical analysis.

The Daemon Statistics feature supports the background publishing of their operational information via UM messages. Monitoring systems can now subscribe to this information in much the same way that UM transport statistics can be subscribed.

6.1 Daemon Statistics Structures

The operational information is published as messages of different types sent over a normal UM topic source (topic name configurable). Each message is in the form of a binary, C-style data structure.

There are generally two categories of messages: *config* and *stats*. A given instance of a category "config" message does not have content which changes over time. An instance of a category "stats" message has content that *does* change over time. The daemon-specific documentation indicates which messages are in which category.

Each message type is configured for a publishing interval. When the publishing interval for a message type expires, the possible messages are checked to see if its content has materially changed since the last interval. If not, then the message is *not* republished. The publishing interval for a stat message is typically set to shorter periods to see those changes as they occur.

6.2 Daemon Statistics Binary Data

The messages published are in binary form and map onto the C data structures defined for each message type.

The byte order of the structure fields is defined as the host endian architecture of the publishing daemon. Thus, if a monitoring host receiving the messages has the same endian architecture, the binary structures can be used directly. If the monitoring host has the opposite endian architecture, the receiver must byte-swap the fields.

The message structure is designed to make it possible for a monitoring application to detect a mismatch in endian architecture. Detection and byte swapping is demonstrated with daemon-specific example monitoring applications.

6.3 Daemon Statistics Versioning

Each message sent by the daemon consists of a standard header followed by a message-type-specific set of fields. The standard header contains a `version` field which identifies the version of the C include file used to build the daemon.

The UMDS Server is built with the include file `umdsdmonmsgs.h`. With each daemon statistics message sent by the UMDS Server, it sets the header version field to **LBM_UMDSD_DMON_VERSION**. With each new release of the UMDS package, if that include file changes in a substantive way, the value of **LBM_UMDSD_DMON_VERSION** is increased. In this way, a monitoring application can determine if it is receiving messages from a store daemon whose data structures match the monitoring application's structure definitions.

6.4 Daemon Statistics Requests

The daemon can optionally be configured to accept command-and-control requests from monitoring applications. There are two categories of these requests: *"snapshot"* and *"config"*. "Snapshot" requests tell the daemon to immediately republish the desired stats and/or configs without waiting until the next publishing interval. These requests might be sent by a monitoring application which has only just started running and needs a full snapshot of the operational information. "Config" requests tell the daemon to modify an operational parameter of the running daemon.

The monitoring application sends a request to the daemon, and the daemon sends status messages in response. The exchanges are made via standard UM topicless immediate Request Response messaging. Informatica recommends the use of Unicast Immediate Messaging (UIM) for sending the requests using lbm_unicast_immediate_↩ request(). To use UIM effectively, Informatica recommends configuring the daemon monitor context for a specific UIM interface and port using: request_tcp_port (context) and request_tcp_interface (context). This enables the monitoring application to know how to address the request UIMs to the proper daemon.

The request message is formatted as a simple ASCII string. For the SRS service, the request message is formatted as a JSON message. The request is sent as a non-topic unicast immediate request message. The daemon reacts by parsing the request and sending a UM response with a success/failure response. If the request was parsed successfully, the daemon then performs the requested operation (republishing the data or modifying the operational parameter). There are daemon-specific example applications which demonstrate the use of this request feature.

6.5 UMDS Daemon Statistics Structures

The different message types are:

- **UMDS_DSTATTYPE_CFG**
- **UMDS_DSTATTYPE_MALLINFO**
- **UMDS_DSTATTYPE_CONNSUMMARY**
- **UMDS_DSTATTYPE_CLIENTPERMS**
- **UMDS_DSTATTYPE_CLIENTATTRS**
- **UMDS_DSTATTYPE_PERTOPIC**
- **UMDS_DSTATTYPE_TOPICTOTALS**
- **UMDS_DSTATTYPE_SOURCE**

- **UMDS_DSTATTYPE_RECEIVER**
- **UMDS_DSTATTYPE_SMARTHEAP**
- **UMDS_DSTATTYPE_WORKER**

Each one has a specific structure associated with it, as detailed in the file umdsdmonmsgs.h.

Note that message type ending with "CFG" is in the config category. All others are in the stats category. See Daemon Statistics Structures for information on how the two categories are handled differently.

6.6 UMDS Daemon Statistics Byte Swapping

A monitoring application receiving these messages must detect if there is an endian mismatch (see Daemon Statistics Binary Data). The header structure **umdsd_dstat_msg_hdr_t_stct** contains a 16-bit field named magic which is set equal to **LBM_UMDS_DMON_MAGIC**. The receiving application should compare it to **LBM_UMDS_DMO↩ N_MAGIC** and **LBM_UMDS_DMON_ANTIMAGIC**. Anything else would represent a serious problem.

If the receiving app sees:

```
magic == LBM_UMDS_DMON_MAGIC
```

then it can simply access the binary fields directly. However, if it sees:

```
magic == LBM_UMDS_DMON_ANTIMAGIC
```

then *most* (but not all) binary fields need to be byte-swapped. See umdsdmon.c for an example, paying special attention to the macros COND_SWAPxx (which *conditionally* swaps based on the magic test) and the functions byte_swapXX() (which performs the byte swapping).

6.7 UMDS Daemon Statistics String Buffers

UMDS Daemon Statistics data structures sometimes contain string buffers. Strings in these data structures are always null-terminated. These messages are generally sent as fixed-length equal to the sizes of the structures, and therefore include all of the declared bytes of the string fields, even if the contained string uses fewer bytes than declared. For example, the structure **umdsd_dstat_connection_summary_record_stct** contains the field user_name which is a char array of size UMDS_DSTAT_CFG_EL_NAME_SZ + 1. If user_name is set to "p1", then only 3 bytes of the buffer are used (including the null string terminator). However, all UMDS_DSTAT_C↩ FG_EL_NAME_SZ + 1 bytes will be sent in the **UMDS_DSTATTYPE_CONNSUMMARY** message type.

There is one exception to this rule: **UMDS_DSTATTYPE_CFG**.

The **UMDS_DSTATTYPE_CFG** message is of type **umdsd_dstat_config_msg_stct**, which contains the structure **umdsd_dstat_config_record_stct**, which contains the field data. This field is variable length and contains a null-terminated string.

6.8 UMDS Daemon Statistics Configuration

UMDS daemon statistics are configured by the UMDS Element "<daemon-monitor>" in the UMDS server configuration.

6.9 UMDS Daemon Statistics Requests

The UMDS Daemon supports a monitoring application to send a specific set of requests to control the operation of Daemon Statistics. The remote-snapshot-request and remote-config-changes-request configuration elements control whether the Store enables this request feature (defaults to disabled).

If enabled, the monitoring application can send a command message to the UMDS in the form of a topicless unicast immediate "request" message (see lbm_unicast_immediate_request() with NULL for topic). The format of the message is a simple ascii string, with or without null termination. Due to the simple format of the message, no data structure is defined for it.

When the UMDS receives and validates the command, it sends a UM response message back to the requesting application containing a status message (which is *not* null-terminated). If the status was OK, the Store also performs the requested action.

The example program `umdsdcmd.c` demonstrates the correct way to send the messages and receive the responses.

Commands enabled by remote-snapshot-request:

version

The UMDS Server returns in its command response the value of **LBM_UMDSD_DMON_VERSION**. No daemon statistics messages are published.

snap mallinfo

The UMDS Server immediately publishes the memory allocation usage message of type **UMDS_DSTATTYPE_MALLINFO**.

snap cfg

The UMDS Server immediately publishes the UMDS configuration message(s) **UMDS_DSTATTYPE_CFG**.

snap connsum

The UMDS Server immediately publishes connection summary information message(s) **UMDS_DSTATTYPE_CONNSUMMARY**.

snap conndet

The UMDS Server immediately publishes connection details message(s), consisting of an initial **UMDS_DSTATTYPE_CONNSUMMARY** message, followed by zero or more of the following messages: **UMDS_DSTATTYPE_CLIENTPERMS**, **UMDS_DSTATTYPE_PERTOPIC**, **UMDS_DSTATTYPE_TOPICTOTALS**, **UMDS_DSTATTYPE_RECEIVER**, **UMDS_DSTATTYPE_SOURCE**, and **UMDS_DSTATTYPE_CLIENTATTRS**.

snap worksum

The UMDS Server immediately publishes worker summary message(s), consisting of an initial **UMDS_DSTATTYPE_WORKER** message, followed by zero or more **UMDS_DSTATTYPE_CONNSUMMARY** messages.

snap workdet

The UMDS Server immediately publishes worker summary message(s), consisting of an initial **UMDS_DSTATTYPE_WORKER** message, followed by zero or more of the following messages: **UMDS_DSTATTYPE_CLIENTPERMS**, **UMDS_DSTATTYPE_PERTOPIC**, **UMDS_DSTATTYPE_TOPICTOTALS**, **UMDS_DSTATTYPE_RECEIVER**, **UMDS_DSTATTYPE_SOURCE**, and **UMDS_DSTATTYPE_CLIENTATTRS**.

Commands enabled by remote-config-changes-request:

mallinfo N

Set the publishing interval for memory allocation usage.
For example: `mallinfo 5`

worksum N

Set the publishing interval for the worker summary messages.
For example: `worksum 5`

workdet N

Set the publishing interval for the worker detail messages.
For example: `workdet 5`

6.10 UMDS Daemon Statistics Example Files

The following files are provided in source code form to assist users in writing monitoring applications using the UMDS Daemon Statistics feature.

- `umdsdmon.c` - C program to read UMDS Daemon Statistics and print them.
- `umdsdcmd.c` - C program to send Daemon Statistics commands to the UMDS server.
- `umdsdmonmsgs.h` - C header file which defines the internal structures. You can also see its `Doxygen documentation`.
- `getopt.c` - GNU command-line option parsing code (useful for building `umdsdmon.c` and `umdsdcmd.c` on Windows platform).
- `replgetopt.h` - C header file for getopt.c.

Chapter 7

UMDS Web Monitor

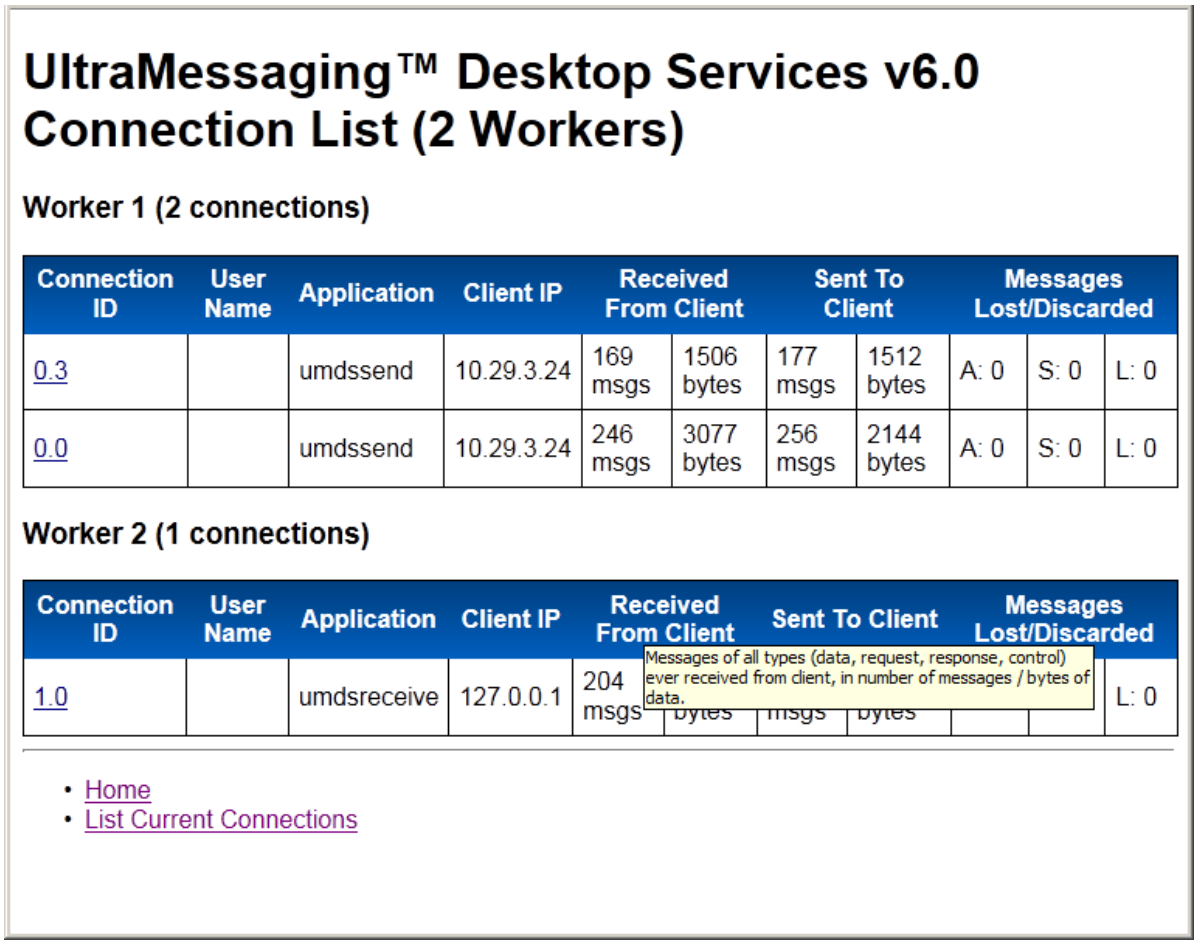
Use the UMDS Web Monitor to monitor the UMDS Server's connections. The monitor displays statistics for each connection, with a link to more details about the client connection. You configure the UMDS Web Monitor with the UMDS Element "<web-monitor>" in the UMDS daemon configuration file.

