inteBroker™

Administrative User's Guide

inteBroker™

MQue Systems™

InteBroker™ Administrative User's Guide

2408 Green Street, Middleboro, MA 02356 USA
Sales: 508.380.8291

MQue Systems™

Systems that Plug-and-Share™

Introduction

The MQue Integration Suite

The MQue Integration Suite is a set of message middleware development and runtime components from MQue Systems that are designed for highly distributed, high-volume, and mission-critical eCommerce and business-to-business environments. These components deliver superior reliability, performance, portability and flexibility, while defining a new standard for ease of development and runtime administration.

InteBroker

The InteBroker message server is the heart of the MQue Integration Suite of middleware components. The server provides reliable asynchronous message delivery, and supports both point-to-point and publish / subscribe models of message routing. InteBroker employs a patented message queuing technology that delivers unprecedented performance and reliability. This100% Java™ middleware server runs on any Java 2 SE compliant platform, including Microsoft Windows™, and Linux.

Contacting MQue Support

MQue Systems offers a variety of support options. They include free services on our web site, as well as a number of maintenance and support packages. Contact your MQue representative, or email us at support@mque.com for more information.

Overview

Server Queues and Message Processing

Messages that are written to the sever contains a message header. The value of the *recipient* and *topic* fields in the message header determines the routing of the message. Every message contains a *topic* value. Messages may or may not contain a *recipient* value.

The message server maintains a unique FIFO (first-in-first-out) queue for each remote subscriber / recipient process.[1] A message that has a *recipient* identified in the header is appended to that recipient's queue. (This form of message routing is termed *point-to-point* routing, since it involves one sender and one recipient.) If a message has no recipient identifier, a copy of that message is appended to the queue of each subscriber that is currently registered for the topic defined in the message header. (This form of routing is termed *publisher-subscriber* routing, since a recipient "subscribes" to one or more message "topics.")

Subscribing and Unsubscribing

A subscriber / recipient agent will normally subscribe to one or more topics before connecting to the server as a subscriber. If a subscriber disconnects before explicitly "unsubscribing" to a topic, messages with that topic will continue to be appended to the subscriber's queue. As a result, a subscriber should never loose a message because of a communications failure. Additionally, whenever a subscriber process is shut down for routine maintenance, messages intended for that subscriber can be queued and available when the subscriber reattaches to the server.

[1] The terms "subscriber" and "recipient" are used synonymously in this document. The term refers to a remote application that has a unique identifier (name), and is able to connect to the server (via a TCP/IP connection) and fetch message in the order they were written by a message "publisher".

Installation

Preparation

InteBroker ships with Javasoft's Java Runtime Environment (JRE). Prior to installing the InteBroker message server, make certain that the JRE, or a more recent version of the Java 2 JRE, has been installed on the host computer. Java 2 Standard Edition (J2SE) is freely available from the Sun Microsystems Java web site. To download the latest version, go to Sun's web site at *http://java.sun.com*.

The Install Program

InteBroker should be installed in its own file directory. The setup program creates an InteBroker directory and several subdirectories.

Loading the Message Broker

Once you have run the installation program, InteBroker can be run as either

- a background task or
- a graphical user interface.

Background Mode

Running the server in background mode requires less memory than in graphical mode, and provides marginally better performance.

To start the broker in *background mode,* execute the `cbroker` command file:

- Unix/Linux

  ```
  intebroker/cbroker.sh
  ```

- Windows

  ```
  intebroker\cbroker.cmd
  ```

The next section describes how to run the server in graphical mode.

Graphical Mode Operations

Graphical mode operations allows the user to monitor and control the message server from a local console. Running a server in graphical mode involves first loading the server program, and then starting the server's muli-threaded message processor.