7.1 Main Menu

Use the UMDS Web Monitor Main Menu to view connections, the configuration file, current memory allocation statistics, or to stop the UMDS Server daemon.

7.2 List Current Connections

The Connection List page shows all current UMDS client-server connections. The page organizes connections by Worker subsystem. If you mouse over table elements, you see pop-up tooltips displays.



UltraMessaging™ Desktop Services v6.0 Connection List (2 Workers)

Worker 1 (2 connections)

Connection ID	User Name	Application	Client IP	Received From Client		Sent To Client		Messages Lost/Discarded		
0.3		umdssend	10.29.3.24	169 msgs	1506 bytes	177 msgs	1512 bytes	A: 0	S: 0	L: 0
0.0		umdssend	10.29.3.24	246 msgs	3077 bytes	256 msgs	2144 bytes	A: 0	S: 0	L: 0

Worker 2 (1 connections)

Connection ID	User Name	Application	Client IP	Received From Client	Sent To Client	Messages Lost/Discarded
1.0		umdsreceive	127.0.0.1	204 msgs		L: 0

Messages of all types (data, request, response, control) ever received from client, in number of messages / bytes of data.

- Home
- List Current Connections

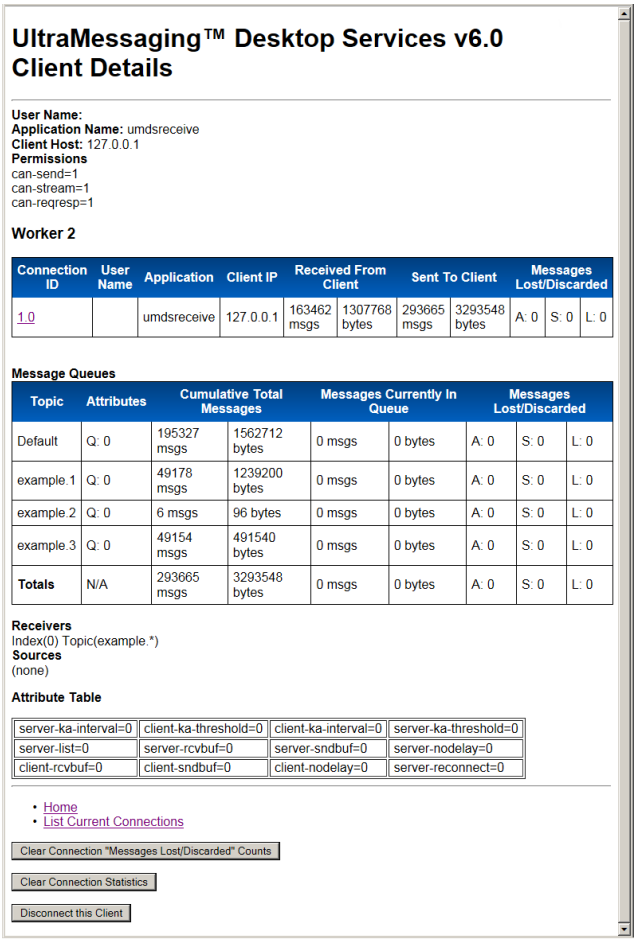
The Connection List display has the following column headings:

- **Connection ID** - Identifies a connection to a UMDS Client in *x.y* format, where *x* designates the worker thread number and *y* is a connection identifier for that thread. Click a Connection ID to go to the Connection Details page.
- **User Name** - Name of the user logged in for this connection, as sent by the client. If the UMDS Client does not supply a user name, this item is blank. You specify authenticated users in the Basic Authentication File.
- **Application** - Name of the client application connected to the server, as sent by the client. You can specify an application name in the Basic Authentication File or from within the application.
- **Client IP** - IP address of the host where the UMDS Client application is running.
- **Received From Client** - Number of messages and number of bytes that the UMDS Server has received from the UMDS Client applications. Pause on the value to see a tooltip display separating the value into user data, requests, responses, and control data.
- **Sent To Client** - Number of messages and number of bytes that the UMDS Server has sent to UMDS Client applications. Pause on the value to see a tooltip display separating the value into user data, requests, responses, and control data.
- **Messages Lost/Discarded** - Total number of messages that the UMDS Server either lost or discarded, based on the following reasons:
  - **A:** - messages dropped because the message queue has reached the limit set by parameter `msg-age-limit`.
  - **S:** - messages dropped because the message queue has reached the limit set by parameter `msg-q-size-limit`.

– **L:** - messages never enqueued on the client queue in the UMDS Server. Transport level loss can happen between the UMDS Server and external Ultra Messaging sources, or between sources and receivers internal to the UMDS Server.

7.3 Connection Details

This page displays information specific to the Connection ID clicked on in the UMDS Connection List page.



UltraMessaging™ Desktop Services v6.0
Client Details

User Name:
Application Name: umdsreceive
Client Host: 127.0.0.1
Permissions
can-send=1
can-stream=1
can-reqresp=1

Worker 2

Connection ID	User Name	Application	Client IP	Received From Client		Sent To Client		Messages Lost/Discarded		
1.0		umdsreceive	127.0.0.1	163462 msgs	1307768 bytes	293665 msgs	3293548 bytes	A: 0	S: 0	L: 0

Message Queues

Topic	Attributes	Cumulative Total Messages		Messages Currently In Queue		Messages Lost/Discarded		
Default	Q: 0	195327 msgs	1562712 bytes	0 msgs	0 bytes	A: 0	S: 0	L: 0
example.1	Q: 0	49178 msgs	1239200 bytes	0 msgs	0 bytes	A: 0	S: 0	L: 0
example.2	Q: 0	6 msgs	96 bytes	0 msgs	0 bytes	A: 0	S: 0	L: 0
example.3	Q: 0	49154 msgs	491540 bytes	0 msgs	0 bytes	A: 0	S: 0	L: 0
Totals	N/A	293665 msgs	3293548 bytes	0 msgs	0 bytes	A: 0	S: 0	L: 0

Receivers
Index(0) Topic(example.*)
Sources
(none)

Attribute Table

server-ka-interval=0	client-ka-threshold=0	client-ka-interval=0	server-ka-threshold=0
server-list=0	server-rcvbuf=0	server-sndbuf=0	server-nodelay=0
client-rcvbuf=0	client-sndbuf=0	client-nodelay=0	server-reconnect=0

- Home
- List Current Connections

Clear Connection "Messages Lost/Discarded" Counts
Clear Connection Statistics
Disconnect this Client

The Client Details page begins with the following items:

- **User Name** - Name of the user authenticated for this connection, as sent by the client. This item is blank if no user is authenticated. You specify authenticated users in the Basic Authentication File.
- **Application Name** - Name of the client application connected to the server, as sent by the client. You specify applications in the Basic Authentication File.
- **Client Host** - IP address of the host where the UMDS Client application is running.
- **Permissions** - Permissions configured for the Application or User Name. *These settings are deprecated and have no effect.*

The Message Queues display has the following column headings:

- **Topic** - For per-topic message queues, this is the topic name. `Default` is the non-topic-specific default message queue.
- **Attributes** - The configured queue size limit for this message queue.
- **Cumulative Total Messages** - The number of messages that have entered the queue since being created or reset.
- **Messages Currently in Queue** - The number of messages the queue is holding at the time the page was loaded or refreshed. The UMDS Server has not yet delivered these messages to a UMDS Client receiving application.
- **Messages Lost/Discarded** - Total number of messages that the UMDS Server either lost or discarded, based on the following reasons:
  - **A:** - messages dropped because the message queue has reached the limit set by parameter `msg-age-limit`
  - **S:** - messages dropped because the message queue has reached the limit set by parameter `msg-q-size-limit`
  - **L:** - messages never enqueued on the client queue in the UMDS Server. Transport level loss can happen between the UMDS Server and external Ultra Messaging sources, or between sources and receivers internal to the UMDS Server.

The Client Details page ends with the following items:

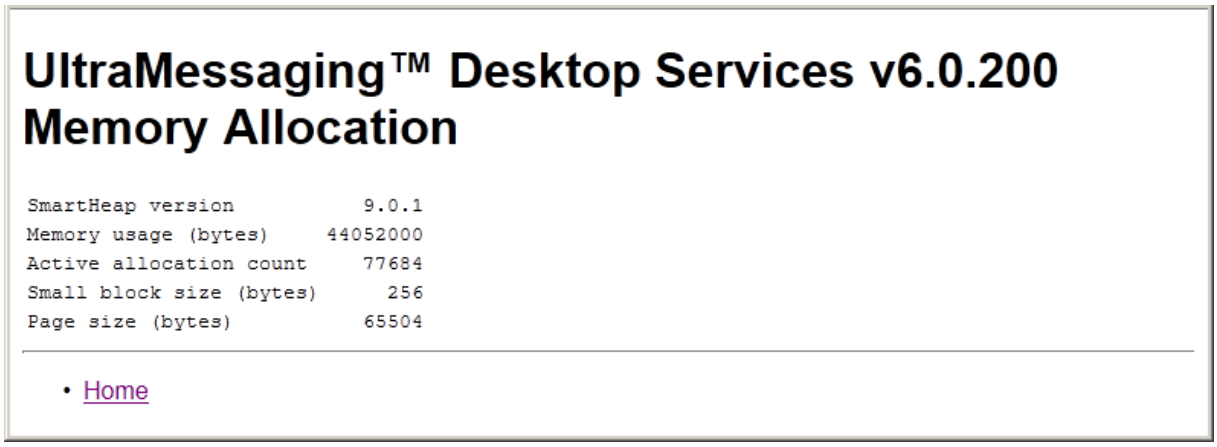
- **Receivers** - Receivers listed by index number and topic name.
- **Sources** - The number of sources associated with this UMDS Client.
- **Attribute Table** - A display of the configuration option values for this UMDS Client.
- **Clear Connection "Messages Lost/Discarded" Counts** - Click this button to reset the **Messages Lost/Discarded** values to 0.
- **Clear Connection Statistics** - Click this button to clear the connection statistics for this UMDS Client.
- **Disconnect this Client** - Click this button to disconnect this UMDS Client from the UMDS Server. You can configure this button to be hidden with the `<server>` element `allow-shutdown-via-webmon` attribute.

7.4 Current Server Configuration File

This page displays the umdsd configuration file.

7.5 Dump Current Memory Allocation

This page displays current memory allocation statistics.



7.6 Quit Server

Stop the UMDS Server. This option closes all server connections and terminates the umdsd process. You can configure this button to be hidden with the `<server>` element `allow-shutdown-via-webmon` attribute.

Chapter 8

UMDS Server Configuration

You configure the UMDS Server with four files:

1. The umdsd Configuration File (in xml format). Required and typically specified on the UMDS Server program command line.
2. The UM License File (in Ultra Messaging license format). Optional and specified in the UMDS server configuration file.
3. The UM Configuration File that affects the Ultra Messaging Context running on the same host as the UMDS Server. Does not directly affect the activity between UMDS Server and Client Applications. See the Ultra Messaging Configuration Guide. Optional and specified in the UMDS server configuration file.
4. The Basic Authentication File (in xml format). Optional and specified in the UMDS server configuration file.

8.1 umdsd Configuration File

The following example shows the element structure of the xml configuration file that you use for the umdsd UMDS Server daemon.

```
<?xml version="1.0" encoding="UTF-8"?>
<umds-daemon version="1.0">
  <daemon>
  <log type="file" xml:space="preserve">umdsd.log</log>
  <uid>12345</uid>
     <gid>23456</gid>
  <pidfile xml:space="preserve">example.pid</pidfile>
  <lbm-license-file>example.lic</lbm-license-file>
  <lbm-config>example.lbmcfg</lbm-config>

  <server bind-addr="*:14701" num-workers="3" msg-age-limit="1000"
      msg-q-size-limit="1048576"/>
  <client>
    <server-list client-write="yes">LIST</server-list>
    <server-ka-interval client-write="range"
        min="0" max="2147483648">2000</server-ka-interval>
    <client-ka-threshold client-write="range"
        min="0" max="2147483648">3000</client-ka-threshold>
    <client-ka-interval client-write="range"
        min="0" max="2147483648">10000</client-ka-interval>
```

```
    <server-ka-threshold client-write="yes">11000</server-ka-threshold>
    <server-rcvbuf client-write="yes">0</server-rcvbuf>
    <server-sndbuf client-write="yes">0</server-sndbuf>
    <server-nodelay client-write="yes">0</server-nodelay>
    <client-rcvbuf client-write="yes">0</client-rcvbuf>
    <client-sndbuf client-write="yes">0</client-sndbuf>
    <client-nodelay client-write="yes">0</client-nodelay>
    <server-reconnect client-write="yes">0</server-reconnect>
  </client>
  <authentication>
    <basic xml:space="preserve">BASIC_FILE</basic>
  </authentication>
  <web-monitor>172.16.254.1:8080</web-monitor>
  <monitor object="source" interval="7">
    <transport module="lbm" options="string"/>
    <format module="csv" options="string"/>
    <application-id xml:space="preserve">STRING</application-id>
  </monitor>
    <topics>
      <topic pattern="BEW.xyz.*" type="PCRE">
        <umds-attributes>
          <option type="umds-receiver" name="receiver-queue-type"
              value="topic" />
          <option type="umds-receiver" name="topic-queue-size-limit"
              value="200000" />
        </umds-attributes>
      </topic>
      <topic pattern="BEW.xyz" type="direct">
        <umds-attributes>
          <option type="umds-receiver" name="receiver-queue-type"
              value="default" />
          <option type="umds-receiver" name="topic-queue-size-limit"
              value="200000" />
        </umds-attributes>
      </topic>
    </topics>
  </daemon>
</umds-daemon>
```

8.1.1 UMDS Element "<umds-daemon>"

Container element which holds the UMDS server's configuration. Also defines the version of the configuration format used by the file.

Required.

- **Children:** <daemon>

XML Attributes:

Attribute	Description	Valid Values	Default Value
version	Version number of user's configuration file.	**"1.0"** - Initial version	**"1.0"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
```

```
  ...
</umds-daemon>
```

8.1.2 UMDS Element "<daemon>"

Container element which holds the UMDS server's configuration.

Required.

- **Cardinality:** 0 .. 1
- **Parent:** <umds-daemon>
- **Children:** <log>, <uid>, <gid>, <pidfile>, <lbm-license-file>, <lbm-config>, <server>, <client>, <permissions>, <authentication>, <web-monitor>, <daemon-monitor>, <monitor>, <topics>

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    ...
  </daemon>
</umds-daemon>
```

8.1.3 UMDS Element "<topics>"

Container element for topics that the UMDS server forwards to UMDS client applications. Use this element to apply UMS Configuration Options to individual topics or topic patterns. You can also configure topics and topic patterns to have individual message queues, which can mitigate possible message loss.

Optional. If omitted, topics use the UMDS Server default message queue, and these topics do not use specific UMS configuration options.

- **Cardinality:** 0 .. 1
- **Parent:** <daemon>
- **Children:** <topic>

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <topics>
      ...
    </topics>
    ...
  </daemon>
</umds-daemon>
```

8.1.4 UMDS Element "<topic>"

Holds the configuration for a specific set of topics.

Required, if UMDS Element "<topics>" is present.

Note: if type `"direct"` is used, the pattern must exactly match the entire topic string. For example, `pattern="x.y" type="direct"` will only match the topic `"x.y"`. Topcis `"x.yz"` and `"zx.y"` and will be excluded. However, if the type is a regular expression, no assumption is made regarding the start or end of the topic name. The user is expected to make use of anchor metacharacters "'^'" and "'$'" if needed. For example, `pattern="x" type="PCRE"` will match topics `"x"`, `"xyz"`, `"zyx"`, and `"axe"`. If it is desired to match only topics that start with "x", use `pattern="^x" type="PCRE"`. That will match `"x"` and `"xyz"`, but exclude `"zyx"` and `"axe"`. Also remember that a period (`"."`) is a metacharacter which matches any character, and must be escaped if an actual period is desired. For example, `pattern="^NASD\\." type="PCRE"` will match topics "NASD.a", "NASD.a.b", and "NASD\.", but will exclude "NASDa.b" and "XNASD.a".

- **Parent:** <topics>
- **Children:** <umds-attributes>

XML Attributes:

Attribute	Description	Valid Values	Default Value
pattern	Specifies a pattern used to match topic names. Used to select incoming topics to apply the configuration.	string	(no default; must be specified)
type	Specifies how the pattern should be interpreted.	**`"direct"`** - Exact match for full topic name. **`"PCRE"`** - Regular expression match using PCRE syntax. **`"regexp"`** - (Not recommended.) Regular expression match using POSIX extended regular expressions syntax.	**`"direct"`**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <topics>
      <topic pattern="^NASD\..*" type="PCRE">
        ...
      </topic>
      ...
    </topics>
    ...
  </daemon>
</umds-daemon>
```

8.1.5 UMDS Element "<umds-attributes>"

Container for one or more UMDS Element "<option>" elements which configure the topic(s) matching the parent UMDS Element "<topic>".

Optional, but there is no use case for omission.

- **Cardinality:** 0 .. 1
- **Parent:** <topic>
- **Children:** <option>

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <topics>
      <topic pattern="^NASD\..*" type="PCRE">
        <umds-attributes>
          ...
        </umds-attributes>
      </topic>
      ...
    </topics>
    ...
  </daemon>
</umds-daemon>
```

8.1.6 UMDS Element "<option>"

Specifies a configuration option for the topic(s) matching the parent UMDS Element "<topic>".

Required, if UMDS Element "<umds-attributes>" is present.

Each option supplied is of one of six types divided into two classes:
- Types "lbm-receiver", "lbm-wildcard-receiver", "lbm-source", and "lbm-context". These are used to specify UM configuration options, as described in the UM Configuration Guide.
- Types "umds-receiver" and "umds-source". These are used to specify UMDS-specific options, as described in section UMDS Topic Options.

Note

Although "umds-source" is a valid "type" attribute for the UMDS Element "<option>", there are currently no supported "umds-source" options available. The "umds-source" type is defined in the DTD for future expansion.

- **Parent:** <umds-attributes>

XML Attributes:

Attribute	Description	Valid Values	Default Value
type	Specifies the scope of the configuration option being set.	**"lbm-receiver"** - UM configuration option of "receiver" scope. See Config Guide. **"lbm-wildcard-receiver"** - UM configuration option of "wildcard_receiver" scope. See Config Guide. **"lbm-context"** - UM configuration option of "context" scope. See Config Guide. **"lbm-source"** - UM configuration option of "source" scope. See Config Guide. **"umds-receiver"** - Configuration option specific to UMDS client-side receivers. See UMDS Topic Options. **"umds-source"** - Configuration option specific to UMDS client-side sources. See UMDS Topic Options.	(no default; must be specified)
name	Specifies the name of the configuration option being set.	attr_name	(no default; must be specified)
value	Specifies the desired value for the configuration option being set.	string	(no default; must be specified)

Example:

In this example, the server will configure its wildcard receiver for topics such as "NASD.X" and "NASD.Y" to not stop topic resolution queries using the UM configuration option resolver_query_minimum_duration_wildcard (receiver).

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <topics>
      <topic pattern="^NASD\..*" type="PCRE">
        <umds-attributes>
          <option type="lbm-wildcard-receiver"
              name="resolver_query_minimum_duration"
              value="0"/>
          ...
        </umds-attributes>
      </topic>
      ...
    </topics>
    ...
  </daemon>
</umds-daemon>
```

8.1.7 UMDS Element "<monitor>"

Enables and configures the UM transport statistics monitoring function. Multiple instances of this element are typically supplied to enable monitoring of the different types of UM objects created by the UMDS server during its operation.

Opional. If omitted, no monitoring takes place.

- **Cardinality:** 0 .. 1
- **Parent:** <daemon>
- **Children:** <transport>, <format>, <application-id>

XML Attributes:

Attribute	Description	Valid Values	Default Value
object	The UM object type to monitor.	**"context"** - Monitor UM contexts. **"source"** - Monitor UM sources. **"receiver"** - Monitor UM receivers.	(no default; must be specified)
interval	The time, in seconds, that monitoring statistics are sampled and published.	string	"5"

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <monitor object="context" interval="60">
      ...
    </monitor>
    <monitor object="receiver" interval="30">
      ...
    </monitor>
    ...
  </daemon>
</umds-daemon>
```

8.1.8 UMDS Element "<application-id>"

Identification string, used by monitoring applications to identify the application (where the UMDS server itself is the application in this case).

Optional. If omitted, application ID is not used.

- **Parent:** <monitor>

XML Attributes:

Attribute	Description	Valid Values	Default Value
xml:space	Controls how XML handles spaces in the value string.	**"default"** - Trims most whitespace. **"preserve"** - Retains whitespace as entered.	**"default"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <monitor object="context" interval="60">
      <application-id>
        UMDS server 123
      </application-id>
      ...
    </monitor>
    ...
  </daemon>
</umds-daemon>
```

8.1.9 UMDS Element "<format>"

Configures the data formatting module for publishing monitoring statistics.

Optional. If omitted, uses csv format.

This element is normally not supplied since there is only one supported format, "csv", and UMDS defaults to that format. The element is defined in the DTD for future expansion.

- **Parent:** <monitor>

XML Attributes:

Attribute	Description	Valid Values	Default Value
module	Specifies the formatting module to use. Currently, only "csv" is supported.	**"csv"** - Formatting module which produces delimiter-separated values.	**"csv"**
options	Options string to be passed to the formatting module.	string	(null string)

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <monitor object="context" interval="60">
      <format module="csv"/>
      ...
    </monitor>
    ...
  </daemon>
</umds-daemon>
```

8.1.10 UMDS Element "<transport>"

Configures the data transmission module for publishing monitoring statistics.

Optional. If omitted, uses lbm transport.

- **Parent:** <monitor>

XML Attributes:

Attribute	Description	Valid Values	Default Value
module	Specifies the transmission module to use.	**"lbm"** - Use normal UM source to publish. **"udp"** - Use a simple UDP socket to publish.	**"lbm"**
options	Options string to be passed to the transport module.	string	(null string)

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <monitor object="context" interval="60">
      <transport module="lbm"/>
      ...
    </monitor>
    ...
  </daemon>
</umds-daemon>
```

8.1.11 UMDS Element "<daemon-monitor>"

Configures the Daemon Statistics feature. See Daemon Statistics for information on Daemon Statistics.

Optional. If omitted, Daemon Statistics are not published.

- **Cardinality:** 0 .. 1
- **Parent:** <daemon>
- **Children:** <publishing-interval>, <remote-snapshot-request>, <remote-config-changes-request>, <lbm-config>

XML Attributes:

Attribute	Description	Valid Values	Default Value
topic	Topic name to use for publishing Daemon Statistics.	string	**"umdsd.monitor"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <daemon-monitor>
```

```
      ...
    </daemon-monitor>
    ...
  </daemon>
</umds-daemon>
```

8.1.12 UMDS Element "<lbm-config>"

Specifies the file that contains UM configuration options associated with the parent element.

- **Cardinality:** 0 .. 1
- **Parent:** <daemon>, <daemon-monitor>

XML Attributes:

Attribute	Description	Valid Values	Default Value
xml:space	Controls how XML handles spaces in the value string.	**"default"** - Trims most whitespace. **"preserve"** - Retains whitespace as entered.	**"default"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <daemon-monitor>
      <lbm-config>/
        etc/umds_dmon.cfg
      </lbm-config>
      ...
    </daemon-monitor>
    ...
  </daemon>
</umds-daemon>
```

8.1.13 UMDS Element "<remote-config-changes-request>"

Configures whether the UMDS server will respond to monitoring apps requests to change the rate at which Daemon Statistics messages are published.

Optional. If omitted, change requests will be ignored.

See Daemon Statistics for information on Daemon Statistics.