Loading the Server Program

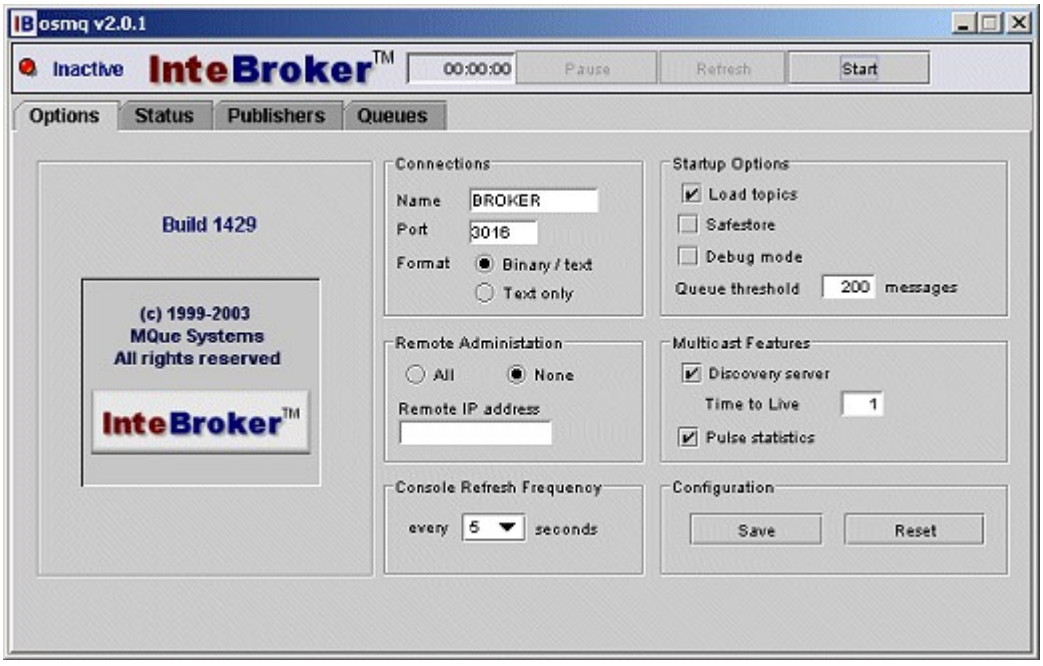
You can load the message server by clicking on the ***Start Broker*** icon on your windows desktop. You can also load the message server by running the appropriate jar or executable file:

```
InteBroker/intebroker.jar
```

When the server's console window appears, you will see a flashing LED indicator on the tool bar which indicates that the server is *inactive* and has not been started. The server must be started before it will begin processing messages.

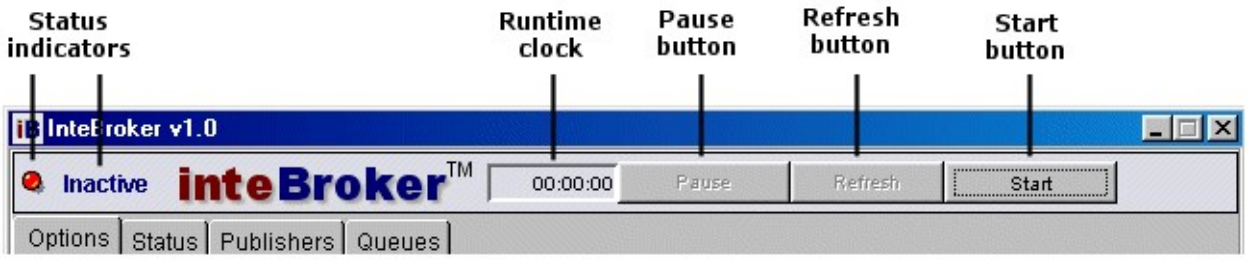
Starting the Server

To start the server, simply click the ***Start*** button on the horizontal tool bar.



The Main Tool Bar

The server window contains a horizontal toolbar and a set of tabbed panels. The tool bar, which provides general status information, is used to perform key control functions.



Status indicators include the LED status indicator and a status label. These indicators identify the server's status as *inactive* (flashing red), *active* (solid green), *paused* (flashing yellow), or *standby* mode (flashing yellow).

Runtime clock displays the lapsed time since the server was started.

Pause button is a toggle button used to temporarily halt processing of published messages. This button is not enabled until the server is started.

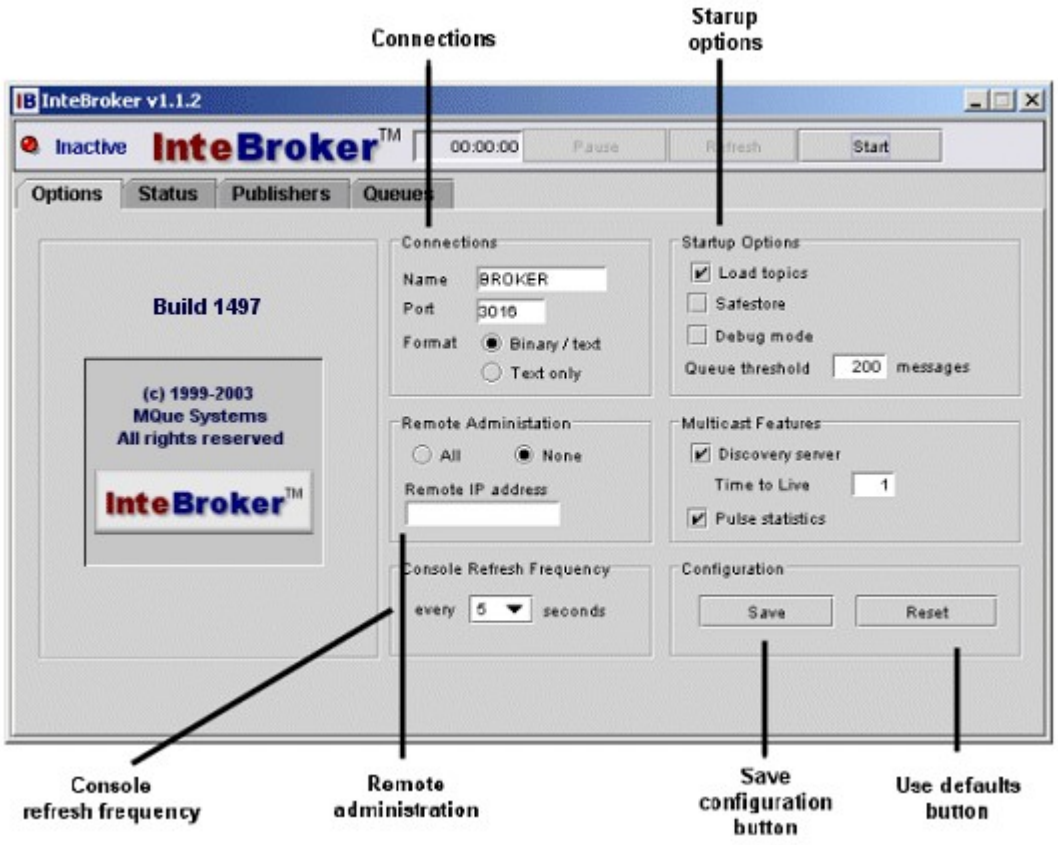
Refresh button allows the user to refresh statistical information that is presented in *the Publishers Panel*, *Queues Panel* and *Topics Panel*. This button is not enabled until one of these panels has focus.

Start/stop button is a toggle button used to both start the server and shut it down.

Server Panels

Options Panel

The *Options Panel* is presented to the user when the server console window appears. The *Options Panel* displays a set of controls with the default startup parameters, and these parameters should normally be used.



Connections display the client attachment options related to dynamic discovery, server port address, and message content type:

- **Name** defines the name that will be used by clients to dynamically locate the InteBroker server. The name that appears when the broker is loaded is the default server name, and you should not change the name unless you want to run a server that overrides the default names used by publishers and subscribers.

- **Port** is the port number that the server will use to accept publisher and subscriber connections. This ID should be changed only if it conflicts with another service that uses the same port number.

- **Binary/text** and **Text only** formats are two modes of message transfer**.**. Bimary/text mode supports messages with both binary and non-binary (text and dataset) content, whereas text-only mode supports only text and dataset content.

Startup options display optional runtime functions that can be selected before starting the Intebroker server:

- **Discovery Server** identifies whether a discovery server process will listen for client where-is discovery requests and respond with the Intebroker server's IP address and port number

- **Load topics** loads the topic table that was automatically saved by the server when it was last shut down.

- **Pulse statistics** generates a periodic "*heartbeat"* of statistical information over an IP multicast channel. This option must be enabled if you want to remotely monitor the status of the message server from a *Remote Console adapter.*

- **Safestore** is a local time-stamped log file of every message received by the server. Normally, this feature should not be activated. Instead, a comprehensive log file should be created by running the Silent Auditor™ adapter on a remote computer.

- **Debug mode** creates a local trace log of program execution. Do not generate a trace log unless you are instructed to do so by a member of the BSG support staff. The trace log is a very detailed log file, and running the server with trace turned *on* will significantly impact message throughput and disk utilization.

- **Queue threshold** defines the maximum number of messages that will be queued in memory for a named subscriber. Whenever the number of messages exceeds this threshold, they are saved to disk.

Console refresh frequency is the frequency (in seconds) at which statistic on the *Status Panel* will be updated. Valid rates are 0, 5, 10 and 15. Setting the rate to zero disables the refresh function.

Remote administration defines the remote workstations that will be permitted administrative access to the server. The opions include *All* workstations (not recommended), *None*, and the *Remote IP* address of a single workstation. (Remote IP should be a dotted IP address, such as "10.0.34.130*"*.) If a value is entered for Remote IP, the server will permit remote administration only from that address.