- **Cardinality:** 0 .. 1
- **Parent:** <daemon-monitor>

XML Attributes:

Attribute	Description	Valid Values	Default Value
allow	Enable or disable change requests.	**"0"** - UMDS will ignore change requests. **"1"** - UMDS will respond to change requests.	**"0"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <daemon-monitor>
      <remote-config-changes-request allow="1"/>
      ...
    </daemon-monitor>
    ...
  </daemon>
</umds-daemon>
```

8.1.14 UMDS Element "<remote-snapshot-request>"

Configures whether the UMDS server will respond to monitoring apps requests to send on-demand snapshots of daemon statistics.

Optional. If omitted, snapshot requests will be ignored.

See Daemon Statistics for information on Daemon Statistics.

- **Cardinality:** 0 .. 1
- **Parent:** <daemon-monitor>

XML Attributes:

Attribute	Description	Valid Values	Default Value
allow	Enable or disable snapshot requests.	**"0"** - UMDS will ignore snapshot requests. **"1"** - UMDS will respond to snapshot requests.	**"0"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <daemon-monitor>
      <remote-snapshot-request allow="1"/>
      ...
    </daemon-monitor>
    ...
  </daemon>
</umds-daemon>
```

8.1.15 UMDS Element "<publishing-interval>"

Configures the rate at which Daemon Statistics messages are published.

Optional. If omitted, default publishing intervals will be used (see children elements for defaults).

See Daemon Statistics for information on Daemon Statistics.

- **Cardinality:** 0 .. 1
- **Parent:** <daemon-monitor>
- **Children:** <group>

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <daemon-monitor>
      <publishing-interval>
        ...
      </publishing-interval>
      ...
    </daemon-monitor>
    ...
  </daemon>
</umds-daemon>
```

8.1.16 UMDS Element "<group>"

Configures the rate at which one particular grouping of Daemon Statistics messages are published.

Optional. If omitted, the group publishes at its default rate.

See Daemon Statistics for information on Daemon Statistics.

- **Parent:** <publishing-interval>

XML Attributes:

Attribute	Description	Valid Values	Default Value
name	Statistics group to set.	**"default"** - Sets interval for those statistics groups which are not explicity set by other <group> elements. **"malloc-info-ivl"** - Sets interval for message type **umdsd_dstat_mallinfo_msg_stct**. **"worker-details-ivl"** - Sets interval for worker detail reports, which consists of a set of **UMDS_DSTATTYPE_WORKER** messages, one per worker, and a set of **UMDS_DSTATTYPE_CLIENTPERMS**, **UMDS_DSTATTYPE_PERTOPIC**, **UMDS_DSTATTYPE_TOPICTOTALS**, **UMDS_DSTATTYPE_RECEIVER**, **UMDS_DSTATTYPE_SOURCE**, and **UMDS_DSTATTYPE_CLIENTATTRS** messages. **"worker-summary-ivl"** - Sets interval for worker summery reports, which consists of a set of **UMDS_DSTATTYPE_WORKER** messages, one per worker, and a set of **UMDS_DSTATTYPE_CONNSUMMARY** messages, one for each connection.	(no default; must be specified)
ivl	Time, in seconds, between publishing the statistics group.	string	(no default; must be specified)

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <daemon-monitor>
      <publishing-interval>
        <group name="default" ivl="5">
        <group name="worker-details-ivl" ivl="30">
        ...
      </publishing-interval>
      ...
    </daemon-monitor>
    ...
  </daemon>
</umds-daemon>
```

8.1.17 UMDS Element "<web-monitor>"

Enables the web-based server monitoring and control functions, and configures the IP address and port to listen on. Value is in IP:PORT format. An IP value of * indicates any interface (for example: `*:8080`).

Optional. If omitted, the web monitor is disabled.

- **Cardinality:** 0 .. 1
- **Parent:** <daemon>

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <web-monitor>
      172.16.254.1:8080
    </web-monitor>
    ...
  </daemon>
</umds-daemon>
```

8.1.18 UMDS Element "<authentication>"

Determines if UMDS clients use authentication. If empty (<authentication/>), no authentication occurs.

Required.

- **Parent:** <daemon>
- **Children:** <none>, <basic>

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <authentication>
      ...
    </authentication>
    ...
  </daemon>
</umds-daemon>
```

8.1.19 UMDS Element "<basic>"

Enables basic authentication of the client with the server, and supplies the name of the authentication file. See Basic Authentication File.

Optional.

- **Cardinality:** 0 .. 1
- **Parent:** <authentication>

XML Attributes:

Attribute	Description	Valid Values	Default Value
xml:space	Controls how XML handles spaces in the value string.	**"default"** - Trims most whitespace. **"preserve"** - Retains whitespace as entered.	**"default"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <authentication>
      <basic>/
        etc/umds_basic_auth.txt
      </basic>
    </authentication>
    ...
  </daemon>
</umds-daemon>
```

8.1.20 UMDS Element "<none>"

No authentication is done.

Optional.

- **Cardinality:** 0 .. 1
- **Parent:** <authentication>

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <authentication>
      <none/>
    </authentication>
    ...
  </daemon>
</umds-daemon>
```

8.1.21 UMDS Element "<permissions>"

This option is deprecated. Setting values has no effect.

- **Cardinality:** 0 .. 1
- **Parent:** <daemon>
- **Children:** <can-send>, <can-stream>, <can-reqresp>

Do not use.

8.1.22 UMDS Element "<can-reqresp>"

This option is deprecated. Setting values has no effect.

- **Cardinality:** 0 .. 1
- **Parent:** <permissions>

Do not use.

8.1.23 UMDS Element "<can-stream>"

This option is deprecated. Setting values has no effect.

- **Cardinality:** 0 .. 1
- **Parent:** <permissions>

Do not use.

8.1.24 UMDS Element "<can-send>"

This option is deprecated. Setting values has no effect.

- **Cardinality:** 0 .. 1
- **Parent:** <permissions>

Do not use.

8.1.25 UMDS Element "<client>"

Sets optional client operating parameters.

Required.

Each client child element can be configured to be overwritten by a client application with the client-write attribute. Some client elements (keep-alive, receive and send socket buffers) can also restrict the ability of a client application to overwrite a client element by specifying a range of acceptable values from the client application.

- **Parent:** <daemon>
- **Children:** <server-list>, <server-ka-interval>, <client-ka-threshold>, <client-ka-interval>, <server-ka-threshold>, <server-rcvbuf>, <server-sndbuf>, <server-nodelay>, <client-rcvbuf>, <client-sndbuf>, <client-nodelay>, <server-reconnect>

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <client>
      ...
    </client>
    ...
  </daemon>
</umds-daemon>
```

8.1.26 UMDS Element "<server-reconnect>"

Indicates whether the client should attempt to reconnect to the server if the connection fails.

Optional. If omitted, reconnection is enabled.

Value of 1 turns on reconnect. The client then tries to reconnect to a server in the UMDS Element "<server-list>". A value of 0 prevents the client from reconnecting to any server after connection failure.

- **Parent:** <client>

XML Attributes:

Attribute	Description	Valid Values	Default Value
client-write	Specify if the client application is allowed to override the server setting for this element.	**"yes"** - Client is allowed to override. **"no"** - Client is not allowed to override.	**"yes"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <client>
      <server-reconnect client-write="no">
        1
      </server-reconnect>
      ...
    </client>
    ...
  </daemon>
</umds-daemon>
```

8.1.27 UMDS Element "<client-nodelay>"

Specify if the UMDS client's TCP connection to the server should set the TCP_NODELAY socket option, which disables Nagle's algorithm.

Optional. If omitted, TCP_NODELAY is not set (Nagle's algorithm is retained).

This option should be set if the lowest-possible latency is desired. Leaving it unset permits more-efficient use of network resources.

- **Parent:** <client>

XML Attributes:

Attribute	Description	Valid Values	Default Value
client-write	Specify if the client application is allowed to override the client setting for this element.	**"yes"** - Client is allowed to override. **"no"** - Client is not allowed to override.	**"yes"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <client>
      <client-nodelay client-write="yes">
        1
      </client-nodelay>
      ...
    </client>
    ...
  </daemon>
</umds-daemon>
```

8.1.28 UMDS Element "<client-sndbuf>"

Specify the UMDS client's TCP's SO_SNDBUF (send-side socket buffer size) in its connection to the server.

Optional. If omitted, client's operating system sets it.

It is usually recommended not to set this option.

- **Parent:** <client>

XML Attributes:

Attribute	Description	Valid Values	Default Value
client-write	Specify if the client application is allowed to override the server setting for this element.	**"yes"** - Client is allowed to override. **"no"** - Client is not allowed to override. **"range"** - Client is allowed to override within a given range of values. If `range` attribute is used, must also supply `min` and `max` attributes.	**"yes"**
min	Client's override must be at least this value.	string	**"0"**
max	Client's override must be no more than this value.	string	**"2147483648"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <client>
      <client-sndbuf client-write="yes">
```

```
        524288
      </client-sndbuf>
      ...
    </client>
    ...
  </daemon>
</umds-daemon>
```

8.1.29 UMDS Element "<client-rcvbuf>"

Specify the UMDS client's TCP's SO_RCVBUF (receive-side socket buffer size) in its connection to the server.

Optional. If omitted, client's operating system sets it.

It is usually recommended not to set this option.

- **Parent:** <client>

XML Attributes:

Attribute	Description	Valid Values	Default Value
client-write	Specify if the client application is allowed to override the server setting for this element.	**"yes"** - Client is allowed to override. **"no"** - Client is not allowed to override. **"range"** - Client is allowed to override within a given range of values. If `range` attribute is used, must also supply `min` and `max` attributes.	**"yes"**
min	Client's override must be at least this value.	string	**"0"**
max	Client's override must be no more than this value.	string	**"2147483648"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <client>
      <client-rcvbuf client-write="yes">
        524288
      </client-rcvbuf>
      ...
    </client>
    ...
  </daemon>
</umds-daemon>
```

8.1.30 UMDS Element "<server-nodelay>"

Specify if the UMDS server's TCP connection to the client should set the TCP_NODELAY socket option, which disables Nagle's algorithm.

Optional. If omitted, TCP_NODELAY is not set (Nagle's algorithm is retained).

This option should be set if the lowest-possible latency is desired. Leaving it unset permits more-efficient use of network resources.

- **Parent:** <client>

XML Attributes:

Attribute	Description	Valid Values	Default Value
client-write	Specify if the client application is allowed to override the server setting for this element.	**"yes"** - Client is allowed to override. **"no"** - Client is not allowed to override.	**"yes"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <client>
      <server-nodelay client-write="yes">
        1
      </server-nodelay>
      ...
    </client>
    ...
  </daemon>
</umds-daemon>
```

8.1.31 UMDS Element "<server-sndbuf>"

Specify the UMDS server's TCP's SO_SNDBUF (send-side socket buffer size) in its connection to the client.

Optional. If omitted, server's operating system sets it.

It is usually recommended not to set this option.

- **Parent:** <client>

XML Attributes:

Attribute	Description	Valid Values	Default Value
client-write	Specify if the client application is allowed to override the server setting for this element.	**"yes"** - Client is allowed to override. **"no"** - Client is not allowed to override. **"range"** - Client is allowed to override within a given range of values. If `range` attribute is used, must also supply `min` and `max` attributes.	**"yes"**

Attribute	Description	Valid Values	Default Value
min	Client's override must be at least this value.	string	**"0"**
max	Client's override must be no more than this value.	string	**"2147483648"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <client>
      <server-sndbuf client-write="yes">
        524288
      </server-sndbuf>
      ...
    </client>
    ...
  </daemon>
</umds-daemon>
```

8.1.32 UMDS Element "<server-rcvbuf>"

Specify the UMDS server's TCP's SO_RCVBUF (receive-side socket buffer size) in its connection to the client.

Optional. If omitted, server's operating system sets it.

It is usually recommended not to set this option.

- **Parent:** <client>

XML Attributes:

Attribute	Description	Valid Values	Default Value
client-write	Specify if the client application is allowed to override the server setting for this element.	**"yes"** - Client is allowed to override. **"no"** - Client is not allowed to override. **"range"** - Client is allowed to override within a given range of values. If `range` attribute is used, must also supply `min` and `max` attributes.	**"yes"**
min	Client's override must be at least this value.	string	**"0"**
max	Client's override must be no more than this value.	string	**"2147483648"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <client>
      <server-rcvbuf client-write="yes">
        524288
```

```
      </server-rcvbuf>
      ...
    </client>
    ...
  </daemon>
</umds-daemon>
```

8.1.33 UMDS Element "<server-ka-threshold>"

Number of milliseconds of silence to wait before connection is declared dead.

Optional. If omitted, defaults to 11000

n the absence of message or keep-alive traffic for the threshold, the server declares the connection dead. This value should be at least one second (1000 ms) greater than the UMDS Element "<client-ka-interval>". See also Keep Alive Timers During Idle Periods.

- **Parent:** <client>

XML Attributes:

Attribute	Description	Valid Values	Default Value
client-write	Specify if the client application is allowed to override the server setting for this element.	**"yes"** - Client is allowed to override. **"no"** - Client is not allowed to override. **"range"** - Client is allowed to override within a given range of values. If `range` attribute is used, must also supply `min` and `max` attributes.	**"yes"**
min	Client's override must be at least this value.	string	**"0"**
max	Client's override must be no more than this value.	string	**"2147483648"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <client>
      <server-ka-threshold client-write="yes">
        12000
      </server-ka-threshold>
      ...
    </client>
    ...
  </daemon>
</umds-daemon>
```

8.1.34 UMDS Element "<client-ka-interval>"

Milliseconds between keep-alive messages from the client.

Optional. If omitted, defaults to 10000

In the absence of message traffic, the client sends keep-alive messages at this interval. This value should be at least one second (1000 ms) less than UMDS Element "<server-ka-threshold>" Element. See also Keep Alive Timers During Idle Periods.

- **Parent:** <client>

XML Attributes:

Attribute	Description	Valid Values	Default Value
client-write	Specify if the client application is allowed to override the server setting for this element.	**"yes"** - Client is allowed to override. **"no"** - Client is not allowed to override. **"range"** - Client is allowed to override within a given range of values. If `range` attribute is used, must also supply `min` and `max` attributes.	**"yes"**
min	Client's override must be at least this value.	string	**"0"**
max	Client's override must be no more than this value.	string	**"2147483648"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <client>
      <client-ka-interval client-write="yes">
        8000
      </client-ka-interval>
      ...
    </client>
    ...
  </daemon>
</umds-daemon>
```

8.1.35 UMDS Element "<client-ka-threshold>"

Number of milliseconds of silence to wait before connection is declared dead.

Optional. If omitted, defaults to 3000

In the absence of message or keep-alive traffic for the threshold, the client declares the connection dead and attempts to reconnect. This value should be at least one second (1000 ms) greater than the UMDS Element "<server-ka-interval>". See also Keep Alive Timers During Idle Periods.

- **Parent:** <client>

XML Attributes:

Attribute	Description	Valid Values	Default Value
client-write	Specify if the client application is allowed to override the server setting for this element.	**"yes"** - Client is allowed to override. **"no"** - Client is not allowed to override. **"range"** - Client is allowed to override within a given range of values. If `range` attribute is used, must also supply `min` and `max` attributes.	**"yes"**
min	Client's override must be at least this value.	string	**"0"**
max	Client's override must be no more than this value.	string	**"2147483648"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <client>
      <client-ka-threshold client-write="yes">
        4000
      </client-ka-threshold>
      ...
    </client>
    ...
  </daemon>
</umds-daemon>
```

8.1.36 UMDS Element "<server-ka-interval>"

Milliseconds between keep-alive messages from the client.

Optional. If omitted, defaults to 2000

In the absence of message traffic, the client sends keep-alive messages at this interval. This value should be at least one second (1000 ms) less than UMDS Element "<client-ka-threshold>" Element. See also Keep Alive Timers During Idle Periods.

- **Parent:** <client>

XML Attributes:

Attribute	Description	Valid Values	Default Value
client-write	Specify if the client application is allowed to override the server setting for this element.	**"yes"** - Client is allowed to override. **"no"** - Client is not allowed to override. **"range"** - Client is allowed to override within a given range of values. If `range` attribute is used, must also supply `min` and `max` attributes.	**"yes"**
min	Client's override must be at least this value.	string	**"0"**

Attribute	Description	Valid Values	Default Value
max	Client's override must be no more than this value.	string	**"2147483648"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <client>
      <server-ka-interval client-write="yes">
        3000
      </server-ka-interval>
      ...
    </client>
    ...
  </daemon>
</umds-daemon>
```

8.1.37 UMDS Element "<server-list>"

Comma-separated list of UMDS server addresses (IP:Port) that the client should use.

Optional. If omitted, the client uses the server list in its own configuration.

This allows a server to be used as a "redirection" service. I.e. a client can be initially configured to connect to UMDS server A, which re-directs the client to the production server.

- **Parent:** <client>

XML Attributes:

Attribute	Description	Valid Values	Default Value
client-write	Specify if the client application is allowed to override the server setting for this element.	**"yes"** - Client is allowed to override. **"no"** - Client is not allowed to override.	**"yes"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <client>
      <server-list client-write="no">
        10.11.12.13:14701,10.11.12.14:14701
      </server-list>
      ...
    </client>
    ...
  </daemon>
</umds-daemon>
```

8.1.38 UMDS Element "<server>"

Configure the operating parameters UMDS server with the attributes supplied.

Required.

- **Parent:** <daemon>

XML Attributes:

Attribute	Description	Valid Values	Default Value
bind-addr	IP/Port that the UMDS server listens on for client connections.	string	"*:14701"
num-workers	Number of workers to create to service clients. See Worker Configuration Guidelines.	string	"3"
msg-age-limit	Maximum age in milliseconds before umdsd deletes the oldest messages when it reaches this limit. To remove this limit, set this attribute to zero.	string	"1000" (1 sec)
msg-q-size-limit	Maximum size in bytes of message queue. Umdsd deletes the oldest messages when it reaches this limit. To remove the limit, set this attribute to zero.	string	"1048576"
allow-shutdown-via-webmon	Control if the UMDS web monitor offers a "Quit Server" button in the main page, and a "Disconnect this Client" button in the Client Details page. Value "0" disables the buttons. Value "1" enables the buttons.	string	"0" (disable)
request-timeout	Duration for each request to remain open (accepting responses).	string	"10,000" (10 sec)

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <server bind-addr="*.14701"
        num-workers="3"
        msg-age-limit="1000"
        msg-q-size-limit="1048576"
        request-timeout="10000"
        allow-shutdown-via-webmon="0"
    \>
    ...
  </daemon>
</umds-daemon>
```

8.1.39 UMDS Element "<lbm-license-file>"

Specifies the pathname where the the user has placed their UM license file.

Optional. if omitted, the license must be supplied via an environment variable.

- **Cardinality:** 0 .. 1
- **Parent:** <daemon>

XML Attributes:

Attribute	Description	Valid Values	Default Value
xml:space	Controls how XML handles spaces in the value string.	**"default"** - Trims most whitespace. **"preserve"** - Retains whitespace as entered.	**"default"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <lbm-licnese-file>/etc/umdsd.lic</lbm-licnese-file>
    ...
  </daemon>
</umds-daemon>
```

8.1.40 UMDS Element "<pidfile>"

Specifies the pathname where the UMDS server stores its Process ID (PID).

Optional. If omitted, the server does not store its PID in a file.

- **Cardinality:** 0 .. 1
- **Parent:** <daemon>

XML Attributes:

Attribute	Description	Valid Values	Default Value
xml:space	Controls how XML handles spaces in the value string.	**"default"** - Trims most whitespace. **"preserve"** - Retains whitespace as entered.	**"default"**

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <pidfile>/var/run/umdsd.pid</pidfile>
    ...
  </daemon>
</umds-daemon>
```

8.1.41 UMDS Element "<gid>"

Specifies the Group ID (GID) for the server process (if run as root).

Optional. If omitted, the GID of the parent process is inherited.

- **Cardinality:** 0 .. 1
- **Parent:** <daemon>

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <gid>5555</gid>
    ...
  </daemon>
</umds-daemon>
```

8.1.42 UMDS Element "<uid>"

Specifies the User ID (UID) for the server process (if run as root).

Optional. If omitted, the UID of the parent process is inherited.

- **Cardinality:** 0 .. 1
- **Parent:** <daemon>

Example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <uid>5555</uid>
    ...
  </daemon>
</umds-daemon>
```

8.1.43 UMDS Element "<log>"

Configures UMDS server's logging behavior. The value contained within the `<log>...</log>` is a file name, but is only used if the "type" attribute is set to "file".

Optional. If omitted, logs are written to Standard Out.

- **Cardinality:** 0 .. 1
- **Parent:** <daemon>

XML Attributes:

Attribute	Description	Valid Values	Default Value
type	Specifies the method of logging.	**"file"** - Logs to a text file. **"syslog"** - Logs to the Unix SYSLOG facility. **"console"** - Logs to standard output.	**"console"**
xml:space	Controls how XML handles spaces in the value string.	**"default"** - Trims most whitespace. **"preserve"** - Retains whitespace as entered.	**"default"**

Example 1: (write log messages to Standard Out)

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <log type="console"/>
    ...
  </daemon>
</umds-daemon>
```

Example 2: (write log messages to "umds.log" file)

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-daemon version="1.0">
  <daemon>
    <log type="file" frequency="daily">umds.log</log>
    ...
  </daemon>
</umds-daemon>
```

8.1.44 UMDS Topic Options

The UMDS server's configuration file can contain one or more UMDS Element "<option>" elements to allow the user to have topic-specific configurations.

Note

Although "umds-source" is a valid type attribute for the UMDS Element "<option>", there are currently no supported "umds-source" options available. The "umds-source" type is defined in the DTD for future expansion.

Available options for type "umds-receiver":

Option	Description	Default Value
receiver-queue-type	topic or default. The value topic creates a queue for the topic. The value default places messages for the topic in the client's default queue.	topic
topic-queue-size-limit	Maximum size in bytes of topic message queue. UMDS Server deletes oldest messages when the queue reaches this configured limit. A value of 0 (zero) means no limit.	1048576

Option	Description	Default Value
use-late-join	Controls whether UMDS receivers use Late Join for this topic. To disable Late Join for UMDS receivers, set this option to 0 (Off).	1 (On)
message-cache-size	Specifies the size, in number of messages, of the Late Join message cache for this topic.	1
ignore-unique-receiver-attributes	Indicates which set of lbm-receiver options a UMDS Client application uses when it discovers a source/topic that matches a wildcard receiver pattern. Both the wildcard pattern and the individual topic might have lbm-receiver options. The default value of 1 ignores the lbm-receiver individual topic options, and instead uses the options configured for the wildcard pattern. This value can provide more efficient control over options for all receivers. Setting this option to 0 (zero) instructs the UMDS Client application to use lbm-receiver options configured for the individual topic.	1 (Ignore)

8.2 UM License File

The Ultra Messaging license file contains the Ultra Messaging license key. The file name is specified with the UMDS Element "<lbm-license-file>". If omitted from the umdsd Configuration File, umdsd looks for the environment variables, LBM_LICENSE_INFO or LBM_LICENSE_FILENAME.

8.3 UM Configuration File

This file is optional. You specify this file in the umdsd Configuration File with the UMDS Element "<lbm-config>". The Ultra Messaging Configuration File contains configuration options for the UMDS Server's Ultra Messaging context. If omitted, Ultra Messaging uses the factory default values. See the Ultra Messaging Configuration Guide for complete details.

Do not include the following options in the Ultra Messaging configuration file when you use it with the umdsd daemon. UMDS ignores these options if they appear in the file.

- operational_mode (context)
- mim_ordered_delivery (context)
- ordered_delivery (receiver)
- use_transport_thread (receiver)
- use_late_join (receiver)
- ume_use_store (receiver)

- umq_queue_participation (receiver)
- umq_queue_name (source)

8.4 Basic Authentication File

The Basic Authentication File specifies user and application records which contain client operational parameters. You optionally specify this file in the umdsd Configuration File. If the Basic Authentication File is omitted, umdsd does not perform Basic Authentication.