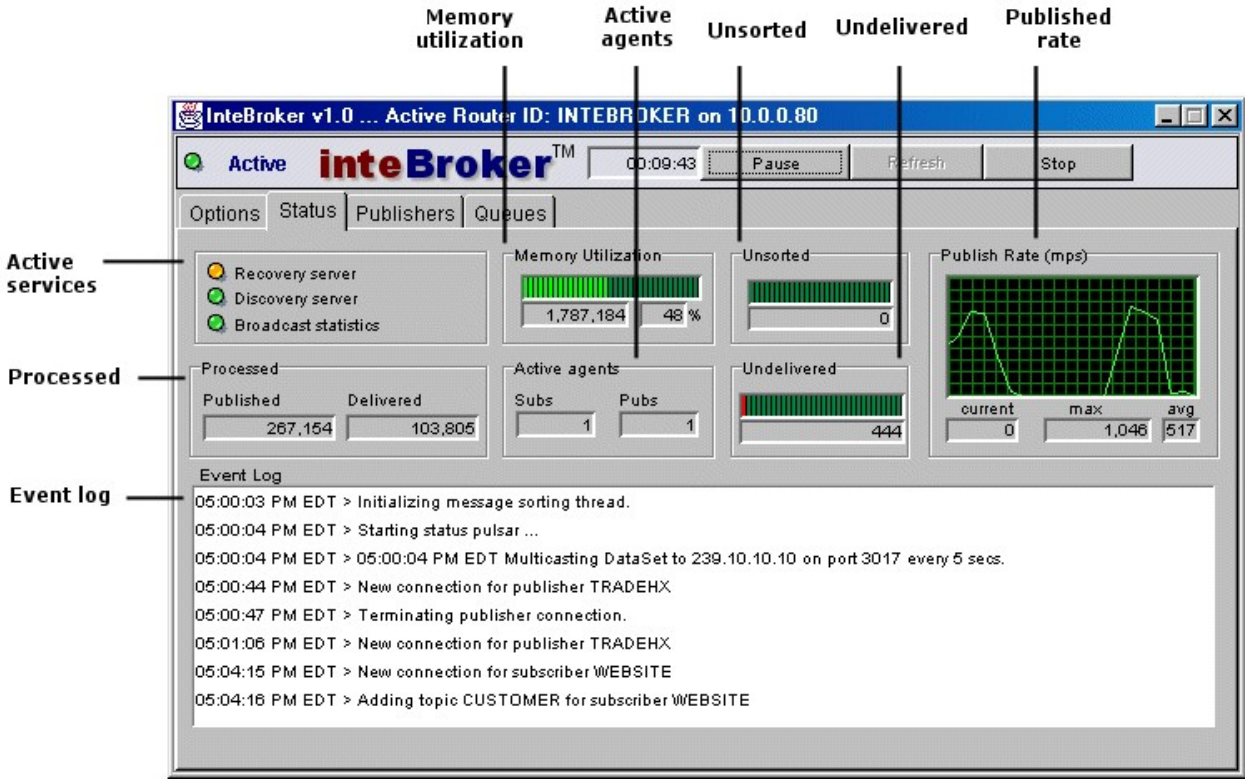
Configuration options

- **Reset** button reloads the standard Intebroker starup parameters.

- **Save** button saves the currently defined startup parameters as the new default set of parameters.

Status Panel

Once the server engine has completed the server startup process, the console displays the *Status Panel*. The *Status Panel* provides a great deal of information about the state of the message server.



Memory utilization displays the message server's current memory utilization, both as a size (number of bytes) and percent of available and allocated computer memory.

Active agents displays the number of publisher and subscriber agents that are currently attached to the server. A separate server thread handles the processing of each attached publisher or subscriber agent. The *Publisher Panel* and *Queues Panel* provide more detail regarding the individual publishers and subscriber activities.

Unsorted messages are those messages that have been received by the message server, but have not yet been sorted into subscriber queues.

Undelivered messages are those that are currently being held by the server in its various subscriber queues.

Publish rate displays the rate at which the server is processing published messages. The *Publish Rate* sub-panel displays the current rate, maximum rate, and average rate for the session in messages per second.

Active services displays three LED images that represent the status of three broker services. If an LED is green, the corresponding service is active.

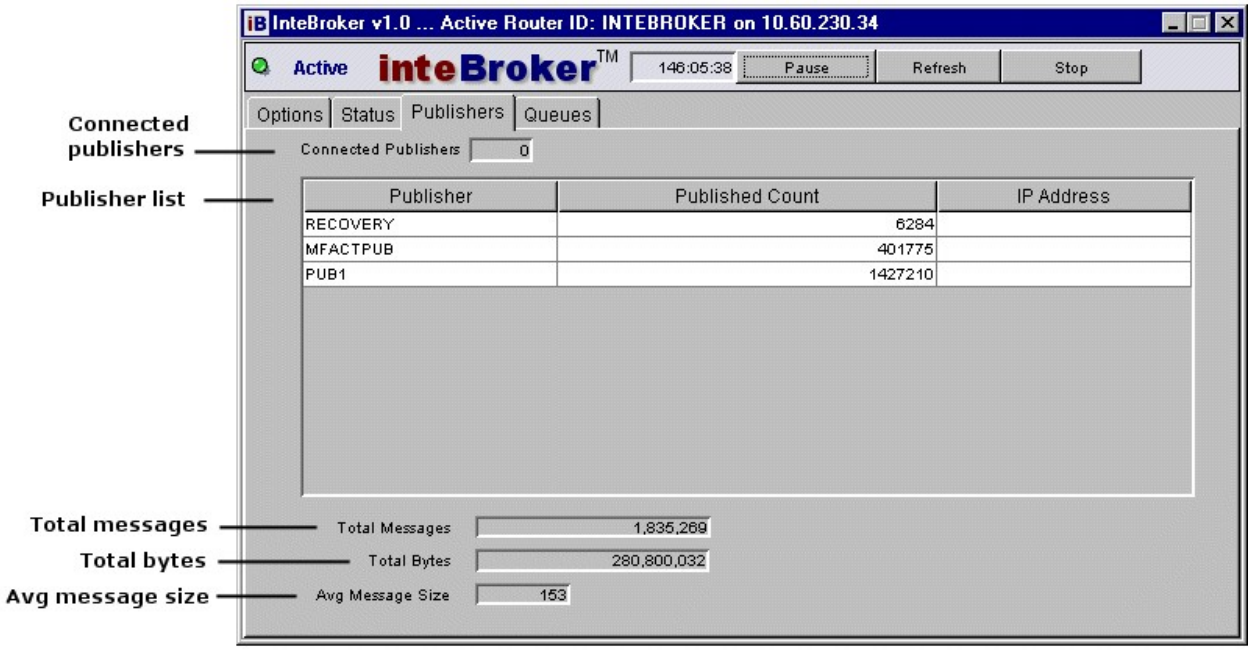
Processed displays a count of messages that are published or undelivered:

- **Published** displays the number of messages that have been written to the server during this session.
- **Delivered** displays the number of messages that have been delivered to the various remote subscribers during this session.

Event log displays a running chronology of events that include startup processing, topical subscription changes, and the attaching / detaching of the various publishers and subscribers.

Publishers Panel

The Publishers Panel provides detail and summary information about message publisher agents.



Connected publishers displays the number of the remote publisher agents that are currently connected to the server.

The panel's list box contains an entry for each publisher that is or has been connected during the session.

- **Publisher** is the remote publisher's unique identifier.
- **Published count** displays the number of messages that have been written by the publisher during this session.
- **IP Address** defines the publisher's host address. If there is no entry in this column, the publisher is not currently attached to the server.

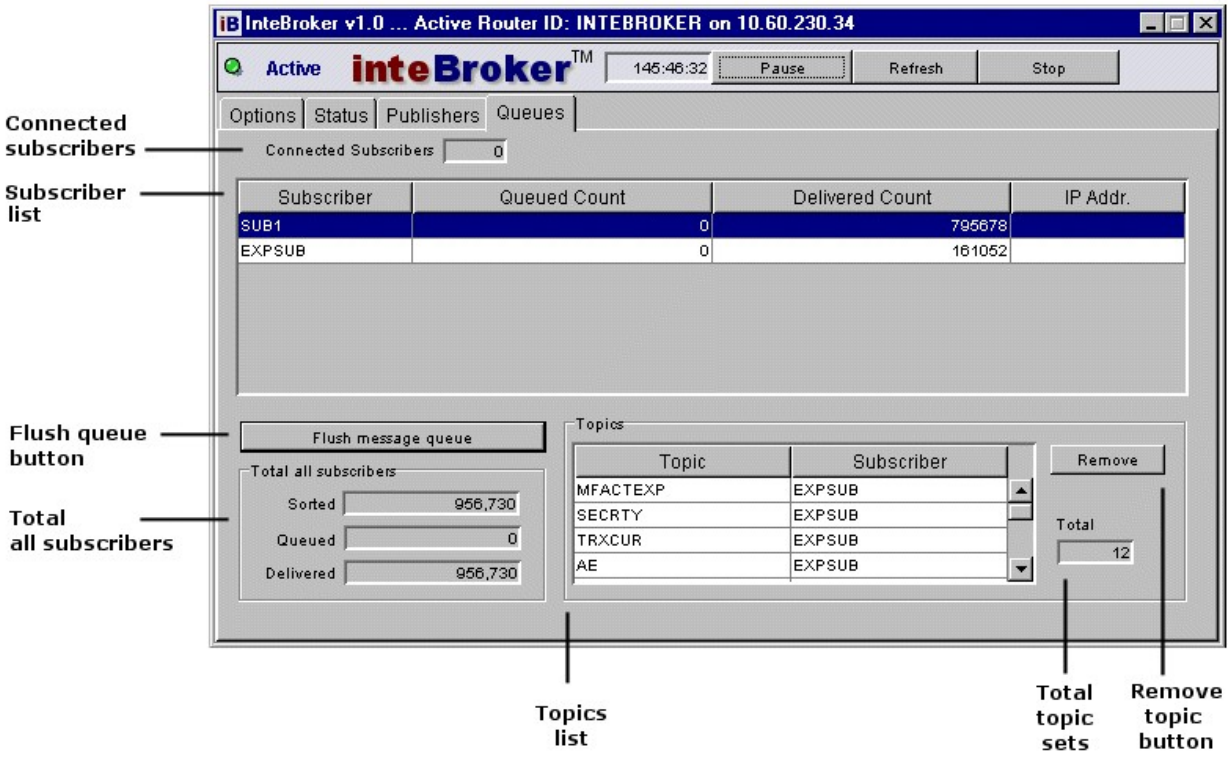
Total messages is the count of all messages that have been published during this session.