The format of the Basic Authentication File is xml. The <client> child elements of both the Application and User elements are identical to those specified for the umdsd Configuration File. Values in a Basic Authentication File override those in the umdsd Configuration File. The following example shows a Basic Authentication File:

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-authentication-basic="1.0">
  <application name="NAME">
    <client>
      ...
    </client>
  </application>
  <user name="NAME" password="PASW">
    <client>
      ...
    </client>
  </user>
</umds-authentication-basic>
```

8.4.1 UMDS application Element

Optional. Name of application to associate client operating parameters.

Attribute	Values
name	Name is limited to 31 characters.

8.4.2 UMDS user Element

Optional. Name and password for an application user. Client operating parameters can be set for individual users.

Attribute	Values
name	Name is limited to 31 characters.
password	Password used to authenticate user. Password is limited to 31 characters and is stored in plain text.

8.5 UMDS Configuration DTD

Here is the umdsd server's configuration DTD, used to validate the user's XML file:

```
<!ELEMENT umds-daemon (daemon?)>
<!ATTLIST umds-daemon
          version (1.0) #REQUIRED
>
<!ELEMENT daemon (log?, uid?, gid?, pidfile?, lbm-license-file?, lbm-config?,
    server, client, permissions?, authentication, web-monitor?, daemon-monitor?,
    monitor?, topics?)>
<!ELEMENT log ( #PCDATA )>
<!ATTLIST log
          type (file | syslog | console) "console"
           xml:space (default | preserve) "default"
>
<!ELEMENT uid ( #PCDATA )>
<!ELEMENT gid ( #PCDATA )>
<!ELEMENT pidfile ( #PCDATA )>
<!ATTLIST pidfile
           xml:space (default | preserve) "default"
>
<!ELEMENT lbm-license-file ( #PCDATA )>
<!ATTLIST lbm-license-file
           xml:space (default | preserve) "default"
>
<!ELEMENT lbm-config ( #PCDATA )>
<!ATTLIST lbm-config
           xml:space (default | preserve) "default"
>
<!ELEMENT server EMPTY>
<!ATTLIST server
          bind-addr CDATA "*:14701"
          num-workers CDATA "3"
          msg-age-limit CDATA "1000"
          msg-q-size-limit CDATA "(1024*1024)"
          allow-shutdown-via-webmon CDATA "0"
          request-timeout CDATA "10000"
>
<!ELEMENT client (server-list | server-ka-interval | client-ka-threshold |
    client-ka-interval | server-ka-threshold | server-rcvbuf | server-sndbuf |
    server-nodelay | client-rcvbuf | client-sndbuf | client-nodelay |
    server-reconnect )* >
<!ELEMENT server-list ( #PCDATA ) >
<!ATTLIST server-list
          client-write (yes | no) "yes"
>
<!ELEMENT server-ka-interval ( #PCDATA ) >
<!ATTLIST server-ka-interval
          client-write (yes | no | range) "yes"
          min CDATA "0"
          max CDATA "2147483648"
>
<!ELEMENT client-ka-threshold ( #PCDATA ) >
<!ATTLIST client-ka-threshold
          client-write (yes | no | range) "yes"
          min CDATA "0"
          max CDATA "2147483648"
>
<!ELEMENT client-ka-interval ( #PCDATA ) >
```

```
<!ATTLIST client-ka-interval
          client-write (yes | no | range) "yes"
          min CDATA "0"
          max CDATA "2147483648"
>
<!ELEMENT server-ka-threshold ( #PCDATA ) >
<!ATTLIST server-ka-threshold
          client-write (yes | no | range) "yes"
          min CDATA "0"
          max CDATA "2147483648"
>
<!ELEMENT server-rcvbuf ( #PCDATA ) >
<!ATTLIST server-rcvbuf
          client-write (yes | no | range) "yes"
          min CDATA "0"
          max CDATA "2147483648"
>
<!ELEMENT server-sndbuf ( #PCDATA ) >
<!ATTLIST server-sndbuf
          client-write (yes | no | range) "yes"
          min CDATA "0"
          max CDATA "2147483648"
>
<!ELEMENT server-nodelay ( #PCDATA ) >
<!ATTLIST server-nodelay
          client-write (yes | no) "yes"
>
<!ELEMENT client-rcvbuf ( #PCDATA ) >
<!ATTLIST client-rcvbuf
          client-write (yes | no | range) "yes"
          min CDATA "0"
          max CDATA "2147483648"
>
<!ELEMENT client-sndbuf ( #PCDATA ) >
<!ATTLIST client-sndbuf
          client-write (yes | no | range) "yes"
          min CDATA "0"
          max CDATA "2147483648"
>
<!ELEMENT client-nodelay ( #PCDATA ) >
<!ATTLIST client-nodelay
          client-write (yes | no) "yes"
>
<!ELEMENT server-reconnect ( #PCDATA ) >
<!ATTLIST server-reconnect
          client-write (yes | no) "yes"
>
<!ELEMENT permissions (can-send?, can-stream?, can-reqresp? ) >
<!ELEMENT can-send ( #PCDATA ) >
<!ELEMENT can-stream ( #PCDATA ) >
<!ELEMENT can-reqresp ( #PCDATA ) >
<!ELEMENT authentication (none?, basic?)>
<!ELEMENT none EMPTY>
<!ELEMENT basic ( #PCDATA )>
<!ATTLIST basic
           xml:space (default | preserve) "default"
>
<!ELEMENT web-monitor ( #PCDATA )>
<!ELEMENT monitor (transport | format | application-id)*>
<!ATTLIST monitor
          object (context | source | receiver) "context"
          interval CDATA "5"
>
```

```
<!ELEMENT topics (topic+)>
<!ELEMENT topic (umds-attributes?)>
<!ATTLIST topic pattern CDATA #REQUIRED
                type (direct | PCRE | regexp) #IMPLIED
>
<!ELEMENT umds-attributes (option+)>
<!ELEMENT option EMPTY>
<!ATTLIST option type (lbm-receiver | lbm-wildcard-receiver | lbm-context |
    lbm-source | umds-receiver | umds-source) #IMPLIED>
<!ATTLIST option name CDATA #REQUIRED>
<!ATTLIST option value CDATA #REQUIRED>
<!ELEMENT transport EMPTY>
<!ATTLIST transport
          module (lbm | udp) "lbm"
          options CDATA #IMPLIED
>
<!ELEMENT format EMPTY>
<!ATTLIST format
          module (csv) "csv"
          options CDATA #IMPLIED
>
<!ELEMENT application-id ( #PCDATA )>
<!ATTLIST application-id
           xml:space (default | preserve) "default"
>
<!ELEMENT daemon-monitor (publishing-interval?, remote-snapshot-request?,
    remote-config-changes-request?, lbm-config?)>
<!ATTLIST daemon-monitor topic CDATA "umdsd.monitor">
<!ELEMENT publishing-interval (group+)>
<!ELEMENT group EMPTY>
<!ATTLIST group name (default | malloc-info-ivl | worker-details-ivl |
    worker-summary-ivl ) #REQUIRED>
<!ATTLIST group ivl CDATA #REQUIRED>
<!ELEMENT remote-snapshot-request EMPTY>
<!ATTLIST remote-snapshot-request allow (0 | 1) "0">
<!ELEMENT remote-config-changes-request EMPTY>
<!ATTLIST remote-config-changes-request allow (0 | 1) "0">
```

8.6 Example UMDS Configuration Files

This section presents the following example UMDS Server configuration files, which contain comments that explain sections of the xml files.

8.6.1 Minimum Configuration File

The following sample UMDS Server configuration file contains the minimum configuration information required to start the UMDS Server `umdsd` daemon. The daemon uses default values for the empty elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<umds-daemon version="1.0">
<daemon>
  <!-- This line is required. Defaults are: bind-addr="*:14701" num-workers="3" -->
  <!-- msg-age-limit= "1000" msg-q-size-limit="1048576" -->
  <server/>
```

```
  <!-- This line is required. -->
  <client/>

  <!-- This line is required. Default is "none" -->
  <authentication/>
</daemon>
</umds-daemon>
```

8.6.2 Typical Configuration File

The second example is a typical starting point for most users. It relies on `example.lic` (which contains the Ultra Messaging license information and is not shown), Sample UM Configuration File, and Sample Authentication File. The UMDS Server generates the file named `example.log`.

```
<?xml version="1.0" encoding="UTF-8"?>
<umds-daemon version="1.0">
<daemon>
  <!-- Write log messages to a file -->
  <log type="file" xml:space="preserve">example.log</log>

  <!-- License contains your Informatica license key -->
  <!-- <lbm-license-file>example.lic</lbm-license-file> -->

  <!-- Override LBM configuration parameters. -->
  <lbm-config>example.lbmcfg</lbm-config>

  <!-- Select a unique port and set the other parameters. -->
  <server bind-addr="*:17500" num-workers="2" msg-age-limit= "7000"
      msg-q-size-limit="4000000" />

  <!-- Select reasonable defaults and prevent users from overriding.  -->
  <!-- (Specific users can override via the authentication file.)  -->
  <client>
    <!-- Pick default server-list settings -->
    <server-list/>

    <!-- Server sends a keep alive msg every 5 seconds -->
    <server-ka-interval client-write="no">5000</server-ka-interval>

    <!-- Client times-out if no keep alive received in 12 seconds -->
    <client-ka-threshold client-write="no">12000</client-ka-threshold>

    <!-- Client sends a keep alive msg every 9 seconds -->
    <client-ka-interval client-write="no">9000</client-ka-interval>

    <!-- Server times-out if no keep alive received in 20 seconds -->
    <server-ka-threshold client-write="no">20000</server-ka-threshold>

    <!-- Set buffers to 1/2 megabyte -->
    <server-rcvbuf client-write="no">524288</server-rcvbuf>
    <server-sndbuf client-write="no">524288</server-sndbuf>
    <!-- Do not change TCP nodelay from OS default -->
    <server-nodelay client-write="no">0</server-nodelay>
```

```
    <!-- Set client side to same as server side -->
    <client-rcvbuf client-write="no">524288</client-rcvbuf>
    <client-sndbuf client-write="no">524288</client-sndbuf>
    <client-nodelay client-write="no">0</client-nodelay>

    <!-- Do not allow client to automatically reconnect to the server -->
    <server-reconnect client-write="no">0</server-reconnect>
  </client>

  <!-- Block unathorized users. -->
  <authentication>
    <basic>authentication.xml</basic>
  </authentication>

  <!-- Web monitoring can be a valuable feature -->
  <web-monitor>*:8080</web-monitor>
</daemon>
</umds-daemon>
```

8.6.3 Complete Configuration File

This sample UMDS Server configuration file contains values for all configuration elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<umds-daemon version="1.0">
<daemon>
  <!-- This line is optional. Default sends log information to the screen -->
  <log type="file" xml:space="preserve">example.log</log>

  <!-- These lines are optional. To set UID and GID, you need to be root -->
<!--
  <uid>12345</uid>
  <gid>23456</gid>
-->

  <!-- This line is optional. If omitted, no PID file is created -->
  <pidfile xml:space="preserve">example.pid</pidfile>

  <!-- This line is optional; it is not the only way to designate a license. -->
  <!-- <lbm-license-file>example.lic</lbm-license-file> -->

  <!-- Sets Ultra Messaging configuration options for contexts used by UMDS. -->
  <lbm-config>example.lbmcfg</lbm-config>

  <!-- This line is required.  -->
  <server/>

  <!-- All client attributes and their defaults appear below. -->
  <!-- Attributes must be listed in this order -->
  <client>
    <server-list client-write="yes">0.0.0.0:14701</server-list>

    <!-- Server sends keep alive at interval...client times out at threshold -->
    <server-ka-interval client-write="range"
        min="0" max="2147483648">2000</server-ka-interval>
    <client-ka-threshold client-write="range"
        min="0" max="2147483648">3000</client-ka-threshold>
```

```
  <!-- Client sends keep alive at interval...server times out at threshold -->
  <client-ka-interval client-write="range"
      min="0" max="2147483648">10000</client-ka-interval>
  <server-ka-threshold client-write="range"
      min="0" max="2147483648">11000</server-ka-threshold>

  <!-- Zero means use the OS default settings -->
  <server-rcvbuf client-write="range"
      min="0" max="2147483648">0</server-rcvbuf>
  <server-sndbuf client-write="range"
      min="0" max="2147483648">0</server-sndbuf>

  <!-- Zero means don't enable TCP_NODELAY (improves network efficiency) -->
  <server-nodelay client-write="yes">0</server-nodelay>

  <!-- Zero means use the OS default settings -->
  <client-rcvbuf client-write="range"
      min="0" max="2147483648">0</client-rcvbuf>
  <client-sndbuf client-write="range"
      min="0" max="2147483648">0</client-sndbuf>

  <!-- Zero means don't enable TCP_NODELAY -->
  <client-nodelay client-write="yes">0</client-nodelay>

  <!-- Zero indicates that the client will not attempt to reconnect -->
  <!-- to the server after connection is lost                       -->
  <server-reconnect client-write="yes">0</server-reconnect>
</client>

<!-- This line is required. An empty element specifies the default "none" -->
<authentication/>

<!-- This line is optional. Web monitoring can be a valuable feature -->
<web-monitor>*:8080</web-monitor>

<!-- Enables and configures the UM transport statistics monitoring functionality. -->
<!-- See the Informatica Ultra Messaging Concepts Guide.      -->
<monitor object="context" interval="5">
  <transport module="lbm" options=""/>
  <format module="csv" options=""/>
  <application-id xml:space="preserve"></application-id>
</monitor>

<topics>
  <topic pattern="BEW.xyz.*" type="PCRE">
    <!-- This configures a message queue for a wildcard topic pattern. -->
    <umds-attributes>
      <option type="umds-receiver" name="receiver-queue-type"
          value="topic" />
      <option type="umds-receiver" name="topic-queue-size-limit"
          value="200000" />
    </umds-attributes>
  </topic>
  <topic pattern="BEW.xyz" type="direct">
    <!-- This directs the topic, BEW.xyz, to use the default message queue. -->
    <umds-attributes>
      <option type="umds-receiver" name="receiver-queue-type"
          value="default" />
      <option type="umds-receiver" name="topic-queue-size-limit"
          value="200000" />
```

```
      </umds-attributes>
    </topic>
    <topic pattern="wild.*" type="direct">
      <!-- This is a direct match for a wildcard to set explicit wildcard and receiver attributes
      <umds-attributes>
        <option type="lbm-wildcard-receiver" name="resolver_no_source_linger_timeout"
            value="30000"/>
        <option type="lbm-receiver" name="use_late_join" value="0" />
        <option type="umds-receiver" name="ignore-unique-receiver-attributes"
            value="1" />
      </umds-attributes>
    </topic>
  </topics>
</daemon>
</umds-daemon>
```

8.6.4 Sample UM Configuration File

The following sample UMS configuration file contains override values used by the UMDS Server for the specified UMS options. You use the UMDS Element "<lbm-config>" to specify a file such as this.