Total bytes displays the total number of bytes that have been written in all the messages that have been published.

Avg message size displays the average message size in bytes.

Queues Panel

The Queues Panel presents detailed and summary information about subscribers and subscriber queues.



Connected subscribers defines the number of remote subscribers that are currently attached to the server.

Subscriber list contains an entry for each subscriber that was registered as a topic subcriber or had messages queued during the session:

- **Subscriber** is the remote subscriber's unique identifier.
- **Queued count** is the number of messages currently being held in the subscriber's queue.
- **Delivered count** is the number of messages that have been successfully read by the remote subscriber agent.
- **IP address** is the subscriber's host address. If there is no value in this column, the subscriber is not currently attached to the server.

Flush queue button deletes all queued (undelivered) messages from the highlighed subscriber queue

Total all subscribers contains summary message totals:

- **Sorted** is the total number of messages that have been written to subscriber queues during the session.
- **Queued** is the total number of messages currently being held the various subscriber queues.
- **Delivered** is the total number of messages that have been read by to all subscribers.

Topics list is a sortable list of topic/subscriber pairs. The list reflects the currently registered topical subscriptions; each entry reflects a topical subscription by one subscriber.

- **Topic** defines a named topic associated with the corresponding subscriber. Clicking on the column heading sorts the list in topical order.
- **Subscriber** defines a named subscriber associated with the corresponding topic. Clicking on the column heading sorts the list in subscriber order.

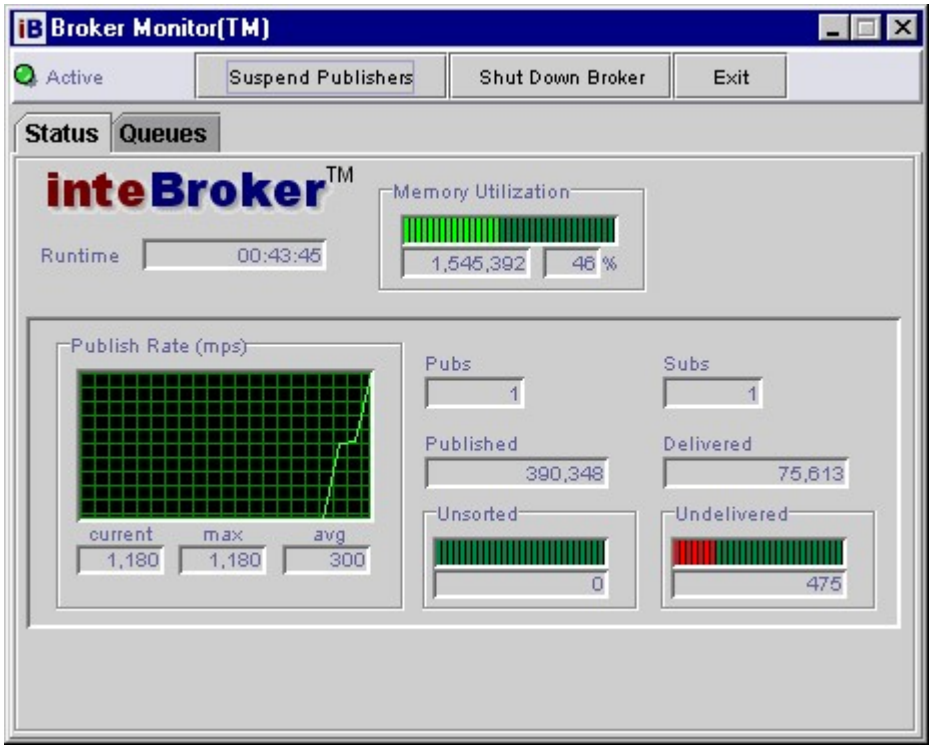
Total topic sets is the total number of active subscriber/topic subscriptions.

Remove topic button deletes the highlighted topic set.

Remote Monitors

The Remote Monitor captures *pulse statistics* that are generated by the server, and graphically displays the information on an administrative console. Additionally, the monitor continuously tests to ensure that the server is generating *heartbeats*. The monitor will begin flashing an alert message if the server stops transmitting these heartbeats. Server The Remote Monitor console also provides several key administrative functions if those functions have been enabled on the server's Options panel.