```
context transport_tcp_receiver_socket_buffer   4000000
source transport_tcp_sender_socket_buffer      4000000

context transport_lbtrm_receiver_socket_buffer 4000000
context transport_lbtrm_source_socket_buffer   4000000

context transport_lbtru_receiver_socket_buffer 4000000
context transport_lbtru_source_socket_buffer   4000000

context mim_implicit_batching_minimum_length   8192

context transport_lbtrm_data_rate_limit        400000000
context transport_lbtrm_retransmit_rate_limit  40000000
```

8.6.5 Sample Authentication File

The following sample authentication file specifies settings for a master application, a set of users who can only monitor prices, a second set of users who can post trades and monitor prices, and settings for an administrative user. You specify this file with the UMDS Element "<authentication>".

```
<?xml version="1.0" encoding="UTF-8" ?>
<umds-authentication-basic version="1.0">

  <application name="master_app">
    <client>
      <!-- Allow application "master_app" to reconnect and give ability to override this setting
      <server-reconnect client-write="yes">0</server-reconnect>
    </client>
  </application>
```

```
  <!-- users allowed to only monitor prices...except when using master_app -->
  <user name="john_doe" password="id1" />
  <user name="jane_doe" password="id2" />
  <user name="jim_doe" password="id3" />
  <user name="jackie_doe" password="id4" />
  <user name="john_smith" password="id5" />
  <user name="jane_smith" password="id6" />
  <user name="jim_smith" password="id7" />
  <user name="jackie_smith" password="id8" />
  <!-- This list could be 1,000's of users -->

  <!-- users allowed to monitor prices and post trades -->
  <user name="rob_smith" password="priv1"> </user>
  <user name="rose_smith" password="priv2"> </user>
  <user name="rod_smith" password="priv3"> </user>
  <!-- This list could be 100's or even 1,000's of users -->

  <user name="patel" password="admin">
    <client>
      <!-- allow this user to override any setting -->
      <server-ka-interval client-write="yes">5000</server-ka-interval>
      <client-ka-threshold client-write="yes">12000</client-ka-threshold>
      <client-ka-interval client-write="yes">9000</client-ka-interval>
      <server-ka-threshold client-write="yes">20000</server-ka-threshold>
      <server-rcvbuf client-write="yes">524288</server-rcvbuf>
      <server-sndbuf client-write="yes">524288</server-sndbuf>
      <server-nodelay client-write="yes">0</server-nodelay>
      <client-rcvbuf client-write="yes">524288</client-rcvbuf>
      <client-sndbuf client-write="yes">524288</client-sndbuf>
      <client-nodelay client-write="yes">0</client-nodelay>
      <server-reconnect client-write="yes">0</server-reconnect>
    </client>

  </user>
  <!-- This list would probably be limited -->

</umds-authentication-basic>
```

Chapter 9

UMDS Log Messages

Umds-10372-10: unable to create umds cfg stat group: s	Failure when creating daemon monitor umds config stats group	Verify all the attributes in the daemon-monitor section of the xml file are correct
Umds-10372-11: unable to create memory stat group: s	Failure created while creating daemon monitor malloc info stat group	Verify all the attributes in the daemon-monitor section of the xml file are correct
Umds-10372-12: unable to schedule timer for gateway config stat group call back, s	Error setting up UMDS config stat group callback timer	Contact Informatica Support.
Umds-10372-13: unable to schedule timer for umds config stat group call back, s	Failure setting up gateway config stat group callback timer	Contact Informatica Support.
Umds-10372-14: umdsd_publish↩ _cfg: s	UM was unable to publish dmon message.	Contact Informatica Support.
Umds-10372-15: error trying to publish config record	Failure publishing config record	Contact Informatica Support.
Umds-10372-16: error reading memory info record	Error attempting to read daemon stats memory info record	This is an information message only.
Umds-10372-17: unable to schedule timer for memory stat group callback: s	Failure creating memory stat group callback timer	Contact Informatica Support.
Umds-10372-18: error reading malloc info record	Error attempting to read daemon stats malloc info record	Contact Informatica Support.
Umds-10372-19: unable to schedule timer for memory stat group call back, s	failure scheduling timer for memory stat group callback timer	Contact Informatica Support.
Umds-10372-1: pointer to stats_↩ info is NULL: s	pointer to stats_info is NULL	Contact Informatica Support.
Umds-10372-20: Error reading memory record	Error reading memory record	Contact Informatica Support.
Umds-10372-21: worker id number is out of range	Worker number is out of range	Contact Informatica Support.
Umds-10372-22: worker_list pointer is NULL	worker_list pointer is NULL	Contact Informatica Support.
Umds-10372-23: stats_info pointer is NULL	stats_info pointer is NULL	Contact Informatica Support.
Umds-10372-24: unable to schedule timer for gateway config stat group call back, s	Error setting up UMDS config stat group callback timer	Contact Informatica Support.

Umds-10372-25: unable to schedule timer for gateway config stat group call back, s	Error setting up UMDS config stat group callback timer	Contact Informatica Support.
Umds-10372-26: No workers defined in umdsd_dstat_allworker_↩ stat_grp_create	No workers defined	Verify that the number of workers defined in the xml file is > 0
Umds-10372-27: error initializing a worker stat object	Error initializing a worker stat object	Contact Informatica Support.
Umds-10372-28: unable to schedule timer for worker stat group call back, s	failure scheduling timer for worker stat group callback timer	Contact Informatica Support.
Umds-10372-29: unable to schedule timer for worker stat group call back, s	failure scheduling timer for worker stat group callback timer	Contact Informatica Support.
Umds-10372-2: unable to create context attributes: s	Failure while creating context attributes	Contact Informatica Support.
Umds-10372-30: Worker ID number d is out of range	Worker ID is out of range	Contact Informatica Support.
Umds-10372-31: error reading worker stats	Failure reading worker stats	This is an information message only
Umds-10372-32: connection status read failed	Failure reading connection stats	This is an information message only
Umds-10372-33: umdsd_dstat_↩ send_thread_main: Unable to publish message of type d: s	UM was unable to publish dmon message.	Contact Informatica Support.
Umds-10372-34: invalid UMDS dmon message [s] from s [s]	UM dmon received an invalid/corrupted immediate message.	Verify that messages sent on the request port are valid.
Umds-10372-35: UMDS dmon failed to send error response [s]	UM could not respond to a dmon immediate message.	Contact Informatica Support.
Umds-10372-36: UMDS dmon failed to send success response [s]	UM could not respond to a dmon immediate message.	Contact Informatica Support.
Umds-10372-37: UMDS dmon received control message exceeding 255 bytes	UM daemon monitor received an invalid control message exceeding 255 bytes.	Verify that messages sent on the control channel are $<=$ 255 bytes.
Umds-10372-38: UMDS dmon failed to send error response [s]	UM could not respond to a dmon immediate message.	Contact Informatica Support.
Umds-10372-39: UMDS received unknown lbm_msg_t type x [s][s]	UM daemon monitor received unknown lbm_msg_t type.	Stop the source of unknown messages to the daemon monitor.
Umds-10372-3: lbmaux_context↩ _attr_setopt_from_file() failed, s	Failure while setting up extra config opts for UMDS daemon monitor	Check attributes in "lbm-config" config file specified in the "daemon-monitor" section of the UMDS's xml file
Umds-10372-40: error from umdsd_dstat_mallinfo_stat_grp_↩ snapshot()	Failure reading malloc info stat group record	Contact Informatica Support.
Umds-10372-41: error from umdsd_cfg_dstat_stat_grp_↩ snapshot()	Failure reading config info stat group record	Verify that the xml file has not been removed since starting the daemon.
Umds-10372-42: error from umdsd_dstat_workers_snapshot()	Failure reading worker stat group record	This is an information message only.
Umds-10372-43: error from umdsd_dstat_memory_stat_grp↩ _snapshot()	Failure reading malloc info stat group record	Contact Informatica Support.

Umds-10372-44: error from umdsd_dstat_cfg_stat_grp_↩ snapshot()	Failure reading config info stat group record	This is an information message only
Umds-10372-45: No connection ID specified for connection snapshot	No connection ID specified for the connections snapshot	Verify that a valid connection ID was specified
Umds-10372-46: error from umdsd_dstat_conn_snapshot()	Failure reading connection stat group record	This is an information only message.
Umds-10372-47: bad worker ID returned from umdsd_getworkerID↩ _fromstring()	The worker ID is invalid	Verify that the worker ID is within the range of Workers specified U↩ MDS XML file
Umds-10372-48: error from umdsd_dstat_workers_snapshot()	Failure reading worker stat group record	This is an information only message
Umds-10372-49: error returned from umdsd_dstat_setinterval()	Failure changing the publishing interval for config stat group	Contact Informatica Support.
Umds-10372-4: lbm_context_attr↩ _setopt() failed, s	Failure setting up attributes for daemon monitor remote control handler	Contact Informatica Support.
Umds-10372-50: error returned from umdsd_dstat_setinterval()	Failure changing the publishing interval for mallinfo stat group	Contact Informatica Support.
Umds-10372-51: bad worker ID returned from umdsd_getworkerID↩ _fromstring()	The worker ID is invalid	Verify that the worker ID is within the range of workers specified in the UMDS XML file
Umds-10372-52: error returned from umdsd_dstat_setinterval()	Failure changing the publishing interval for worksum stat group	Contact Informatica Support.
Umds-10372-53: invalid command s	Attempt to obtain snapshot of record for invalid stat group	Contact Informatica Support.
Umds-10372-54: unable to schedule timer	Failure to schedule callback timer	Contact Informatica Support.
Umds-10372-55: Worker ID d is out of range	Worker ID out of range	Contact Informatica Support.
Umds-10372-56: Unable to start daemon stats monitor	unable to start daemon stats monitor for UMDS	Verify all daemon monitor related attributes are correct in xml and config files
Umds-10372-57: NULL webmon pointer	NULL webmon pointer	Contact Informatica Support.
Umds-10372-58: monitor section lbm-config must have a value	Expecting a string that contains the path to the config file.	Please specify a string that contains the path to the config file.
Umds-10372-59: monitor section xml-config must have a value	Expecting a string that contains the path to the config file.	Please specify a string that contains the path to the config file.
Umds-10372-5: unable to create context attributes: s	Failure creating lbm context for daemon stats monitor	Contact Informatica Support.
Umds-10372-60: lbmaux_src_↩ topic_attr_setopt_from_file() failed, s	Failure while setting up extra config opts for UMDS daemon monitor source object	Check attributes in "lbm-config" config file specified in the "daemon-monitor" section of the UMDS's xml file
Umds-10372-6: unable to create src topic attributes: s	Error creating source attributes for daemon stats monitor	Contact Informatica Support.
Umds-10372-7: unable to alloc src topic: s	Error allocating src topic for daemon stats monitor	Contact Informatica Support.
Umds-10372-8: unable to create src: s	Error creating source for daemon stats monitor	Contact Informatica Support.
Umds-10372-9: unable to allworker stat group	Failure returned while creating allworker stats group	Verify all the attributes in the daemon-monitor section of the xml file are correct

Umds-10633-1: umdsd_main↩ : Daemon setup failed. Exiting umdsd.	There was a failure trying to setup the UMDS Daemon. The daemon cannot continue and exits.	Check previous errors and correct appropriately.
Umds-10759-1: Unable to create daemon stats sender thread: s	Failure to create sender thread for daemon stats	Contact Informatica Support.
Umds-10759-2: umdsd_dstat_↩ send_thread_create() error creating TL queue: s	Unable to create two-lock queue	Contact Informatica Support.
Umds-10759-3: umdsd_dstat_↩ send_thread_main(): Error while dequeueing	Error while dequeueing from TL queue	Contact Informatica Support.
Umds-4892-1: Attempt to set use↩ _late_join failed for attrs p err(d)	ERROR: Attempt to turn use_late↩ _join ON in the rcvr attrs failed.	
Umds-4892-2: Attempt to set ume_use_store failed for attrs p err(d)	ERROR: Attempt to turn ume↩ _use_store ON in the rcvr attrs failed.	
Umds-4892-3: Error creating Topic Queue: No Hash function found.	FATAL: A hash function is required to create a Topic Queue. This is set from the resolver_string_↩ hash_function in the topic queue map init function.	Check that the Hash function for Topic Resolution has been set correctly.
Umds-4892-4: Attempt to get ume_session_id failed or session id is zero: lu	ERROR: Trying to create a persistent receiver but the session id is either zero, or reading the attribute failed.	Check that session ID sent by the UMDS client is not zero
Umds-4892-5: Attempt to set ume_explicit_ack_only failed for attrs p err(d)	ERROR: The attempt to set ume↩ _explicit_ack_only (to ON) failed while creating a persistent receiver.	
Umds-5688-5609: umdsd_↩ worker_api_mim_loss_advisory: worker(p<d>) not running	The indicated worker is not in the RUNNING state	Contact Informatica support with all relevant log files
Umds-5688-5617: PCRE exec [s][s][d] error d	An error occurred while trying to match the pattern listed in the first bracketed expression. The topic string attempting to be matched is supplied as the second bracketed expression, and its length is supplied as the third bracketed expression. The error that occurred was internal to PCRE, and the error code is listed in the PCRE documentation for return values of pcre_exec.	
Umds-6033-635: Ultra Messaging UMDS server version " UMDS_V↩ ERSION " Build s, s (s)	Reports the version of UMDS, build time and date, and version of the underlying UM library.	No resolution, this in formation is provided for audit and debugging purposes.
Umds-6033-637: umdsd_main↩ : webmon interface not found (s)	The interface specified for the web monitor could not be found.	Review the setting in the <web-monitor> tag in the server's xml configuration file.
Umds-6033-638: umdsd_main↩ : client interface not found (s)	The interface specified for the client connections could not be found.	Review the setting for the bind-addr attribute in the <server> tag in the server's xml configuration file.

Umds-6033-641: umdsd_main: Error opening pidfile 's' (s)	Opening (creating) the pid file failed.	The error message includes the OS error message associated with the open attempt. Check that files can be created in the target directory and that the device is not full.
Umds-6033-656: umdsd_worker↩ _cont_sending_cntl: sendb header error (s)	Sending a UMDS Control message to the client resulted in an UM error.	A description of the UM error is included in the message text.
Umds-6033-657: umdsd_worker↩ _cont_sending_data: sendb data error (s)	Sending on the client socket encountered an error.	Included in the message text is a description of the particular error encountered.
Umds-6033-664: umdsd_↩ webmon_api_create: failed to init web server (ip=s, port=s)	The web monitor subsystem failed to start.	The web server library will have reported additional details to the console.
Umds-6033-685: s: worker p<d> connection p<d> invalid conn_↩ state (d)	The indicated state for the client connection is inappropriate for the requested operation.	The client connection will be deleted and if configured, the client will retry. If this error repeats, call Informatica support with all relevant server and client log files.
Umds-6033-705: umdsd_worker↩ _internal_cmd_del_conn: worker p<d> connection p<d> (s:u) deleted, bytes_in=lld, bytes_↩ out=lld	The indicated connection has been deleted	No resolution is required.
Umds-6033-706: umdsd_↩ worker_internal_cmd_mim_loss↩ _advisory: worker(p<d>) not running	The indicated worker is not in the RUNNING state	Contact Informatica support with all relevant log files
Umds-6033-708: umdsd_worker↩ _keepalive_tmr_cb: worker(p<d>) not running	The indicated worker is not in the RUNNING state	Contact Informatica support with all relevant log files
Umds-6033-709: umdsd_worker↩ _keepalive_tmr_cb: send_↩ period=d, disconnecting worker p<d> connection p<d>	The keep alive state has been PE↩ NDING for too long; it is being disconnected as unresponsive.	This can occur if the client application is spending long periods of time in any of the library call back functions and preventing the client sid socket from being read.
Umds-6033-710: umdsd_worker↩ _keepalive_tmr_cb: rcv_period=d, disconnecting worker p<d> connection p<d>	The keep alive timer has expired for the indicated worker connection; it is being disconnected as unresponsive.	This can occur if the keep alive threshold and intervals are not appropriate for the connection or if the client application is spending long periods of time in any of the library call back functions.
Umds-6033-711: umdsd_worker↩ _api_conn_add: worker(p<d>) not running	The indicated worker is not in the RUNNING state	Contact Informatica support with all relevant log files
Umds-6033-712: umdsd_worker↩ _api_conn_add: worker p<d> waiting to add new connection	The client request to add a connection to this worker is still pending.	The main UM context thread is unusually busy at this time but will eventually serve this request.
Umds-6033-715: umdsd_worker↩ _api_delete: quit skipped (ctx=p, thrd_running=d)	The context or worker thread has already shutdown	Shutdown is already in progress

Umds-6033-716: umdsd_worker↩ _api_delete: error joining worker (p<d>) thread (d)	An error occurred joining the worker thread during shutdown.	It is likely this is a result of multiple shutdown requests. However if this error is seen on multiple occasions, please report it along with any applicable configuration and log files to GCS.
Umds-8218-1: s: error: 's', appl_↩ name='s'	The UMDS client failed to authenticate.	Check authentication credentials and server auth configuration.
Umds-8366-1: Unknown receiver type deleting umdsd_rcv <p>	An unknown receiver type was encountered while deleting a UMDS receiver object.	This is an internal error and should be reported to customer support; please include all appropriate version numbers (UM and UMDS), associated configuration files and any other pertinent details.
Umds-8366-2: failed to free umds unique receiver p	The UMDS server (umdsd) encountered an error deleting the UM receiver associated with the umds_rcv object	This is an internal error and should be reported to customer support; please include all appropriate version numbers (UM and UMDS), associated configuration files and any other pertinent details.
Umds-8366-3: failed to free umds wc receiver p	The UMDS server (umdsd) encountered an error deleting the UM wildcard receiver associated with the umds_rcv object	This is an internal error and should be reported to customer support; please include all appropriate version numbers (UM and UMDS), associated configuration files and any other pertinent details.
Umds-8366-4: Unknown receiver type deleting umdsd_rcv <p>	An unknown receiver type was encountered while freeing a UMDS receiver object.	This is an internal error and should be reported to customer support; please include all appropriate version numbers (UM and UMDS), associated configuration files and any other pertinent details.
Umds-8366-5: Unknown receiver type deleting umdsd_rcv <p>	An unknown receiver type was encountered while deleting a UMDS receiver object.	This is an internal error and should be reported to customer support; please include all appropriate version numbers (UM and UMDS), associated configuration files and any other pertinent details.
Umds-8406-1: umdsd_stats↩ _queue_internal_cmd_cb: src create before delete. is <d>	The webmon statistics subsystem got a source create for an already existing source id (the intended src structure was not NULL in the source array).	It is possible for creation and deletion to happen out of order.
Umds-8406-2: umdsd_stats↩ _queue_internal_cmd_cb: src delete before create. id <d>	The webmon statistics subsystem got a source delete for an already deleted source id (the intended src structure was NULL in the source array).	It is possible for creation and deletion to happen out of order.
Umds-8406-3: umdsd_stats↩ _queue_internal_cmd_cb: rcv create before delete. id <d>	The webmon statistics subsystem got a receiver create for an already existing receiver id (the intended rcv structure was not NULL in the receiver array).	It is possible for creation and deletion to happen out of order.

Umds-8406-4: umdsd_stats↩ _queue_internal_cmd_cb: rcv delete before create. id <d>	The webmon statistics subsystem got a receiver delete for an already deleted source id (the intended rcv structure was NULL in the receiver array).	It is possible for creation and deletion to happen out of order.
Umds-8408-1: umdsd_worker_↩ api_delete: waiting for worker p<d> to quit	The request to remove a worker is still pending.	The worker thread is unusually busy at this time but will eventually serve this request.
Umds-8447-1: umdsd_worker_↩ handle_blocked_msg: Parse Error	The client connection parser encountered an unrecoverable error.	This is an internal error, if it recurs, please report it along with any relavant log files to GCS.
Umds-8499-1: LBM error while sending request: s	LBM returned an unhandled error code.	The LBM error code is given in the log message. Please refer to the LBM error code.
Umds-8499-2: LBM error while sending message: s	LBM returned an unhandled error code.	The LBM error code is given in the log message. Please refer to the LBM error code.
Umds-8499-3: LBM error while sending response: s	LBM returned an unhandled error code.	The LBM error code is given in the log message. Please refer to the LBM error code.
Umds-8519-1: Attempt to send without a valid source created↩ : conn p	The UMDS client has sent a message before the umdsd server has created the corresponding source.	This will result in the loss of the client message. Please notify G↩ CS with all suitable logs (client and server).
Umds-8519-2: Attempt to send without a valid source created↩ : conn p	The UMDS client has sent a message before the umdsd server has created the corresponding source.	This will result in the loss of the client message. Please notify G↩ CS with all suitable logs (client and server).
Umds-8519-3: Attempt to send without a valid source created↩ : conn p	The UMDS client has sent a message before the umdsd server has created the corresponding source.	This will result in the loss of the client message. Please notify G↩ CS with all suitable logs (client and server).
Umds-8542-1: umdsd_worker_↩ client_src_create: transport <s> not allowed, using TCP instead	The UMDS server configuration file specified the use of LBT-SMX as a transport type, which is not supported. The server will use TCP instead.	Change the server configuration file to specify one of the supported transport types.
Umds-8544-11: Error creating source: <d>: s	An error occurred creating the request source.	The text of the warning will provide additional information for the resolution of the problem.
Umds-8544-1: Error creating source: <d>: s	An error occurred creating the request source.	The text of the warning will provide additional information for the resolution of the problem.
Umds-8697-1: umdsd_worker_↩ api_create: Error creating R↩ O-Contex while creating worker p<d>	Creating the reactor only context for a worker failed.	This fatal error is usually due to specifying too many workers.
Umds-8697-2: umdsd_worker_↩ api_create: Error creating thread while creating worker p<d>	Creating the worker application thread failed.	This fatal error is usually due to specifying too many workers.
Umds-8753-1: Attempt to send without a valid source created↩ : conn p	The UMDS client has sent a message with a wrong or garbage tidx; either the client is buggy or t he server is receiving garbage data.	This will result in the loss of the client message. Please notify G↩ CS with all suitable logs (client and server).

Umds-8753-2: Attempt to send without a valid source created↩ : conn p	The UMDS client has sent a message with a wrong or garbage tidx; either the client is buggy or the server is receiving garbage data.	This will result in the loss of the client message. Please notify G↩ CS with all suitable logs (client and server).
Umds-8757-1: s: malformed connect capabilities	The UMDS client sent a malformed capabilities string.	Make sure the client and server versions are compatible and that data from another application isn't being erroneously sent to the UM↩ DS server.
Umds-8796-100: Error creating umdsd_rcv_topic: <d>: s	An internal error occurred while creating a receiver in the UMDS server.	Contact Informatica support.
Umds-8894-1: umdsd_worker↩ _internal_cmd_add_sock↩ : worker(p<d>) not running	The indicated worker is not in the RUNNING state	Contact Informatica support with all relevant log files
Umds-8894-2: umdsd_worker_↩ internal_cmd_add_sock: worker p<d> connection p<d> (s:u) created	Notification that a new client connection has been added.	No resolution is required.
Umds-8896-1: umdsd_webmon_↩ api_create: failed to init web server (ip=s, port=s)	The web monitor subsystem failed to start.	The web server library will have reported additional details to the console.
Umds-8909-1: UMDS Permissions are no longer applied	Permissions are no longer supported in the UMDS server XML configuration file.	Remove any permissions sections from the server's XML config file.
Umds-8909-2: get_application↩ : UMDS Permissions are no longer applied	Permissions are no longer supported in the UMDS server XML configuration file.	Remove any permissions sections from the server's XML config file.
Umds-8909-3: get_user: UMDS Permissions are no longer applied	Permissions are no longer supported in the UMDS server XML configuration file.	Remove any permissions sections from the server's XML config file.
Umds-8947-1: Error creating umdsd_rcv_topic: <d>: s	An underlying regular receiver for a topic could not be created for a wildcard receiver.	This would usually imply an out of memory problem or some other internal error; contact Informatica support.