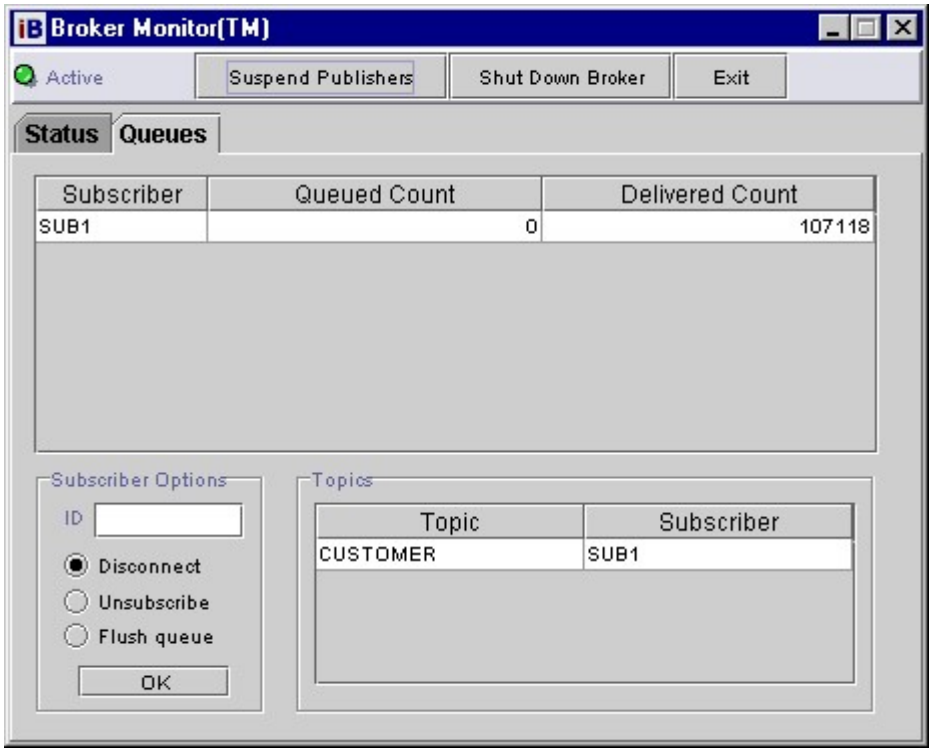
A monitor can be run on any computer that has IP multicast discovery access to the message:



Suspend Publishers acts like the ***Pause*** button on the server's tool bar, permitting the user to temporarily suspend and the resume accepting publisher messages

Shut Down Broker performs a normal shutdown of the remote message server.

Exit closes the monitor.



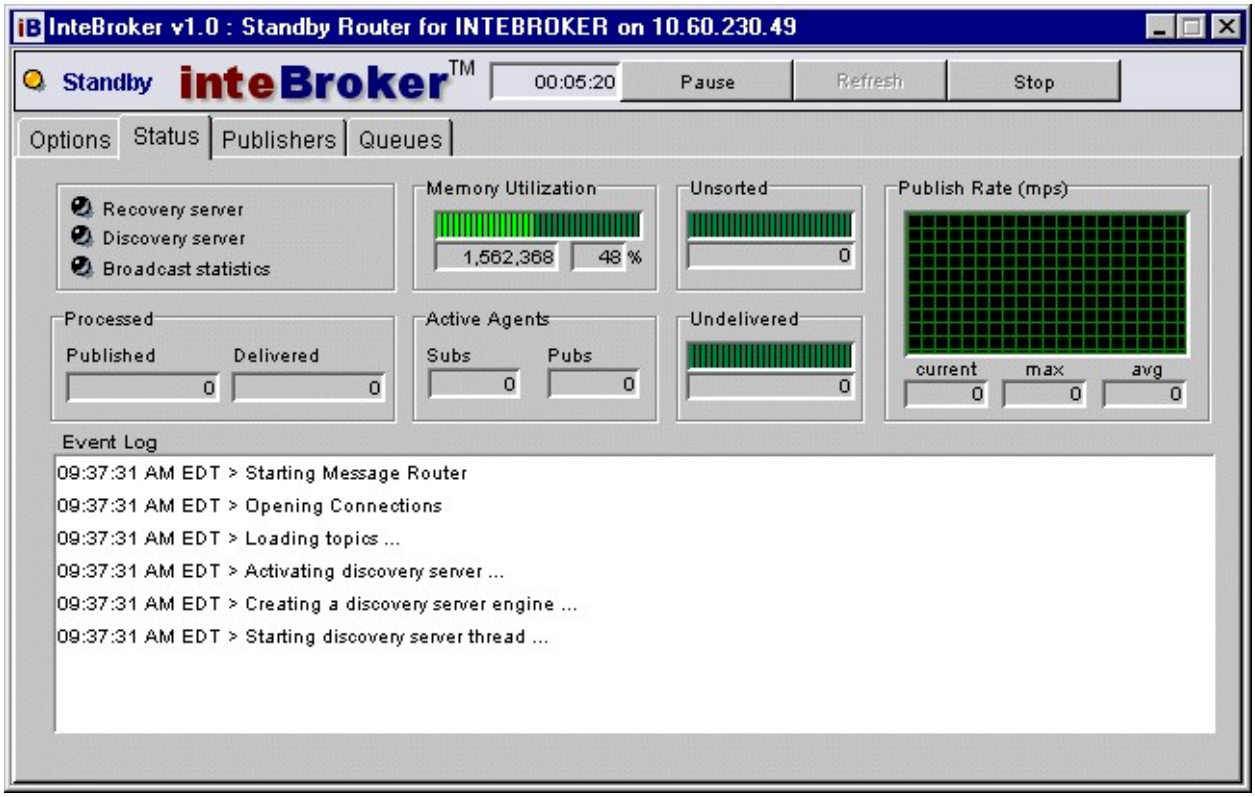
Subscriber Options permits an authorized user to modify the state of a subscriber, registered topics, or undelivered messages. To perform an action, the user enters the name of a subscriber, selection an action, and presses the "OK" button. The three supported actions are:

- **Disconnect** terminates the server's connection with the named subscriber.
- **Unsubscribe** terminates all topical subscriptions for the named subscriber.
- **Flush queue** deletes all undelivered messages that are currently queued for a named subscriber.

Standby Servers

A *standby* or *slave* server attaches to the *master* server as a "universal subscriber." A copy of each message that is written to the master is forwarded to the slave server, and the slave appends these messages to a local queue. When the slave server is shut down, it writes all the messages to a local hard drive. The messages are reload when the server is next started.

If the master server should ever fail, the slave can be shut down and then restarted as the master server. Assuming no other sever is running, the restarted server becomes the *master server*, and it will begin responding to all discovery requests from message publishers and subscribers, and accepting publisher and subscriber connections.



Server Features

Master and Standby Servers

When a server is started, it uses a dynamic discovery protocol (described below) to determine whether there is a message server already running on another host. One reason for performing this step is to ensure that only one server (the "master" server) responds to any client's dynamic discovery request. If a server determines through discovery that another server is active, the second server automatically becomes a standby message server, and goes into "slave" or "standby" mode." For details regarding standby server processing, see the section entitled *Standby Servers*.

Dynamic Discovery

Publishers and subscribers use dynamic discovery to locate the message server. The message server responds to discovery requests using MQue's Multi-Cast Naming Service (MCNS™) protocol, an important and distinguishing feature of InteBroker that is described in more detail in a subsequent section of this manual. The server responds to discovery requests by returning its IP address and server port address to the discovery client. The discovery client then uses this information to attach to the server as either a publisher or subscriber.

Initializing Table and Queues

InteBroker is a reliable store-and-forward message broker / router. After a server has determined that it is the master server, it performs a series of steps to rebuild its internal state, so that it reflects its state when it was last shut down. This involves rebuilding its topical subscription table and subscriber queues, and reloading previously undelivered messages into the subscriber queues. (The topical subscription table, queues and messages are automatically saved as a final step whenever a server is shut down.)

Communications and Sorting Threads

After the message server has loaded its tables and queues, it initiates two communications threads. The first thread begins listening on the published IP port, accepting socket connections from subscribers and publishers. (The published port is the port number identified in the *Options Panel*.) The second thread, the *discovery server* thread, begins listening for discovery requests. The server responds to a *discovery client* by transmitting the IP address and the port number where it is accepting connections.

Once the two communications threads have been started, the server's message sorting thread is started. The sorting thread is responsible for placing copies of all messages that are written to the server into appropriate subscriber queues.

At this point, initialization is completed, and the server begins processing messages.

Frequently Asked Questions

Q. Does InteBroker *support message group hierarchies* or *content-based routing*?

A. No, InteBroker does not support message group hierarchies or content-based routing.

Message Groups

It is rare that business organization can agree upon a common message hierarchy. This is because messages are essentially *normalized data entities*. Based on relational data theory, the relationship of one entity to another is relatively dynamic. There is seldom a fixed hierarchy of entity relationships.

Content-based Routing

Content-based message routing is a relatively static and centralized data distribution model. That model assumes that content-based routing results in fewer data transfers between the server and subscriber, and that the subscriber process benefits from not having to parse a message to determine its suitability. InteBroker and the BSG Integration Model delegate the task of content filtering to the subscriber. However, experience with this model reveals that the impact on overall traffic is often negligible.

For more detailed information regarding message normalization and the benefits of recipient-based routing, see the BSG Integration Model documentation.

Q Does InteBroker support a reliable mode of operations?

A InteBroker does not distinguish between reliable and unreliable modes of operations. Reliability is an inherent attribute of the server's design. MQue's MCNS dynamic discovery feature, its patented queue technology that supports the automated paging of over-committed queues, and its automated hot standby server features provide an environment of unprecedented reliability, without sacrificing performance.

Q What throughput performance can I expect from InteBroker?

A Throughput will vary based on several factors, including the message server's CPU speed, operating system, communications line speed, the ratio of publishers and subscribers, and the size of the messages being processed.

A recent benchmark of InteBroker on an a 1000MHz uni-processor running a Win2K client and connected to a 100 megabit Ethernet LAN revealed a sustained message throughput rate of 4600 published messages / second. The average message size was 500 bytes, and the benchmark included one remote publisher and three remote subscribers. The benchmark was conducted over a 10-day period, and resulted in over two billion messages being published and distributed.